<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8" />

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

<title>Task Tracker — Digital Daktar Bari (Phase 01: Feb–Jan 2026)</title>

<style>

:root {

--bg: #0b1220;

--card: #121a2b;

--muted: #93a0b8;

--text: #e6ecff;

--primary: #4f7cff;

--accent: #7ef0d0;

--danger: #ff6b6b;

--warn: #ffb454;

--success: #36d399;

--ring: rgba(79,124,255,.25);

}

\* { box-sizing: border-box; }

html, body { height: 100%; }

body { margin:0; font-family: ui-sans-serif,system-ui,-apple-system,Segoe UI,Roboto,Helvetica,Arial; background: var(--bg); color: var(--text); }

a { color: var(--accent); }

.container { max-width: 1200px; margin: 0 auto; padding: 24px; }

.topbar { display:flex; align-items:center; gap:16px; justify-content: space-between; }

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

.brand h1 { font-size: 18px; margin:0; font-weight: 600; }

.brand small { display:block; color: var(--muted); }

.row { display:flex; gap:16px; flex-wrap: wrap; }

.card { background: var(--card); border: 1px solid rgba(255,255,255,0.05); border-radius: 14px; padding: 16px; box-shadow: 0 10px 30px rgba(0,0,0,0.25); }

.card h2 { margin: 0 0 12px; font-size: 16px; font-weight: 600; }

.btn { appearance:none; border:1px solid rgba(255,255,255,0.12); background: #111827; color: var(--text); padding:10px 14px; border-radius: 12px; cursor:pointer; font-weight: 600; }

.btn:hover { border-color: rgba(255,255,255,0.2); }

.btn.primary { background: var(--primary); border-color: transparent; }

.btn.primary:hover { filter: brightness(1.05); }

.btn.ghost { background: transparent; }

.pill { padding:4px 10px; border-radius:999px; font-size: 12px; font-weight:700; white-space:nowrap; }

.pill.overdue { background: rgba(255,107,107,0.15); color: var(--danger); }

.pill.duetoday { background: rgba(255,180,84,0.15); color: var(--warn); }

.pill.duesoon { background: rgba(255,180,84,0.15); color: var(--warn); }

.pill.completed { background: rgba(54,211,153,0.15); color: var(--success); }

.pill.open { background: rgba(79,124,255,0.15); color: var(--primary); }

.table-wrap { overflow: auto; }

table { width:100%; border-collapse: collapse; }

th, td { padding:10px 8px; border-bottom: 1px solid rgba(255,255,255,0.06); vertical-align: top; }

th { text-align:left; color: var(--muted); font-weight:600; position: sticky; top:0; background: var(--card); }

tbody tr:hover { background: rgba(255,255,255,0.03); }

input[type="text"], input[type="date"], textarea, input[type="email"] {

width: 100%; background: #0e1626; color: var(--text); border:1px solid rgba(255,255,255,0.1); border-radius: 10px; padding:8px 10px;

outline: none; box-shadow: none;

}

input:focus, textarea:focus { border-color: var(--primary); box-shadow: 0 0 0 4px var(--ring); }

textarea { resize: vertical; min-height: 38px; }

.grid { display:grid; gap: 10px; grid-template-columns: repeat(12, 1fr); }

.col-3 { grid-column: span 3; }

.col-4 { grid-column: span 4; }

.col-5 { grid-column: span 5; }

.col-6 { grid-column: span 6; }

.col-8 { grid-column: span 8; }

.col-12 { grid-column: span 12; }

.bell { position: relative; }

.badge { position:absolute; right:-2px; top:-2px; background: var(--danger); color:#fff; border-radius:999px; padding:2px 6px; font-size:11px; font-weight:700; }

.dropdown { position:absolute; right:0; top: 42px; width: 380px; max-height: 420px; overflow:auto; background: var(--card); border:1px solid rgba(255,255,255,0.08); border-radius: 12px; display:none; }

