# 📘 DRAFT BUKU: BIKIN APLIKASI TANPA JADI PROGRAMMER

**Panduan untuk Orang Sibuk yang Butuh Solusi, Bukan Teori**

*Cara Merencanakan Sistem Digital dari Nol - AI yang Coding, Anda yang Mikir*

**Tagline:** *“Marie Kondo untuk Aplikasi: Buang yang Ribet, Simpan yang Penting”*

## 📑 DAFTAR ISI

### **BAGIAN 1: MINDSET (Bab 1-3)**

* **Bab 1:** Mengapa Anda TIDAK Perlu Jadi Programmer
* **Bab 2:** Kompleksitas itu Ilusi, Kesederhanaan itu Kemenangan
* **Bab 3:** Skill yang Benar-Benar Anda Butuhkan

### **BAGIAN 2: PERENCANAAN (Bab 4-7)**

* **Bab 4:** Dari Masalah ke Solusi dalam 5 Langkah
* **Bab 5:** Gambar Kotak-Kotak: Arsitektur untuk Manusia Biasa
* **Bab 6:** Database Tanpa Pusing: Spreadsheet Sebagai Otak Aplikasi
* **Bab 7:** Checklist Anti-Gagal: Validasi Ide Sebelum Coding

### **BAGIAN 3: EKSEKUSI (Bab 8-12)**

* **Bab 8:** Google Gratis: Sheets + Apps Script = Backend Instant
* **Bab 9:** Studi Kasus: Task Manager dari Nol dalam 30 Menit
* **Bab 10:** CRUD: 4 Fungsi yang Bisa Solve 90% Masalah
* **Bab 11:** Deploy ke Internet (Gratis, Tanpa Server)
* **Bab 12:** Maintenance: Update Tanpa Coding

### **BAGIAN 4: SCALE UP (Bab 13-15)**

* **Bab 13:** Dari 1 App ke 10 Apps dengan Template yang Sama
* **Bab 14:** Multi-Database: Finance, Inventory, HR dalam Satu Sistem
* **Bab 15:** Kapan Waktunya Hire Programmer (Hint: Nanti Banget)

### **BONUS: TEMPLATES**

* Template Task Manager
* Template Inventory/Stok
* Template Absensi
* Template Finance Tracker

## 📖 OUTLINE LENGKAP

### **BAB 1: MENGAPA ANDA TIDAK PERLU JADI PROGRAMMER**

**Opening Hook:** > “Anda tidak perlu tahu cara mesin mobil bekerja untuk bisa nyetir ke kantor.  
> Begitu juga aplikasi: Anda tidak perlu jadi programmer untuk punya sistem digital.”

**Isi Bab:** - Mitos: “Bikin aplikasi = harus bisa coding” - Realita: 90% orang butuh CRUD sederhana, bukan NASA rocket science - Bedanya “Builder” vs “Coder” (Anda adalah Builder) - Case study: 3 owner UMKM yang bikin app tanpa background IT

**Takeaway:** - Skill yang dibutuhkan: Problem solving, bukan sintaks Python - Tools sudah ada (gratis!), tinggal pakai - Fokus ke: “Apa masalahnya?” bukan “Pakai framework apa?”

### **BAB 2: KOMPLEKSITAS ITU ILUSI, KESEDERHANAAN ITU KEMENANGAN**

**Opening Hook:** > “Aplikasi terbaik bukan yang paling canggih, tapi yang paling DIPAKAI.”

**Isi Bab:** - Filosofi Minimalis: Less is More - Mengapa 2 file HTML bisa mengalahkan React + Node + MongoDB - Perbandingan: - **Aplikasi Kompleks:** 50+ files, butuh server, mahal maintenance → user bingung - **Aplikasi Simple:** 2 files, Google Sheets, gratis → user happy - Prinsip 4-Files Champion: HTML + Sheets + Apps Script = Done

**Studi Kasus Real:** - Bengkel mobil: Pakai spreadsheet + form → tracking 500 customer - Warung makan: Apps Script + Sheets → inventory otomatis update - Guru SD: Google Forms + Sheets → nilai siswa 200 anak

**Takeaway:** - Start with spreadsheet mindset, bukan database mindset - Kalau bisa 2 files, jangan pakai 200 files - User peduli hasil, bukan teknologi di belakangnya

### **BAB 3: SKILL YANG BENAR-BENAR ANDA BUTUHKAN**

**3 Skill Inti:**

**1. Problem Breakdown (Pecah Masalah)** - Contoh: “Bengkel saya kacau” - Pecah jadi: Pelanggan siapa? Motor apa? Kapan selesai? Berapa bayar? - Dari 1 masalah besar → 4 data kecil → beres!

