<!doctype html>

<html lang="en-GB">

<head>

<meta charset="utf-8" />

<meta name="viewport" content="width=device-width,initial-scale=1" />

<title>MiniFit + BellyFit — Women's Nutrition & Belly-Fat Plan</title>

<link href="https://fonts.googleapis.com/css2?family=Inter:wght@300;400;600;700&display=swap" rel="stylesheet">

<style>

:root{

--bg:#fff6fb;

--card:#ffffff;

--muted:#6b7280;

--accent:#d946ef;

--accent-2:#f472b6;

--green:#10b981;

--danger:#ef4444;

--shadow: 0 8px 24px rgba(16,24,40,0.06);

--radius:12px;

}

\*{box-sizing:border-box}

body{

font-family:Inter, system-ui, -apple-system, "Segoe UI", Roboto, "Helvetica Neue", Arial;

margin:0;

background: linear-gradient(180deg,#fffaf6,#f0f7ff);

color:#0b1724;

-webkit-font-smoothing:antialiased;

-webkit-tap-highlight-color: transparent;

}

header{

display:flex;align-items:center;justify-content:space-between;padding:14px 18px;gap:12px;background:linear-gradient(90deg,var(--accent),#fb7185);color:white;

}

.brand{display:flex;gap:12px;align-items:center}

.logo{width:52px;height:52px;border-radius:10px;background:rgba(255,255,255,0.12);display:flex;align-items:center;justify-content:center;font-weight:700;font-size:20px}

main{max-width:1100px;margin:12px auto;padding:14px;display:grid;grid-template-columns:360px 1fr;gap:16px}

.card{background:var(--card);border-radius:var(--radius);box-shadow:var(--shadow);padding:12px}

.small{font-size:13px;color:var(--muted)}

.avatar{width:72px;height:72px;border-radius:12px;background:linear-gradient(180deg,#fff,#fff9ff);display:flex;align-items:center;justify-content:center;font-weight:700;color:var(--accent);font-size:20px}

input[type="text"], input[type="number"], select, input[type="date"]{width:100%;padding:8px;border-radius:8px;border:1px solid #e8eefb;background:#fbfdff;margin-top:6px}

.btn{background:var(--accent);color:white;padding:8px 10px;border-radius:8px;border:none;cursor:pointer;font-weight:600}

.btn.secondary{background:var(--accent-2)}

.statgrid{display:grid;grid-template-columns:1fr 1fr;gap:8px;margin-top:12px}

.stat{background:linear-gradient(180deg,#fff,#fbfdff);padding:10px;border-radius:10px;text-align:center;border:1px solid rgba(11,22,38,0.03)}

.bar{height:14px;background:#f1f5f9;border-radius:12px;overflow:hidden}

.bar-inner{height:100%;background:linear-gradient(90deg,var(--accent),var(--accent-2));width:20%;transition:width 0.5s}

.searchbar{display:flex;gap:8px;align-items:center}

.searchbar input{padding:10px;border-radius:10px;border:1px solid #e6eefc;flex:1}

.meal-tabs{display:flex;gap:6px;margin-top:10px}

.meal-tabs button{flex:1;padding:8px;border-radius:8px;border:1px solid rgba(11,22,38,0.05);background:#fbfdff;cursor:pointer}

.meal-tabs button.active{background:linear-gradient(90deg,var(--accent),var(--accent-2));color:white;border:none}

.foods-list{max-height:260px;overflow:auto;margin-top:10px}

.food-item{display:flex;align-items:center;justify-content:space-between;padding:8px;border-radius:10px;border:1px solid rgba(11,22,38,0.03);margin-bottom:8px;background:#fff}

.chart{width:100%;height:140px;background:linear-gradient(180deg,#fff,#fbfdff);border-radius:10px;padding:8px}

.modal-bg{position:fixed;left:0;top:0;width:100%;height:100%;background:rgba(0,0,0,0.5);display:none;align-items:center;justify-content:center;z-index:9999}

.modal{background:white;border-radius:10px;padding:12px;max-width:720px;width:96%}

.exercise-line{display:flex;justify-content:space-between;align-items:center;padding:6px 0;border-bottom:1px dashed #eee}

.help-block{font-size:13px;color:var(--muted);margin-top:8px}

@media (max-width:960px){main{grid-template-columns:1fr;padding:12px}}

</style>

</head>

<body>

<header>

<div class="brand">

<div class="logo">MF</div>

<div>

<div style="font-weight:700">MiniFit + BellyFit</div>

<div class="small">Women's nutrition & belly-fat programmes</div>

</div>

</div>

<div style="display:flex;gap:8px;align-items:center">

<button id="btnExport" class="btn">Export</button>

<button id="btnImport" class="btn secondary">Import</button>

</div>

</header>

<main>

<!-- Left column -->

<section>

<div class="card">

<div style="display:flex;gap:12px;align-items:center">

<div class="avatar" id="avatarText">G</div>

<div>

<div id="userGreeting"><strong>Guest</strong></div>

<div class="small" id="userEmail">Sign in to personalise</div>

</div>

</div>

<div style="margin-top:12px">

<label class="small">Sign up / Login (local)</label>

<div style="display:flex;gap:8px;margin-top:8px">

<input id="inputName" type="text" placeholder="Name">

<input id="inputEmail" type="text" placeholder="Email">

</div>

<div style="display:flex;gap:8px;margin-top:8px">

<button id="btnSign" class="btn">Sign in</button>

<button id="btnSignOut" class="btn secondary">Sign out</button>

</div>

</div>

<div style="margin-top:12px">

<label class="small">Profile & goals (female-focused)</label>

<input id="age" type="number" placeholder="Age (years)">

<input id="height" type="number" placeholder="Height (cm)">

<input id="weight" type="number" placeholder="Weight (kg)">

<label class="small">Daily calorie goal</label>

<input id="goal" type="number" placeholder="e.g. 1500">

<div style="display:flex;gap:8px;margin-top:8px">

<button id="btnSaveProfile" class="btn">Save profile</button>

<button id="btnCalcGoal" class="btn secondary">Suggest goal</button>

</div>

<div class="small" style="margin-top:8px">Suggestion uses a female-specific BMR estimate and a safe ~500 kcal/day deficit as a starting point.</div>