.dropdown.open { display:block; }

.notif { padding:12px; display:flex; gap:10px; border-bottom:1px solid rgba(255,255,255,0.06); }

.notif small { color: var(--muted); }

canvas { width:100%; height: 260px; background: #0e1626; border-radius: 12px; border:1px solid rgba(255,255,255,0.06); }

.muted { color: var(--muted); }

.sep { height:1px; background: rgba(255,255,255,0.08); margin:14px 0; }

@media (max-width: 900px){ .col-6 { grid-column: span 12; } .col-8 { grid-column: span 12; } .col-4 { grid-column: span 12; } .col-3 { grid-column: span 6; } }

</style>

</head>

<body>

<div class="container">

<div class="topbar">

<div class="brand">

<svg width="28" height="28" viewBox="0 0 24 24" fill="none"><path d="M12 2l7 4v6c0 5-3.5 9-7 10-3.5-1-7-5-7-10V6l7-4z" stroke="var(--accent)" stroke-width="1.5"/></svg>

<div>

<h1>Task Tracker</h1>

<small>Digital Daktar Bari — Phase 01 (Feb–Jan 2026)</small>

</div>

</div>

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

<button class="btn ghost" id="btn-export">Export JSON</button>

<label class="btn ghost" for="import-file">Import JSON<input id="import-file" type="file" accept="application/json" style="display:none"></label>

<button class="btn" id="btn-admin">Admin: <span id="admin-mode">OFF</span></button>

<div class="bell">

<button class="btn" id="btn-bell">🔔 Notifications</button>

<span class="badge" id="badge" style="display:none">0</span>

<div class="dropdown" id="dropdown"></div>

</div>

</div>

</div>

<div class="sep"></div>

<div class="row">

<div class="card col-8">

<h2>Tasks</h2>

<div class="grid">

<div class="col-4"><input type="text" id="new-task" placeholder="Task \*"/></div>

<div class="col-4"><input type="text" id="new-details" placeholder="Details"/></div>

<div class="col-3"><input type="date" id="new-deadline" title="Deadline \*"/></div>

<div class="col-3"><input type="email" id="new-email" placeholder="Email (completion notices)"/></div>

<div class="col-3"><input type="email" id="new-assigned" placeholder="Assigned (assignee email)"/></div>

<div class="col-3"><input type="email" id="new-client" placeholder="Client (weekly only)"/></div>

<div class="col-6"><input type="text" id="new-remarks" placeholder="Remarks"/></div>

<div class="col-12" style="display:flex; gap:8px; justify-content:flex-end;">

<button class="btn primary" id="btn-add">Add Task</button>

<button class="btn" id="btn-clear">Clear Form</button>

</div>

</div>

<div class="sep"></div>

<div class="table-wrap">

<table id="tbl">

<thead>

<tr>

<th>Task</th>

<th>Details</th>

<th>Deadline</th>

<th>Status</th>

<th>Completion</th>

<th>Email</th>

<th>Assigned</th>

<th>Client</th>

<th>Remarks</th>

<th></th>

</tr>

</thead>

<tbody></tbody>

</table>

</div>

</div>

<div class="card col-4">

<h2>Progress (last 7 days)</h2>

<canvas id="chart" width="600" height="260"></canvas>

<p class="muted" id="metric"></p>

</div>

</div>

</div>

<script>

// ====== CONFIG ======

const CAMPAIGN\_LINE = 'this is an update about Digital Daktar Bari campaign, Phase 01 (Feb-Jan 2026)';

const BD\_OFFSET\_HOURS = 6; // Asia/Dhaka (UTC+6, no DST)

// ====== STATE ======

let ADMIN\_MODE = false; // toggles deadline/email field editability

let tasks = loadTasks();

let notifications = loadNotifs();

// ====== UTIL: BD time helpers ======

function nowBD() {

const now = new Date();

const utcMs = now.getTime() + now.getTimezoneOffset()\*60000;

return new Date(utcMs + BD\_OFFSET\_HOURS\*3600000);