**2. Data Thinking (Pikir Data)** - Setiap masalah = data yang belum tertata - Template berpikir: - Masalah: Lupa deadline task - Data: Nama task, PIC, Tanggal deadline, Status - Solusi: Table dengan 4 kolom itu!

**3. Prototype Sketching (Gambar di Kertas)** - Kotak = screen - Arrow = navigasi - Tidak perlu bagus, yang penting jelas - Contoh: Login → Dashboard → Add Task → Done (4 kotak aja!)

**Bonus Skill:** - Google-fu: Cari tutorial dengan keyword yang tepat - Copy-Paste Strategis: Modifikasi code orang lain (legal & smart!) - Prompting AI: Cara ngomong ke ChatGPT/Claude agar jawab yang bener

**Exercise:** - Gambar aplikasi impian Anda di kertas (15 menit) - List 5 data yang perlu disimpan - Tulis user flow: User datang → User ngapain → User pergi

**Takeaway:** - Koding itu 10%, planning itu 90% - Kalau planning jelas, AI bisa coding-in dalam hitungan menit - Skill ini bisa dipelajari dalam 1 hari, bukan 4 tahun kuliah

### **BAB 4: DARI MASALAH KE SOLUSI DALAM 5 LANGKAH**

**Framework “5-Step App Builder”:**

**Step 1: Problem Statement (1 Kalimat)** - Contoh BURUK: “Perusahaan saya tidak efisien” - Contoh BAGUS: “Karyawan lupa task karena pakai WhatsApp grup (500 chat/hari)”

**Step 2: User Flow (3-5 Langkah)** - User login → Lihat daftar task → Klik task → Tandai selesai → Logout - Jangan lebih dari 5 langkah, kalau kebanyakan = over-engineering

**Step 3: Data Structure (Tabel Spreadsheet)** - Buka Google Sheets, buat header: - Task: ID, Nama Task, PIC, Deadline, Status, Catatan - User: Username, Password, Role - Isi 3-5 row contoh data (dummy ok, asal realistis)

**Step 4: UI Sketching (Gambar 3 Screen)** - Screen 1: Login (username + password + button) - Screen 2: Dashboard (list task + button add) - Screen 3: Add Task (form + button save)

**Step 5: Validation Checklist** - [ ] Apakah solve masalah utama? - [ ] Apakah user flow < 5 langkah? - [ ] Apakah data bisa muat di 1 spreadsheet? - [ ] Apakah bisa dijelaskan ke nenek dalam 30 detik?

**Contoh Penerapan:** - Masalah: Owner kos bingung tracking pembayaran 20 kamar - User Flow: Login → Lihat daftar kamar → Klik kamar → Input bayar → Update status - Data: Kamar (Nomor, Penghuni, Harga, Status Bayar, Tanggal Bayar) - UI: 3 screen (Login, Dashboard kamar, Form bayar) - Validation: ✅ Semua yes? Gas build!

**Exercise:** - Pilih 1 masalah di hidup Anda (kantor/rumah/usaha) - Jalankan 5 langkah di atas (30 menit) - Share ke teman, minta feedback: “Apa yang kurang jelas?”

**Takeaway:** - 30 menit planning = hemat 30 jam coding salah arah - Kalau bingung di step 2-3, berarti masalahnya belum jelas (balik ke step 1) - Validation di step 5 = filter ide bagus vs ide mubazir

### **BAB 5: GAMBAR KOTAK-KOTAK: ARSITEKTUR UNTUK MANUSIA BIASA**

**Mengapa Arsitektur Itu Penting?** - Tanpa peta: coding asal-asalan, bug dimana-mana, bingung sendiri - Dengan peta: tau persis komponen apa, ngapain, nyambung kemana