</div>

<div class="statgrid" style="margin-top:12px">

<div class="stat">

<div class="small">Today</div>

<strong id="todayCals">0 kcal</strong>

<div class="small">Calories</div>

</div>

<div class="stat">

<div class="small">Remaining</div>

<strong id="remaining">0 kcal</strong>

<div class="small">To goal</div>

</div>

</div>

<div class="progress-wrap">

<label class="small">Progress to goal</label>

<div class="bar" style="margin-top:6px"><div class="bar-inner" id="barInner"></div></div>

</div>

</div>

<div class="card" style="margin-top:12px">

<div style="display:flex;justify-content:space-between;align-items:center">

<div style="font-weight:600">Daily log (history)</div>

<div class="small">Export a day or clear</div>

</div>

<div id="historyList" style="margin-top:10px"></div>

<div style="display:flex;gap:8px;margin-top:8px">

<button id="btnClearDay" class="btn">Clear today</button>

<button id="btnClearAll" class="btn secondary">Clear all</button>

</div>

</div>

<!-- BellyFit panel -->

<div class="card" style="margin-top:12px" id="bellyfitCard">

<div style="display:flex;justify-content:space-between;align-items:center">

<div style="font-weight:700">BellyFit — Female Focus</div>

<div class="small">Workouts, demos, and teacher CSV export</div>

</div>

<div style="margin-top:10px;display:flex;gap:8px;flex-wrap:wrap">

<button id="btnGeneratePlan" class="btn">Generate 4-Week Plan</button>

<button id="btnQuickHIIT" class="btn secondary">Quick 15-min HIIT</button>

<button id="btnCoreBlast" class="btn">Core 10-min</button>

<button id="btnExportCSV" class="btn" title="Export completed sessions as CSV">Export BellyFit CSV</button>

</div>

<div style="margin-top:12px">

<label class="small">Suggestion inputs</label>

<div style="display:flex;gap:8px;margin-top:6px;flex-wrap:wrap">

<input id="bf\_weight" type="number" placeholder="Weight (kg)">

<input id="bf\_height" type="number" placeholder="Height (cm)">

<input id="bf\_age" type="number" placeholder="Age">

<select id="bf\_sex" style="padding:8px;border-radius:8px"><option value="female" selected>Female</option></select>

<button id="btnSuggestCal" class="btn secondary">Suggest</button>

</div>

<div class="small" style="margin-top:8px">Suggested daily calories: <strong id="bf\_suggested">—</strong></div>

</div>

<div id="bfPlanWrap" style="margin-top:12px"></div>

<div style="margin-top:12px;background:#fff;padding:10px;border-radius:8px">

<div style="font-weight:600">Tips (research-based)</div>

<ul class="small" id="bfTips" style="margin-top:8px"></ul>

<div class="small help-block">Encourage pupils to consult a healthcare professional before beginning a calorie deficit or a new training plan.</div>

</div>

<div style="margin-top:12px;background:#fff;padding:10px;border-radius:8px">

<div style="font-weight:600">PWA & Packaging Help</div>

<div class="small help-block">To convert this web app to a Progressive Web App (PWA) or wrap it for native app stores, you will need two files on your web host: a <code>manifest.json</code> and a service worker <code>sw.js</code>. Download the sample files below and follow the instructions inside them. After hosting, the browser can prompt users to install the app to their device.</div>