}

function startOfDayBD(d) {

const bd = new Date(d.getTime());

bd.setHours(0,0,0,0);

return bd;

}

function fmtDate(d) {

if (!d) return '';

const y = d.getFullYear();

const m = String(d.getMonth()+1).padStart(2,'0');

const day = String(d.getDate()).padStart(2,'0');

return `${y}-${m}-${day}`;

}

function parseISO(s){ if(!s) return null; const d = new Date(s); return isNaN(d) ? null : d; }

function isFriOrSatBD(d){ const day = (d.getDay()+6)%7; // 0=Mon..6=Sun mapping; want Fri=4 Sat=5

const dow = d.getDay(); // 0=Sun..6=Sat JS

return (dow===5 || dow===6); // Fri=5 Sat=6

}

function daysDiffBD(a,b) { // a-b in whole days based on midnight boundaries

const A = startOfDayBD(a).getTime();

const B = startOfDayBD(b).getTime();

return Math.round((A-B)/86400000);

}

// ====== STORAGE ======

function loadTasks(){ try { return JSON.parse(localStorage.getItem('tasks\_v1')||'[]'); } catch(e){ return []; } }

function saveTasks(){ localStorage.setItem('tasks\_v1', JSON.stringify(tasks)); }

function loadNotifs(){ try { return JSON.parse(localStorage.getItem('notifs\_v1')||'[]'); } catch(e){ return []; } }

function saveNotifs(){ localStorage.setItem('notifs\_v1', JSON.stringify(notifications)); }

// ====== STATUS LOGIC ======

function deriveStatus(t){

const today = startOfDayBD(nowBD());

const dl = parseISO(t.deadline);

const comp = parseISO(t.completion);

if (comp) {

if (!dl) return 'Completed';

return comp.getTime() <= dl.getTime() ? 'Completed On Time' : 'Completed Late';

}

if (!dl) return 'Open';

const diff = daysDiffBD(dl, today); // positive if deadline after today

if (today.getTime() > dl.getTime()) return 'Overdue';

if (diff === 0) return 'Due Today';

if (diff <= 2) return 'Due Soon';

return 'Open';

}

function statusPill(status){

const cls = status.startsWith('Completed') ? 'completed' :

status === 'Overdue' ? 'overdue' :

status === 'Due Today' ? 'duetoday' :

status === 'Due Soon' ? 'duesoon' : 'open';

return `<span class="pill ${cls}">${status}</span>`;

}

// ====== NOTIFICATIONS ======

function pushNotif(type, title, msg){

const n = { id: 'n'+Date.now()+Math.random().toString(16).slice(2), type, title, msg, ts: nowBD().toISOString(), read:false };

notifications.unshift(n);

saveNotifs();

renderNotifs();

}

function markAllRead(){ notifications = notifications.map(n=>({...n, read:true})); saveNotifs(); renderNotifs(); }

// ====== RENDER ======

const tbody = document.querySelector('#tbl tbody');

const badge = document.getElementById('badge');

const dropdown = document.getElementById('dropdown');

function renderNotifs(){

const unread = notifications.filter(n=>!n.read).length;

badge.textContent = unread;

badge.style.display = unread? 'inline-block':'none';

dropdown.innerHTML = `

<div style="position:sticky; top:0; background:var(--card); padding:10px; display:flex; justify-content:space-between; align-items:center;">

<strong>Notifications</strong>

<button class="btn" onclick="markAllRead()">Mark all read</button>

</div>

${notifications.map(n=>`

<div class="notif">

<div style="font-size:22px">${iconFor(n.type)}</div>

<div>

<div style="font-weight:700">${n.title}</div>

<div>${n.msg}</div>

<small>${new Date(n.ts).toLocaleString()}</small>

</div>

</div>`).join('') || '<div class="notif"><small class="muted">No notifications yet.</small></div>'}

`;