**3 Kotak Ajaib Setiap Aplikasi:**

┌─────────────┐  
│ FRONTEND │ ← Yang user lihat (HTML)  
│ (Tampilan) │  
└──────┬──────┘  
 │ (Kirim data)  
 ▼  
┌─────────────┐  
│ BACKEND │ ← Logic & keamanan (Apps Script)  
│ (Otak) │  
└──────┬──────┘  
 │ (Simpan/ambil data)  
 ▼  
┌─────────────┐  
│ DATABASE │ ← Penyimpanan (Google Sheets)  
│ (Memori) │  
└─────────────┘

**Template Arsitektur Task Manager:**

1. **Frontend (2 File HTML)**
   * login.html: Form username + password
   * dashboard.html: List task + button add/edit/delete
2. **Backend (1 Apps Script)**
   * Function: validateUser(), getTasks(), addTask(), completeTask(), deleteTask()
   * Komunikasi: POST request dari frontend
3. **Database (1 Google Sheets)**
   * Tab 1 (user): Username, Password, Role
   * Tab 2 (db01): ID Task, Nama, PIC, Status, Timestamp

**Cara Gambar Arsitektur (Pen & Paper):** 1. Gambar 3 kotak vertikal (Frontend, Backend, Database) 2. Tambah panah data flow: - User klik button → POST ke Backend → Backend cek Sheets → Balik ke Frontend 3. Tulis fungsi di setiap kotak: - Frontend: Tampil, Input - Backend: Validasi, CRUD - Database: Simpan

**Contoh Kasus Lain:**

**A. Inventory Toko**

Frontend:   
- List barang (foto, nama, stok)  
- Form tambah barang  
  
Backend:  
- addItem(), updateStock(), getItems()  
  
Database:  
- Tab `inventory`: ID, Nama, Stok, Harga, Foto URL

**B. Absensi Karyawan**

Frontend:  
- Tombol "Check In" / "Check Out"  
- List history absensi  
  
Backend:  
- recordAttendance(), getHistory()  
  
Database:  
- Tab `attendance`: ID, Nama, Tanggal, Jam Masuk, Jam Keluar

**Exercise:** - Pilih 1 ide aplikasi Anda - Gambar 3 kotak (Frontend, Backend, Database) - List 3-5 function di Backend - List kolom-kolom di Database

**Takeaway:** - Arsitektur = peta jalan, bukan penjara - Kalau bisa dijelaskan dalam 3 kotak, berarti cukup simple - Save gambar ini, nanti kasih ke AI untuk generate code

### **BAB 6: DATABASE TANPA PUSING: SPREADSHEET SEBAGAI OTAK APLIKASI**

**Mengapa Google Sheets = Database Terbaik untuk Pemula?**

| Google Sheets | Database Tradisional (MySQL/PostgreSQL) |
| --- | --- |
| Gratis selamanya | Butuh hosting berbayar |
| Bisa diedit manual | Harus pakai SQL command |
| Familiar (kayak Excel) | Perlu belajar syntax baru |
| Backup otomatis | Harus setup sendiri |
| Bisa dibuka di HP | Perlu tool khusus |

**Struktur Database yang Benar:**

**1. Gunakan Multiple Tabs (Sheets) untuk Tabel Berbeda**

Sheet 1: user (Data login)  
Sheet 2: db01 (Data task)  
Sheet 3: logs (Optional: tracking aktivitas)

**2. Naming Convention** - Tab: lowercase, singkat (user, db01, bukan User Data Task Manager v2) - Kolom: camelCase (itemName, bukan Item Name) - ID: Prefix + Number + Huruf (001AB, TR0012)

**3. Data Types yang Support** - Text: Nama, alamat, catatan - Number: Harga, qty, score - Date: Timestamp ISO format 2025-09-30T10:30:00 - Boolean: Pakai text (true/false atau yes/no)

**4. Status Management** Untuk kolom status, pilih nilai yang limitied: - Task: onProgress, overdue, done (jangan sampai ada typo!) - Payment: pending, paid, failed - Approval: waiting, approved, rejected

**Template Database Task Manager:**

**Tab: user** | username | password | role | created | |———-|———-|——|———| | admin | admin123 | admin | 2025-09-30 | | staff1 | pass1 | user | 2025-09-30 |

**Tab: db01** | id | itemName | pic | status | tsCreated | tsCompleted | completionNote | |—-|———-|—–|——–|———–|————-|—————-| | 001AB | Meeting | admin | done | 2025-09-30T10:00 | 2025-09-30T15:00 | Selesai tepat waktu |