<div style="display:flex;gap:8px;margin-top:8px;flex-wrap:wrap">

<button id="btnDownloadManifest" class="btn">Download manifest.json</button>

<button id="btnDownloadSW" class="btn secondary">Download sw.js</button>

<button id="btnShowPWAHelp" class="btn">PWA / Cordova Instructions</button>

</div>

</div>

</div>

</section>

<!-- Right column -->

<section>

<div class="card">

<div style="display:flex;justify-content:space-between;align-items:center">

<div>

<div style="font-size:16px;font-weight:600">Add Food & Log Meal</div>

<div class="small">Search the food catalogue or add a custom item</div>

</div>

<div style="display:flex;gap:8px;align-items:center">

<div class="small">Date</div>

<input id="entryDate" type="date">

</div>

</div>

<div style="margin-top:10px" class="searchbar">

<input id="foodSearch" placeholder="Search food (e.g. banana, rice, ackee)">

<button id="btnSearch" class="btn">Search</button>

</div>

<div style="display:flex;gap:8px;margin-top:10px;align-items:center;">

<div style="flex:1">

<div class="meal-tabs" id="mealTabs">

<button data-meal="breakfast" class="active">Breakfast</button>

<button data-meal="lunch">Lunch</button>

<button data-meal="dinner">Dinner</button>

<button data-meal="snack">Snack</button>

</div>

</div>

<div style="width:120px">

<label class="small">Portion (g)</label>

<input id="portion" type="number" value="100">

</div>

</div>

<div class="foods-list" id="foodsList"></div>

<div style="display:flex;gap:8px;margin-top:12px;align-items:center">

<button id="btnAddCustom" class="btn">Add custom food</button>

<div style="flex:1"></div>

<div class="small">Calories today: <strong id="calTodayMini">0</strong></div>

</div>

<div id="customForm" style="display:none;margin-top:12px">

<label class="small">Food name</label>

<input id="cfName" type="text" placeholder="e.g. Ackee & saltfish">

<label class="small">Calories per 100g</label>

<input id="cfCals" type="number" placeholder="kcal">

<label class="small">Protein per 100g (g)</label>

<input id="cfProtein" type="number" placeholder="g">

<label class="small">Carbs per 100g (g)</label>

<input id="cfCarbs" type="number" placeholder="g">

<label class="small">Fat per 100g (g)</label>

<input id="cfFat" type="number" placeholder="g">

<div style="display:flex;gap:8px;margin-top:8px">

<button id="btnSaveFood" class="btn">Save food</button>

<button id="btnCancelFood" class="btn secondary">Cancel</button>

</div>

</div>

</div>

<div class="card" style="margin-top:12px">

<div style="display:flex;justify-content:space-between;align-items:center">

<div style="font-weight:600">Daily summary</div>

<div class="small">Export or review earlier days</div>

</div>

<div style="display:flex;gap:12px;margin-top:12px;align-items:flex-start">

<div style="flex:1">

<div class="chart" id="chartWrap">

<canvas id="chartCanvas" width="700" height="140"></canvas>

</div>

</div>

<div style="width:260px">

<div class="small">Today details</div>

<div style="margin-top:10px">

<div class="small">Total calories</div>

<div style="font-size:22px;font-weight:700" id="todayTotal">0 kcal</div>

<div class="small" style="margin-top:6px">Macros (g)</div>

<div style="display:flex;gap:8px;margin-top:6px">

<div style="flex:1;background:#fff;padding:8px;border-radius:8px;text-align:center">

<div class="small">Protein</div>

<div id="mProtein" style="font-weight:700">0</div>

</div>

<div style="flex:1;background:#fff;padding:8px;border-radius:8px;text-align:center">

<div class="small">Carbs</div>

<div id="mCarbs" style="font-weight:700">0</div>

</div>

<div style="flex:1;background:#fff;padding:8px;border-radius:8px;text-align:center">

<div class="small">Fat</div>

<div id="mFat" style="font-weight:700">0</div>

</div>

</div>

</div>

<div style="margin-top:12px">

<button id="btnSaveDay" class="btn">Save day</button>

<button id="btnDownload" class="btn secondary">Download JSON</button>

</div>

</div>

</div>

</div>

</section>

</main>

<!-- Modal for exercise details and demo URL -->

<div class="modal-bg" id="modalBg">

<div class="modal" id="modal">

<div style="display:flex;justify-content:space-between;align-items:center">

<div id="modalTitle" style="font-weight:700">Exercise</div>

<button id="modalClose" class="btn secondary" style="padding:6px 8px">Close</button>

</div>

<div id="modalBody" style="margin-top:10px"></div>

</div>

</div>

<footer>MiniFit + BellyFit — Data stored locally. For classroom use, supervise pupils and encourage medical consultation for bespoke plans.</footer>

<script>

/\* Full combined app with:

- Food logging (MiniFit)

- Female BellyFit add-on

- Exercise demo URLs (embed)

- CSV export for BellyFit completed sessions

- PWA helper (downloadable manifest & sw)

Works in Mimo (localStorage).

\*/

/\* ---------- Utilities & storage ---------- \*/

const uid = ()=>Date.now().toString(36)+Math.random().toString(36).slice(2,8);

const $ = id=>document.getElementById(id);

const todayStr = (d=new Date())=>d.toISOString().slice(0,10);

const STORE\_KEY = "minifit\_bellyfit\_v2";

let state = { users:{}, foods:{}, currentUser:null };

function loadState(){

const raw = localStorage.getItem(STORE\_KEY);

if(raw){ try{ state = JSON.parse(raw); }catch(e){ console.warn("corrupt state"); } }

}

function saveState(){ localStorage.setItem(STORE\_KEY, JSON.stringify(state)); }

/\* ---------- Sample foods ---------- \*/

function ensureSampleFoods(){

if(Object.keys(state.foods).length>0) return;

const samples = [

{name:"Banana (100g)", cals:89, protein:1.1, carbs:22.8, fat:0.3},

{name:"White rice (cooked, 100g)", cals:130, protein:2.7, carbs:28.2, fat:0.3},

{name:"Ackee & saltfish (100g)", cals:150, protein:12, carbs:4, fat:9},

{name:"Boiled yam (100g)", cals:118, protein:1.5, carbs:27.9, fat:0.2},

{name:"Fried plantain (100g)", cals:122, protein:1.3, carbs:31.9, fat:0.4},

{name:"Grilled chicken breast (100g)", cals:165, protein:31, carbs:0, fat:3.6},

{name:"Whole milk (100g)", cals:60, protein:3.2, carbs:4.8, fat:3.3}

];

for(const f of samples){

const id = uid();

state.foods[id] = {id, name: f.name, cals100: f.cals, protein100: f.protein, carbs100: f.carbs, fat100: f.fat};

}

saveState();

}

/\* ---------- Auth & profile ---------- \*/

function signIn(){

const name = $("inputName").value.trim();

const email = $("inputEmail").value.trim().toLowerCase();

if(!name || !email){ alert("Enter name and email"); return; }

if(!state.users[email]) state.users[email] = {name, email, profile:{age:null,height:null,weight:null,goal:null},created:Date.now(),logs:{}, bellyFit:{plans:[],completed:[],demoLinks:{}}};

else state.users[email].name = name;

state.currentUser = email;

saveState(); refreshUI(); showToast("Signed in");

}

function signOut(){ state.currentUser = null; saveState(); refreshUI(); showToast("Signed out"); }

function saveProfile(){

if(!state.currentUser){ alert("Sign in first"); return; }

const p = state.users[state.currentUser].profile;

p.age = parseInt($("age").value) || p.age;

p.height = parseFloat($("height").value) || p.height;

p.weight = parseFloat($("weight").value) || p.weight;

p.goal = parseInt($("goal").value) || p.goal;

saveState(); refreshUI(); showToast("Profile saved");

}

/\* ---------- Calorie suggestion (female Mifflin) ---------- \*/

function suggestGoalForProfile(){

if(!state.currentUser){ alert("Sign in first"); return; }

const p = state.users[state.currentUser].profile;

if(!p.weight || !p.height || !p.age){ alert("Please enter weight, height and age for suggestion"); return; }

const bmr = Math.round(10\*p.weight + 6.25\*p.height - 5\*p.age - 161);

const maintenance = Math.round(bmr \* 1.4);

const suggested = Math.max(1200, maintenance - 500);

p.goal = suggested; saveState(); refreshUI(); showToast(`Suggested ${suggested} kcal/day`);