}

function iconFor(type){

if (type==='NEW') return '🆕';

if (type==='REMINDER') return '⏰';

if (type==='DONE') return '✅';

return '🔔';

}

function renderTable(){

// sort by deadline asc, nulls last

tasks.sort((a,b)=>{

const da = a.deadline ? new Date(a.deadline).getTime() : Infinity;

const db = b.deadline ? new Date(b.deadline).getTime() : Infinity;

if (da!==db) return da-db; return (a.task||'').localeCompare(b.task||'');

});

tbody.innerHTML = tasks.map((t,idx)=>{

const status = deriveStatus(t);

return `<tr>

<td><input type="text" value="${escapeHtml(t.task)}" oninput="edit(${idx}, 'task', this.value)" placeholder="Task"></td>

<td><input type="text" value="${escapeHtml(t.details||'')}" oninput="edit(${idx}, 'details', this.value)" placeholder="Details"></td>

<td>${ADMIN\_MODE? `<input type="date" value="${t.deadline||''}" onchange="edit(${idx}, 'deadline', this.value)">` : `<input type="date" value="${t.deadline||''}" onchange="denyEditDeadline(this)" disabled>`}</td>

<td>${statusPill(status)}</td>

<td><input type="date" value="${t.completion||''}" onchange="completeTask(${idx}, this.value)"></td>

<td>${ADMIN\_MODE? `<input type="email" value="${escapeHtml(t.email||'')}" oninput="edit(${idx}, 'email', this.value)" placeholder="Email">` : `<input type="email" value="${escapeHtml(t.email||'')}" disabled>`}</td>

<td><input type="email" value="${escapeHtml(t.assigned||'')}" oninput="edit(${idx}, 'assigned', this.value)" placeholder="Assigned"></td>

<td><input type="email" value="${escapeHtml(t.client||'')}" oninput="edit(${idx}, 'client', this.value)" placeholder="Client"></td>

<td><input type="text" value="${escapeHtml(t.remarks||'')}" oninput="edit(${idx}, 'remarks', this.value)" placeholder="Remarks"></td>

<td><button class="btn" onclick="del(${idx})">Delete</button></td>

</tr>`;

}).join('');

drawChart();

}

function denyEditDeadline(el){

el.value = el.getAttribute('value')||'';

pushNotif('WARN', 'Deadline is locked', 'Enable Admin mode to edit deadlines and Email field.');

renderNotifs();

}

function edit(idx, key, val){

tasks[idx][key] = val;

saveTasks();

renderTable();

}

function del(idx){

const t = tasks[idx];

tasks.splice(idx,1);

saveTasks();

pushNotif('WARN','Task removed', `\"${t.task}\" was deleted.`);

renderTable();

}

function addTask(){

const task = document.getElementById('new-task').value.trim();

const deadline = document.getElementById('new-deadline').value;

const details = document.getElementById('new-details').value.trim();

const email = document.getElementById('new-email').value.trim();

const assigned = document.getElementById('new-assigned').value.trim();

const client = document.getElementById('new-client').value.trim();

const remarks = document.getElementById('new-remarks').value.trim();

if(!task || !deadline){ alert('Task and Deadline are required'); return; }

const t = { id:'t'+Date.now(), task, details, deadline, completion:'', email, assigned, client, remarks, meta:{} };

tasks.push(t); saveTasks();

pushNotif('NEW','New Task Assigned', `Task \"${task}\" — deadline ${deadline}${assigned? ' — assigned to '+assigned: ''}`);

clearForm(); renderTable();

}

function clearForm(){ ['new-task','new-details','new-deadline','new-email','new-assigned','new-client','new-remarks'].forEach(id=>document.getElementById(id).value=''); }

function completeTask(idx, dateStr){

const t = tasks[idx];

const prev = t.completion;

t.completion = dateStr || '';

saveTasks();

const st = deriveStatus(t);

if (dateStr) {

const lateDays = lateByDays(t);

pushNotif('DONE', `Completion — ${st}`, `\"${t.task}\" completed on ${dateStr}${lateDays>0? ' (late by '+lateDays+' day'+(lateDays>1?'s':'')+')':''}.`);