**Tab: logs** (Bonus) | timestamp | username | action | details | |———–|———-|——–|———| | 2025-09-30T10:30 | admin | login | Success | | 2025-09-30T10:31 | admin | addTask | Task: Meeting |

**Validation Rules:** 1. **ID harus unique** (generate di Backend, jangan manual) 2. **Required fields** tidak boleh kosong (validasi di Frontend) 3. **Status harus dari list** (dropdown validation di Sheets) 4. **Timestamp otomatis** (generate di Backend pakai new Date())

**Tips Pro:** - Freeze header row (View → Freeze → 1 row) - Data validation di kolom status (Data → Data validation → List) - Conditional formatting untuk visualisasi: - Status done → hijau - Status overdue → merah - Status onProgress → kuning

**Exercise:** - Buat Google Sheets baru - Setup 2 tabs: user dan tasks - Isi 5 dummy data di masing-masing tab - Test edit manual, pastikan understand struktur

**Takeaway:** - Database = spreadsheet yang disiplin strukturnya - Multiple tabs = multiple tables (relasi sederhana) - Sheets bisa handle 10 juta cell (cukup untuk 100rb+ data!)

### **BAB 7: CHECKLIST ANTI-GAGAL: VALIDASI IDE SEBELUM CODING**

**Pre-Coding Checklist (30 Menit):**

**✅ Problem Clarity** - [ ] Bisa jelaskan masalah dalam 1 kalimat? - [ ] Ada user yang confirm ini emang masalah real? - [ ] Pernah pakai solusi manual (Excel/WhatsApp) dan gagal?

**✅ Scope Definition** - [ ] User flow < 5 langkah? - [ ] Screen count < 5 halaman? - [ ] Database table < 5 tabs?

**✅ Data Readiness** - [ ] Sudah buat dummy data di Sheets? - [ ] Struktur data bisa dijelaskan ke orang lain? - [ ] Tidak ada kolom “lain-lain” atau “misc”? (red flag!)

**✅ UI Sketch** - [ ] Sudah gambar 3 screen utama di kertas? - [ ] Orang lain bisa understand flow-nya? - [ ] Tidak ada button “Advanced Settings”? (YAGNI!)

**✅ Technical Feasibility** - [ ] Pakai Google Sheets cukup? (bukan perlu Firebase?) - [ ] Pakai HTML + JS cukup? (bukan perlu React?) - [ ] Gratis selamanya? (bukan butuh server $50/bulan?)

**Red Flags yang Harus Dihindari:**

🚩 **Feature Creep** - “Nanti bisa export PDF, kan?” - “Tambahin dashboard grafik, dong?” - **Solution:** Catat di list “V2 Nanti”, fokus V1 dulu!

🚩 **Over-Engineering** - “Pakai framework apa? React atau Vue?” - “Database-nya pakai PostgreSQL atau MongoDB?” - **Solution:** Kalau Google Sheets cukup, ya Sheets aja!

🚩 **Unclear User** - “Ini aplikasi untuk siapa?” - “Entah, kayaknya general aja deh…” - **Solution:** Specific! “Untuk owner bengkel motor dengan 1-5 mekanik”

🚩 **No MVP** - “Harus sempurna dari awal!” - “Belum bisa demo, baru 80% jadi” - **Solution:** Launch 50% working, iterate cepat!

**Validation Techniques:**

**1. Paper Prototype Test** - Print UI sketch - Kasih ke calon user - Suruh “pakai” dengan jari (pura-pura klik) - Catat: Di mana mereka bingung?

**2. Spreadsheet Simulation** - Isi Google Sheets dengan 20-30 dummy data - Coba filter, sort, search manual - Kalau manual aja ribet, berarti struktur salah

**3. Explain to Grandma** - Jelaskan aplikasi Anda ke orang non-IT - Kalau mereka nggak paham dalam 60 detik, redesign!

**Exercise: Full Validation Session (60 Menit)**

1. **Tulis Problem Statement** (10 menit)
2. **Gambar UI di kertas** (15 menit)
3. **Buat struktur Sheets + dummy data** (20 menit)
4. **Paper prototype test dengan teman** (10 menit)
5. **Revisi berdasarkan feedback** (5 menit)

**Takeaway:** - 1 jam validation = hemat 10 jam revisi nanti - Kalau gagal di paper prototype, akan gagal di code juga - Red flag adalah teman, bukan musuh (early warning system!)