}

/\* ---------- Food catalogue ---------- \*/

function renderFoodList(filter=""){

const list = $("foodsList"); list.innerHTML = "";

const items = Object.values(state.foods).filter(f => f.name.toLowerCase().includes(filter.toLowerCase()));

items.slice(0,80).forEach(f=>{

const row = document.createElement("div"); row.className="food-item";

row.innerHTML = `<div><div style="font-weight:600">${f.name}</div><div class="small">${f.cals100} kcal /100g • P ${f.protein100}g • C ${f.carbs100}g • F ${f.fat100}g</div></div>

<div style="display:flex;flex-direction:column;gap:6px;align-items:flex-end">

<button data-id="${f.id}" class="btn addBtn" style="padding:6px 8px">Add</button>

<div class="small">per 100g</div>

</div>`;

list.appendChild(row);

});

document.querySelectorAll(".addBtn").forEach(b=>b.addEventListener("click", ()=>addFoodToLog(b.dataset.id)));

}

/\* Custom food \*/

function toggleCustomForm(show){ $("customForm").style.display = show ? "block" : "none"; }

function saveCustomFood(){

const name = $("cfName").value.trim(); const cals = parseFloat($("cfCals").value)||0;

const protein = parseFloat($("cfProtein").value)||0; const carbs = parseFloat($("cfCarbs").value)||0; const fat = parseFloat($("cfFat").value)||0;

if(!name || cals<=0){ alert("Enter name and calories"); return; }

const id = uid(); state.foods[id] = {id,name,cals100:cals,protein100:protein,carbs100:carbs,fat100:fat};

saveState(); toggleCustomForm(false); renderFoodList($("foodSearch").value); showToast("Food saved");

}

/\* ---------- Logging meals ---------- \*/

function currentMeal(){ return document.querySelector(".meal-tabs button.active").dataset.meal; }

function addFoodToLog(foodId){

if(!state.currentUser){ alert("Sign in first"); return; }

const portion = parseFloat($("portion").value) || 100; const meal = currentMeal();

const date = $("entryDate").value || todayStr(); const f = state.foods[foodId]; if(!f) return;

const entry = { id: uid(), foodId: f.id, name: f.name, portion, meal, cals: Math.round(f.cals100 \* portion / 100), protein: +(f.protein100 \* portion / 100).toFixed(1), carbs: +(f.carbs100 \* portion / 100).toFixed(1), fat: +(f.fat100 \* portion / 100).toFixed(1), created: Date.now() };

const user = state.users[state.currentUser]; if(!user.logs[date]) user.logs[date]=[]; user.logs[date].push(entry); saveState(); refreshUI(); showToast("Added to log");

}

/\* ---------- Day summary & history ---------- \*/

function getDayData(dateStr=todayStr()){

if(!state.currentUser) return {cals:0,protein:0,carbs:0,fat:0,entries:[]};

const user = state.users[state.currentUser]; const list = user.logs[dateStr] || [];

const sum = list.reduce((acc,e)=>{ acc.cals+=e.cals; acc.protein+=e.protein; acc.carbs+=e.carbs; acc.fat+=e.fat; return acc; }, {cals:0,protein:0,carbs:0,fat:0});

return {cals: sum.cals, protein: Math.round(sum.protein\*10)/10, carbs: Math.round(sum.carbs\*10)/10, fat: Math.round(sum.fat\*10)/10, entries:list};

}

function renderHistoryList(){

const wrap = $("historyList"); wrap.innerHTML = ""; if(!state.currentUser) return;

const user = state.users[state.currentUser]; const dates = Object.keys(user.logs).sort((a,b)=>b.localeCompare(a));

dates.forEach(d=>{

const day = user.logs[d]; const cals = day.reduce((s,e)=>s+e.cals,0);

const div = document.createElement("div"); div.className="small"; div.style.marginBottom="8px";

div.innerHTML = `<div style="display:flex;justify-content:space-between;align-items:center"><div><strong>${d}</strong><div class="small">${day.length} item(s)</div></div>

<div style="text-align:right"><div style="font-weight:700">${cals} kcal</div><div style="display:flex;gap:6px;margin-top:6px">

<button class="btn viewDay" data-day="${d}" style="padding:6px 8px">View</button>

<button class="btn secondary delDay" data-day="${d}" style="padding:6px 8px">Delete</button></div></div></div>`;

wrap.appendChild(div);

});

document.querySelectorAll(".viewDay").forEach(b=>b.addEventListener("click", e=>showDayModal(e.target.dataset.day)));

document.querySelectorAll(".delDay").forEach(b=>b.addEventListener("click", e=>{ if(!confirm("Delete day "+e.target.dataset.day+"?")) return; delete state.users[state.currentUser].logs[e.target.dataset.day]; saveState(); refreshUI(); }));

}

function showDayModal(day){

const data = getDayData(day); let txt=`Entries for ${day}:\n\n`; data.entries.forEach(en=>txt+=`${en.meal.toUpperCase()} • ${en.name} • ${en.portion}g • ${en.cals} kcal\n`);

alert(txt);

}

/\* ---------- Chart ---------- \*/

function renderChart(){

const canvas = $("chartCanvas"); const ctx = canvas.getContext("2d"); ctx.clearRect(0,0,canvas.width,canvas.height); if(!state.currentUser) return;

const user = state.users[state.currentUser]; const dates=[]; for(let i=6;i>=0;i--){ const dt=new Date(); dt.setDate(dt.getDate()-i); dates.push(dt.toISOString().slice(0,10)); }

const vals = dates.map(d=>{ const arr = user.logs[d]||[]; return arr.reduce((s,e)=>s+e.cals,0); });

const max = Math.max(200,...vals);

ctx.strokeStyle="#f1f5f9"; ctx.lineWidth=1; for(let i=0;i<7;i++){ const x=10+i\*(canvas.width-20)/6; ctx.beginPath(); ctx.moveTo(x,10); ctx.lineTo(x,canvas.height-10); ctx.stroke(); }

ctx.beginPath(); ctx.moveTo(10, canvas.height-10 - (vals[0]/max)\*(canvas.height-30));

for(let i=1;i<7;i++){ const x=10+i\*(canvas.width-20)/6; const y=canvas.height-10 - (vals[i]/max)\*(canvas.height-30); ctx.lineTo(x,y); }

ctx.strokeStyle="#d946ef"; ctx.lineWidth=3; ctx.stroke();

ctx.lineTo(canvas.width-10, canvas.height-10); ctx.lineTo(10, canvas.height-10); ctx.fillStyle="rgba(217,70,239,0.12)"; ctx.fill();

ctx.fillStyle="#0b1730"; ctx.font="12px Inter"; ctx.fillText("Last 7 days (kcal)",10,14);