} else if (prev && !dateStr) {

pushNotif('WARN','Completion cleared', `Completion removed for \"${t.task}\".`);

}

renderTable();

}

function lateByDays(t){

const dl = parseISO(t.deadline); const comp = parseISO(t.completion); if(!dl||!comp) return 0;

const diff = (startOfDayBD(comp) - startOfDayBD(dl))/86400000; return Math.max(0, Math.ceil(diff));

}

// ====== REMINDER ENGINE (in-app) ======

function schedulerTick(){

const now = nowBD();

const today = startOfDayBD(now);

const hhmm = now.getHours()\*60 + now.getMinutes();

const isThu = now.getDay()===4; // Sun=0..Sat=6

const isFri = now.getDay()===5; const isSat = now.getDay()===6; const isSun = now.getDay()===0;

tasks.forEach(t=>{

if (t.completion) return;

if (!t.deadline || !t.assigned) return;

const dl = startOfDayBD(parseISO(t.deadline));

if (!dl) return;

const scheduled = new Date(dl.getTime() - 2\*86400000);

// Early send for Fri/Sat schedule: Thursday at/after 16:00

const scheduledDow = scheduled.getDay();

const scheduledKey = fmtDate(scheduled) + ':' + (t.id);

const sentKeys = JSON.parse(localStorage.getItem('dueSentKeys\_v1')||'[]');

const already = sentKeys.includes(scheduledKey);

// Thu early case

if (!already && isThu && (scheduledDow===5 || scheduledDow===6) && hhmm>= (16\*60)) {

pushNotif('REMINDER','Due in 2 days (early)','Task "'+t.task+'" due '+fmtDate(dl)+' — sent early for Fri/Sat');

sentKeys.push(scheduledKey); localStorage.setItem('dueSentKeys\_v1', JSON.stringify(sentKeys));

}

// Regular day at 10:15 (and not Fri/Sat)

if (!already && !isFri && !isSat && fmtDate(scheduled)===fmtDate(today) && hhmm>= (10\*60+15)){

pushNotif('REMINDER','Due in 2 days','Task "'+t.task+'" due '+fmtDate(dl));

sentKeys.push(scheduledKey); localStorage.setItem('dueSentKeys\_v1', JSON.stringify(sentKeys));

}

});

// Weekly kickoff (simple): Sunday 09:45 once per week

if (isSun && hhmm>= (9\*60+45)) {

const weekKey = 'wk\_'+fmtDate(sundayOf(today));

if (!localStorage.getItem(weekKey)){

const summary = weeklySummary();

pushNotif('NEW','Weekly kickoff', summary.note);

localStorage.setItem(weekKey,'1');

}

}

}

function sundayOf(d){ // Sunday of week

const dow = d.getDay(); // 0..6, Sunday=0

const sun = new Date(d.getTime() - dow\*86400000);

sun.setHours(0,0,0,0); return sun;

}

function weeklySummary(){

const today = startOfDayBD(nowBD());

const start = new Date(today.getTime() - 7\*86400000); // last week window end yesterday

const days = [...Array(7)].map((\_,i)=> new Date(start.getTime()+i\*86400000));

let on=0, late=0;

tasks.forEach(t=>{

if (!t.completion) return;

const c = startOfDayBD(parseISO(t.completion));

if (c < days[0] || c > days[6]) return;

const dld = parseISO(t.deadline);

if (dld && parseISO(t.completion) <= dld) on++; else late++;

});

const rate = (on+late)? Math.round(on/(on+late)\*100):0;

return { note: `Last week on-time: ${rate}% (on-time ${on}, late ${late})` };