}

/\* ---------- Toast ---------- \*/

function showToast(msg){

const el=document.createElement("div"); el.textContent=msg; Object.assign(el.style,{position:"fixed",right:"16px",bottom:"16px",background:"#0b1728",color:"#fff",padding:"10px 12px",borderRadius:"8px",zIndex:9999}); document.body.appendChild(el); setTimeout(()=>el.remove(),1500);

}

/\* ---------- Export/Import ---------- \*/

function exportData(){ const blob=new Blob([JSON.stringify(state,null,2)],{type:"application/json"}); const url=URL.createObjectURL(blob); const a=document.createElement("a"); a.href=url; a.download="minifit\_bellyfit\_export.json"; a.click(); URL.revokeObjectURL(url); }

function importDataFile(){ const inp=document.createElement("input"); inp.type="file"; inp.accept="application/json"; inp.onchange=()=>{ const f=inp.files[0]; if(!f) return; const reader=new FileReader(); reader.onload=()=>{ try{ const obj=JSON.parse(reader.result); if(confirm("Replace local data with imported data? This will overwrite current local data.")){ state=obj; saveState(); refreshUI(); } }catch(e){ alert("Invalid file."); } }; reader.readAsText(f); }; inp.click(); }

function downloadDayJSON(){ if(!state.currentUser){ alert("Sign in first."); return; } const date=$("entryDate").value||todayStr(); const entries=state.users[state.currentUser].logs[date]||[]; const blob=new Blob([JSON.stringify(entries,null,2)],{type:'application/json'}); const url=URL.createObjectURL(blob); const a=document.createElement("a"); a.href=url; a.download=`minifit\_${date}.json`; a.click(); URL.revokeObjectURL(url); }

/\* ---------- Clear ---------- \*/

function clearToday(){ if(!state.currentUser){ alert("Sign in first."); return; } const date=$("entryDate").value||todayStr(); state.users[state.currentUser].logs[date]=[]; saveState(); refreshUI(); showToast("Today cleared"); }

function clearAll(){ if(!confirm("Clear all local data (cannot undo)?")) return; state={users:{},foods:{},currentUser:null}; ensureSampleFoods(); saveState(); refreshUI(); }

/\* ---------- BellyFit add-on ---------- \*/

const evidenceTips = [

"Combine resistance training with a modest calorie deficit to preserve muscle while reducing fat.",

"Aim for ≥150 minutes of moderate aerobic activity per week — or include HIIT sessions.",

"Prioritise sleep and stress management — poor sleep and stress can hinder abdominal fat loss.",

"HIIT is time-efficient but pair it with strength sessions for best results.",

"A practical deficit of ~500 kcal/day supports gradual fat loss; avoid extreme low-calorie diets."

];

const exerciseLibrary = {

warmup:[{name:"Brisk march (in place)",time:"2 min"},{name:"Arm circles & leg swings",time:"2 min"}],

resistance:[

{name:"Glute bridge",reps:"3×12",tip:"Squeeze glutes at top"},

{name:"Goblet squat (bodyweight if needed)",reps:"3×10",tip:"Knees track over toes"},

{name:"Dead bug (core)",reps:"3×12",tip:"Keep lower back pressed"},

{name:"Bent-over row (dumbbell/band)",reps:"3×10",tip:"Squeeze shoulder blades"},

{name:"Side plank",reps:"3×30s/side",tip:"Keep hips high"}

],

hiit:[

{name:"Mountain climbers",time:"30s"},

{name:"Jump squats or pulses",time:"30s"},

{name:"High knees",time:"30s"},

{name:"Rest",time:"30s"}

],

coreBlast:[

{name:"Plank hip dips",time:"45s"},

{name:"Bicycle crunches",time:"45s"},

{name:"Russian twist (light weight)",time:"45s"}

]

};

function pick(arr,n){ const copy=arr.slice(); const out=[]; for(let i=0;i<n;i++){ if(copy.length===0) break; const idx=Math.floor(Math.random()\*copy.length); out.push(copy.splice(idx,1)[0]); } return out; }

function curUserObj(){ if(!state.currentUser) return null; const user = state.users[state.currentUser]; if(!user.bellyFit) user.bellyFit={plans:[],completed:[],demoLinks:{}}; return user; }

/\* Generate 4-week plan \*/

function generate4WeekPlan(){

const user = curUserObj(); if(!user){ alert("Sign in to generate a plan."); return; }

const plan={id:uid(),created:Date.now(),weeks:[]};

for(let w=1;w<=4;w++){

const week={week:w,days:[]};

for(let d=1;d<=7;d++){

let day={day:d,type:"rest",items:[]};

if([1,3,5].includes(d)){ day.type="strength"; day.items=[...exerciseLibrary.warmup,...pick(exerciseLibrary.resistance,3)]; }

if([2,6].includes(d)){ day.type="hiit"; day.items=[...exerciseLibrary.warmup,...exerciseLibrary.hiit,{name:"Cool-down walk",time:"5 min"}]; }

if(d===4){ day.type="active"; day.items=[{name:"Brisk walk or yoga",time:"30-45 min"}]; }

if(d===7){ day.type="rest"; day.items=[{name:"Gentle stretch",time:"10-20 min"}]; }

week.days.push(day);

}

plan.weeks.push(week);

}

user.bellyFit.plans.push(plan); saveState(); showToast("4-week plan created"); renderPlansUI(); refreshUI();

}

/\* Quick sessions \*/

function getQuickHIIT(){ return {id:uid(),title:"15-minute Quick HIIT",items:[...exerciseLibrary.warmup,{name:"Circuit x8: Mountain climbers 30s / Jump squats 30s / High knees 30s / Rest 30s"},{name:"Cool-down & stretch",time:"4 min"}],created:Date.now()}; }

function getCoreBlast(){ return {id:uid(),title:"10-minute Core Blast",items:[...exerciseLibrary.coreBlast,{name:"Plank hold",time:"60s"}],created:Date.now()}; }

/\* Completed session saver \*/

function addCompletedSession(session){

const user = curUserObj(); if(!user){ alert("Sign in to save session"); return; }

const s = Object.assign({},session); s.created=Date.now(); if(!user.bellyFit.completed) user.bellyFit.completed=[];

user.bellyFit.completed.push(s);

const entryDate = ($("entryDate") && $("entryDate").value) || todayStr();

const workoutEntry = { id:uid(), foodId:null, name: s.title || s.name || "Session", portion:0, meal:"workout", cals: s.estimatedCalories || 150, protein:0, carbs:0, fat:0, created:Date.now() };

if(!user.logs) user.logs={}; if(!user.logs[entryDate]) user.logs[entryDate]=[]; user.logs[entryDate].push(workoutEntry);

saveState(); showToast("Session saved to Completed and today's log"); renderPlansUI(); refreshUI();

}

/\* Suggest calories function \*/

function suggestCaloriesForUser(weightKg,heightCm,age){

const bmr = Math.round(10\*weightKg + 6.25\*heightCm - 5\*age - 161);

const maintenance = Math.round(bmr\*1.4);

const suggested = Math.max(1200, maintenance - 500);

return {bmr,maintenance,suggested};

}

/\* Render plans UI with ability to view day details and add demo URLs \*/

function renderPlansUI(){

const wrap = $("bfPlanWrap"); if(!wrap) return; wrap.innerHTML="";

const user = curUserObj(); if(!user){ wrap.innerHTML="<div class='small'>Sign in to save and view plans.</div>"; return; }

const plans = user.bellyFit.plans || [];

if(plans.length===0) wrap.innerHTML="<div class='small'>No saved plans yet — generate a personalised 4-week plan.</div>";

else plans.slice(-3).reverse().forEach(p=>{

const card = document.createElement("div"); card.className="card"; card.style.marginTop="8px";

card.innerHTML = `<div style="display:flex;justify-content:space-between;align-items:center"><div style="font-weight:600">Plan created ${new Date(p.created).toLocaleDateString()}</div></div>`;

const w1 = p.weeks[0];

const list = document.createElement("div"); list.style.marginTop="8px"; list.className="small";

w1.days.forEach(d=>{

const line = document.createElement("div"); line.style.display="flex"; line.style.justifyContent="space-between"; line.style.alignItems="center"; line.style.marginBottom="6px";

line.innerHTML = `<div>Day ${d.day}: <strong>${d.type}</strong> • ${d.items.length} items</div><div style="display:flex;gap:6px"><button class="btn viewDayDetails" data-weekid="${p.id}" data-day="${d.day}" style="padding:6px 8px">Show details</button></div>`;

list.appendChild(line);

});

card.appendChild(list);

wrap.appendChild(card);

});

// Quick session controls

const quickWrap = document.createElement("div"); quickWrap.style.marginTop="8px"; quickWrap.innerHTML = `<div class="small" style="margin-bottom:6px">Quick sessions:</div>`;

const hiitBtn = document.createElement("button"); hiitBtn.className="btn"; hiitBtn.textContent="Add Quick HIIT to Completed"; hiitBtn.style.marginRight="8px"; hiitBtn.addEventListener("click", ()=>addCompletedSession(getQuickHIIT()));

const coreBtn = document.createElement("button"); coreBtn.className="btn secondary"; coreBtn.textContent="Add Core Blast to Completed"; coreBtn.addEventListener("click", ()=>addCompletedSession(getCoreBlast()));

quickWrap.appendChild(hiitBtn); quickWrap.appendChild(coreBtn); wrap.appendChild(quickWrap);

// Completed sessions list

const comp = user.bellyFit.completed || [];

const compCard = document.createElement("div"); compCard.className="card"; compCard.style.marginTop="8px"; compCard.innerHTML = `<div style="font-weight:600">Completed sessions (latest 6)</div>`;

comp.slice(-6).reverse().forEach(s=>{ const r=document.createElement("div"); r.className="small"; r.style.marginTop="6px"; r.textContent = `${new Date(s.created).toLocaleString()} • ${s.title || 'Session'} • ${s.items? s.items.length+' items':''}`; compCard.appendChild(r); });

wrap.appendChild(compCard);

// Attach listeners for view details

document.querySelectorAll(".viewDayDetails").forEach(btn=>{

btn.addEventListener("click", e=>{

const planId = e.target.dataset.weekid; const dayNum = parseInt(e.target.dataset.day,10);

const plan = (user.bellyFit.plans||[]).find(p=>p.id===planId);

if(!plan){ alert("Plan not found"); return; }

const day = plan.weeks[0].days.find(d=>d.day===dayNum);

showDayModalWithDetails(planId, day);

});

});

}

/\* Show day modal with details and demo link options \*/

function showDayModalWithDetails(planId, day){

const modalBg = $("modalBg"), modal = $("modal"), title = $("modalTitle"), body = $("modalBody");

title.textContent = `Day ${day.day} — ${day.type}`;

const user = curUserObj();

body.innerHTML = "";

day.items.forEach((it, idx)=>{

const el = document.createElement("div"); el.className="exercise-line";

el.innerHTML = `<div><strong>${it.name}</strong> ${it.reps?('• '+it.reps):''} ${it.time?('• '+it.time):''}<div class="small">${it.tip?it.tip:''}</div></div>