}

// ====== CHART ======

function drawChart(){

const canvas = document.getElementById('chart');

const ctx = canvas.getContext('2d');

ctx.clearRect(0,0,canvas.width,canvas.height);

const today = startOfDayBD(nowBD());

const start = new Date(today.getTime() - 6\*86400000);

const days = [...Array(7)].map((\_,i)=> new Date(start.getTime()+i\*86400000));

const data = days.map(day=>({on:0, late:0}));

tasks.forEach(t=>{

if (!t.completion) return;

const c = startOfDayBD(parseISO(t.completion));

if (c < days[0] || c > days[6]) return;

const idx = Math.round((c - days[0])/86400000);

const dld = parseISO(t.deadline);

if (dld && parseISO(t.completion) <= dld) data[idx].on++; else data[idx].late++;

});

// axes

const pad = {l:40, r:10, t:10, b:24};

const W = canvas.width - pad.l - pad.r; const H = canvas.height - pad.t - pad.b;

ctx.strokeStyle = 'rgba(255,255,255,.15)';

ctx.beginPath(); ctx.moveTo(pad.l, pad.t); ctx.lineTo(pad.l, pad.t+H); ctx.lineTo(pad.l+W, pad.t+H); ctx.stroke();

const maxY = Math.max(3, ...data.map(d=>d.on+d.late));

const barW = W/7 \* 0.6; const gap = W/7 \* 0.4;

days.forEach((day,i)=>{

const x = pad.l + i\*(barW+gap) + gap/2; const total = data[i].on+data[i].late;

const hOn = H \* (data[i].on/maxY), hLate = H \* (data[i].late/maxY);

// late

ctx.fillStyle = 'rgba(255,107,107,.8)';

ctx.fillRect(x, pad.t + (H - hLate), barW, hLate);

// on

ctx.fillStyle = 'rgba(54,211,153,.9)';

ctx.fillRect(x, pad.t + (H - hLate - hOn), barW, hOn);

// label

ctx.fillStyle = 'rgba(255,255,255,.7)';

ctx.font = '12px system-ui, -apple-system, Segoe UI';

const lbl = day.toLocaleDateString(undefined,{ weekday:'short'});

ctx.fillText(lbl, x, pad.t+H+16);

});

// metric

const on = data.reduce((s,d)=>s+d.on,0), late = data.reduce((s,d)=>s+d.late,0);

document.getElementById('metric').textContent = `On-time: ${on} Late: ${late}`;

}

// ====== EVENTS ======

document.getElementById('btn-add').onclick = addTask;

document.getElementById('btn-clear').onclick = clearForm;

document.getElementById('btn-admin').onclick = ()=>{ ADMIN\_MODE = !ADMIN\_MODE; document.getElementById('admin-mode').textContent = ADMIN\_MODE? 'ON':'OFF'; renderTable(); };

document.getElementById('btn-export').onclick = ()=>{

const blob = new Blob([JSON.stringify(tasks,null,2)], {type:'application/json'});

const a = document.createElement('a');

a.href = URL.createObjectURL(blob);

a.download = 'tasks.json'; a.click(); URL.revokeObjectURL(a.href);

};

document.getElementById('import-file').onchange = (e)=>{

const file = e.target.files[0]; if(!file) return;

const reader = new FileReader(); reader.onload = ()=>{ try{ tasks = JSON.parse(reader.result); saveTasks(); renderTable(); pushNotif('NEW','Imported','Tasks imported from JSON'); }catch(err){ alert('Invalid JSON'); } };

reader.readAsText(file);

};

document.getElementById('btn-bell').onclick = ()=>{ dropdown.classList.toggle('open'); renderNotifs(); };

// ====== INIT ======

renderTable();

renderNotifs();

// periodic scheduler (check every 60s)

setInterval(schedulerTick, 60\*1000);

schedulerTick();

// helper

function escapeHtml(s){ return (s||'').replace(/[&<>"']/g, c=>({"&":"&amp;","<":"&lt;",">":"&gt;","\"":"&quot;","'":"&#39;"}[c])); }

</script>

</body>

</html>