<div style="display:flex;gap:6px;align-items:center">

<button class="btn demoBtn" data-name="${it.name}" style="padding:6px 8px">Show demo</button>

<button class="btn secondary editDemo" data-name="${it.name}" style="padding:6px 8px">Add / Edit demo URL</button>

</div>`;

body.appendChild(el);

});

// Demo & edit listeners

body.querySelectorAll(".demoBtn").forEach(b=>{

b.addEventListener("click", e=>{

const name = e.target.dataset.name;

const links = user && user.bellyFit && user.bellyFit.demoLinks || {};

const url = links[name];

if(url){

// show embedded iframe

body.querySelectorAll(".embeddedFrame").forEach(f=>f.remove());

const iframeWrap = document.createElement("div"); iframeWrap.className="embeddedFrame"; iframeWrap.style.marginTop="10px";

// ensure embed URL: convert watch?v= to embed

const embedUrl = toEmbedUrl(url);

iframeWrap.innerHTML = `<div class="small" style="margin-bottom:6px">Demo for ${name}</div><iframe src="${embedUrl}" width="100%" height="360" frameborder="0" allowfullscreen></iframe>`;

body.appendChild(iframeWrap);

} else {

// open YouTube search in new tab

const query = encodeURIComponent(name + " exercise tutorial");

window.open(`https://www.youtube.com/results?search\_query=${query}`, "\_blank");

}

});

});

body.querySelectorAll(".editDemo").forEach(b=>{

b.addEventListener("click", e=>{

const name = e.target.dataset.name;

const links = user && user.bellyFit && user.bellyFit.demoLinks || {};

const existing = links[name] || "";

const input = prompt(`Paste YouTube URL (watch?v=...) for "${name}"\nLeave empty to remove`, existing);

if(input===null) return;

if(!user.bellyFit.demoLinks) user.bellyFit.demoLinks={};

if(input.trim()===""){ delete user.bellyFit.demoLinks[name]; showToast("Demo removed"); }

else { user.bellyFit.demoLinks[name] = input.trim(); showToast("Demo saved"); }

saveState();

// if currently showing iframe for this name, refresh

// not necessary; user can click Show demo again

});

});

modalBg.style.display = "flex";

}

/\* Convert YouTube watch URL to embed URL if possible \*/

function toEmbedUrl(url){

try{

if(!url) return "";

// If already embed, return

if(url.includes("youtube.com/embed/") || url.includes("youtube-nocookie.com")) return url;

// Extract v=VIDEO\_ID

const m = url.match(/[?&]v=([a-zA-Z0-9\_-]{8,})/);

if(m && m[1]) return `https://www.youtube.com/embed/${m[1]}`;

// If youtu.be short link

const m2 = url.match(/youtu\.be\/([a-zA-Z0-9\_-]{8,})/);

if(m2 && m2[1]) return `https://www.youtube.com/embed/${m2[1]}`;

// Fallback: open URL as-is in iframe (may be blocked)

return url;

}catch(e){ return url; }

}

/\* CSV export for BellyFit completed sessions \*/

function exportBellyFitCSV(){

if(!state.currentUser){ alert("Sign in first"); return; }

const user = state.users[state.currentUser];

const rows = [["timestamp","title","item\_count","estimated\_calories","notes"]];

const done = user.bellyFit && user.bellyFit.completed ? user.bellyFit.completed : [];

done.forEach(s=>{

const ts = new Date(s.created).toISOString();

const title = s.title ? s.title.replace(/,/g," ") : (s.name||"Session").replace(/,/g," ");

const count = s.items ? s.items.length : "";

const cals = s.estimatedCalories || "";

rows.push([ts,title,count,cals,""]);

});

const csv = rows.map(r=>r.map(v=>`"${String(v||"")}"`).join(",")).join("\n");

const blob = new Blob([csv],{type:"text/csv"}); const url = URL.createObjectURL(blob);

const a = document.createElement("a"); a.href=url; a.download = `bellyfit\_completed\_${state.currentUser}.csv`; a.click(); URL.revokeObjectURL(url);

showToast("BellyFit CSV downloaded");

}

/\* PWA helper: generate manifest and sample sw.js for download \*/

function downloadManifest(){

const manifest = {

short\_name: "MiniFit",

name: "MiniFit + BellyFit",

icons: [

{src:"icons/icon-192.png",sizes:"192x192",type:"image/png"},

{src:"icons/icon-512.png",sizes:"512x512",type:"image/png"}

],

start\_url: ".",

display: "standalone",

background\_color: "#fffaf6",

theme\_color: "#d946ef"

};

const blob = new Blob([JSON.stringify(manifest,null,2)],{type:"application/json"}); const url = URL.createObjectURL(blob);

const a = document.createElement("a"); a.href=url; a.download="manifest.json"; a.click(); URL.revokeObjectURL(url); showToast("manifest.json downloaded");

}

function downloadServiceWorker(){

const swCode = `// Simple service worker (cache then network). Place as /sw.js on your server.

const CACHE\_NAME = 'minifit-bellyfit-v1';

const STATIC\_ASSETS = [

'/', '/index.html'

// add any static assets you host (icons, css)

];

self.addEventListener('install', event => {

event.waitUntil(caches.open(CACHE\_NAME).then(cache => cache.addAll(STATIC\_ASSETS)));

self.skipWaiting();

});

self.addEventListener('activate', event => {

event.waitUntil(self.clients.claim());

});

self.addEventListener('fetch', event => {

event.respondWith(

caches.match(event.request).then(cached => cached || fetch(event.request))

);

});`;

const blob = new Blob([swCode],{type:"text/javascript"}); const url = URL.createObjectURL(blob);

const a = document.createElement("a"); a.href=url; a.download="sw.js"; a.click(); URL.revokeObjectURL(url); showToast("sw.js downloaded");

}

/\* PWA & Cordova instructions modal \*/

function showPWAHelp(){

const txt = `PWA (quick):\n1) Host your HTML, manifest.json and sw.js at the same origin.\n2) Add <link rel="manifest" href="/manifest.json"> in head and register /sw.js in JS.\n3) Use recommended icons (192x192, 512x512). After hosting, visiting the site will prompt install on supported browsers.\n\nWrap with Cordova/Capacitor (summary):\n- Cordova: create project, copy web build into www/, then run: cordova platform add android; cordova build android.\n- Capacitor: npx cap init, npx cap add android/ios, copy web build into /www, then npx cap open android.\n\nFor classroom: download manifest.json and sw.js from this app, upload them beside your hosted index.html, then add the script to register the SW.`;

alert(txt);

}

/\* ---------- Render Belly tips ---------- \*/

(function renderBellyTips(){

const el = $("bfTips"); evidenceTips.forEach(t=>{ const li=document.createElement("li"); li.textContent=t; li.style.marginBottom="6px"; el.appendChild(li); });

})();

/\* ---------- UI refresh ---------- \*/

function refreshUI(){

if(state.currentUser){

const user=state.users[state.currentUser];

$("userGreeting").innerHTML = `<strong>${user.name}</strong>`;

$("userEmail").textContent = user.email;

$("avatarText").textContent = user.name ? user.name[0].toUpperCase() : "U";

$("inputName").value = user.name; $("inputEmail").value = user.email;

$("age").value = user.profile.age||""; $("height").value = user.profile.height||""; $("weight").value = user.profile.weight||""; $("goal").value = user.profile.goal||"";

} else {

$("userGreeting").innerHTML = `<strong>Guest</strong>`; $("userEmail").textContent = "Sign in to personalise"; $("avatarText").textContent = "G";

$("inputName").value=""; $("inputEmail").value=""; $("age").value=""; $("height").value=""; $("weight").value=""; $("goal").value="";

}

renderFoodList($("foodSearch").value||"");

const d = getDayData($("entryDate").value||todayStr());

$("todayCals").textContent = `${d.cals} kcal`;

$("calTodayMini") && ($("calTodayMini").textContent = `${d.cals} kcal`);

$("todayTotal").textContent = `${d.cals} kcal`;

$("mProtein").textContent = d.protein; $("mCarbs").textContent = d.carbs; $("mFat").textContent = d.fat;

const goal = state.currentUser ? (state.users[state.currentUser].profile.goal || 1500) : 1500;

const remaining = Math.max(0, goal - d.cals);

$("remaining").textContent = `${remaining} kcal`;

const pct = Math.min(100, Math.round(d.cals / goal \* 100));

$("barInner").style.width = pct + "%";

renderHistoryList(); renderChart(); renderPlansUI(); saveState();

}

/\* ---------- Event wiring ---------- \*/

$("btnSign").addEventListener("click", signIn);

$("btnSignOut").addEventListener("click", signOut);

$("btnSaveProfile").addEventListener("click", saveProfile);

$("btnCalcGoal").addEventListener("click", suggestGoalForProfile);

$("btnSearch").addEventListener("click", ()=>renderFoodList($("foodSearch").value));

$("foodSearch").addEventListener("keydown", e=>{ if(e.key==="Enter") renderFoodList($("foodSearch").value); });

$("btnAddCustom").addEventListener("click", ()=>toggleCustomForm(true));

$("btnCancelFood").addEventListener("click", ()=>toggleCustomForm(false));

$("btnSaveFood").addEventListener("click", saveCustomFood);

document.querySelectorAll(".meal-tabs button").forEach(b=>b.addEventListener("click", e=>{ document.querySelectorAll(".meal-tabs button").forEach(x=>x.classList.remove("active")); e.target.classList.add("active"); }));

$("btnSaveDay").addEventListener("click", ()=>{ saveState(); showToast("Day saved"); });

$("btnDownload").addEventListener("click", downloadDayJSON);

$("btnClearDay").addEventListener("click", clearToday);

$("btnClearAll").addEventListener("click", clearAll);

$("btnExport").addEventListener("click", exportData);

$("btnImport").addEventListener("click", importDataFile);

$("btnSearch").addEventListener("click", ()=>renderFoodList($("foodSearch").value));

/\* BellyFit wiring \*/

$("btnGeneratePlan").addEventListener("click", generate4WeekPlan);

$("btnQuickHIIT").addEventListener("click", ()=>addCompletedSession(getQuickHIIT()));

$("btnCoreBlast").addEventListener("click", ()=>addCompletedSession(getCoreBlast()));

$("btnExportCSV").addEventListener("click", exportBellyFitCSV);

$("btnSuggestCal").addEventListener("click", ()=>{ const w=parseFloat($("bf\_weight").value)||0; const h=parseFloat($("bf\_height").value)||0; const a=parseInt($("bf\_age").value)||30; if(w<=30||h<=100){ alert("Enter realistic weight/height"); return; } const r=suggestCaloriesForUser(w,h,a); $("bf\_suggested").textContent = `${r.suggested} kcal/day (maintenance ≈ ${r.maintenance} kcal)`; });

/\* PWA helper \*/

$("btnDownloadManifest").addEventListener("click", downloadManifest);

$("btnDownloadSW").addEventListener("click", downloadServiceWorker);

$("btnShowPWAHelp").addEventListener("click", showPWAHelp);

/\* Modal behaviour \*/

$("modalClose").addEventListener("click", ()=>{$("modalBg").style.display="none"; $("modalBody").innerHTML="";});

$("modalBg").addEventListener("click", e=>{ if(e.target.id==="modalBg"){ $("modalBg").style.display="none"; $("modalBody").innerHTML=""; }});

/\* Entry date default \*/

$("entryDate").value = todayStr();

/\* ---------- Init ---------- \*/

loadState(); ensureSampleFoods(); refreshUI();

/\* Note about service worker registration:

To register sw.js on your hosting, add:

if('serviceWorker' in navigator){ navigator.serviceWorker.register('/sw.js').then(()=>console.log('SW registered')); }

This only works when index.html, manifest.json and sw.js are served from a proper web server (https recommended).

\*/

</script>

</body>

</html>