### **BAB 8: GOOGLE GRATIS: SHEETS + APPS SCRIPT = BACKEND INSTANT**

**Kenalan dengan Google Apps Script:**

**Apa itu Apps Script?** - JavaScript yang jalan di server Google (gratis!) - Punya akses penuh ke Google Sheets, Gmail, Calendar, Drive - Bisa di-deploy sebagai Web App (API instant!)

**Keuntungan Pakai Apps Script:** - **Gratis:** 20,000 calls/day (cukup untuk 1000+ user) - **Serverless:** Tidak perlu sewa VPS/server - **Auto-scale:** Google yang handle traffic spike - **Built-in Auth:** Integrasi Google Account langsung - **5 Menit Setup:** Tidak perlu install apa-apa

**Setup Pertama Kali (5 Menit):**

1. Buka Google Sheets → Extensions → Apps Script
2. Hapus code default, paste template:

function doGet(e) {  
 return ContentService.createTextOutput('API is running! ✅');  
}  
  
function doPost(e) {  
 const data = JSON.parse(e.postData.contents);  
 return ContentService  
 .createTextOutput(JSON.stringify({  
 success: true,  
 data: data  
 }))  
 .setMimeType(ContentService.MimeType.JSON);  
}

1. Deploy → New deployment → Web app → Anyone → Deploy
2. Copy URL: https://script.google.com/macros/s/ABC.../exec
3. Test di browser → Muncul “API is running! ✅”

**Struktur Code Apps Script:**

// ========== HELPERS ==========  
function getSheet(name) {  
 return SpreadsheetApp.getActiveSpreadsheet()  
 .getSheetByName(name);  
}  
  
function success(data) {  
 return ContentService  
 .createTextOutput(JSON.stringify({  
 success: true,  
 data: data  
 }))  
 .setMimeType(ContentService.MimeType.JSON);  
}  
  
function error(message) {  
 return ContentService  
 .createTextOutput(JSON.stringify({  
 success: false,  
 error: message  
 }))  
 .setMimeType(ContentService.MimeType.JSON);  
}  
  
// ========== AUTH ==========  
function validateUser(username, password) {  
 const sheet = getSheet('user');  
 const data = sheet.getDataRange().getValues();  
   
 for (let i = 1; i < data.length; i++) {  
 if (data[i][0] === username && data[i][1] === password) {  
 return true;  
 }  
 }  
 return false;  
}  
  
// ========== CRUD ==========  
function getTasks() {  
 const sheet = getSheet('db01');  
 const data = sheet.getRange('A2:G').getValues();  
   
 return data  
 .filter(row => row[0])  
 .map(row => ({  
 id: row[0],  
 itemName: row[1],  
 pic: row[2],  
 status: row[3],  
 tsCreated: row[4]  
 }));  
}  
  
function addTask(itemName, username) {  
 const sheet = getSheet('db01');  
 const id = generateId();  
 const now = new Date().toISOString();  
   
 sheet.appendRow([id, itemName, username, 'onProgress', now, '', '']);  
   
 return { id, itemName, pic: username, status: 'onProgress' };  
}  
  
// ========== MAIN ENDPOINT ==========  
function doPost(e) {  
 try {  
 const params = JSON.parse(e.postData.contents);  
   
 if (!validateUser(params.username, params.password)) {  
 return error('Invalid credentials');  
 }  
   
 switch(params.action) {  
 case 'login':  
 return success({ message: 'Login success' });  
   
 case 'getTasks':  
 return success(getTasks());  
   
 case 'addTask':  
 return success(addTask(params.itemName, params.username));  
   
 default:  
 return error('Unknown action');  
 }  
 } catch (err) {  
 return error(err.toString());  
 }  
}

**Pattern yang Harus Dipahami:**

**1. Read Data (SELECT)**

const sheet = getSheet('user');  
const data = sheet.getDataRange().getValues();  
// data = [[header...], [row1...], [row2...]]

**2. Write Data (INSERT)**

sheet.appendRow(['ID123', 'Task Name', 'John', 'onProgress']);

**3. Update Data (UPDATE)**

sheet.getRange(rowNumber, colNumber).setValue('newValue');

**4. Delete Data (DELETE)**

sheet.deleteRow(rowNumber);

**Quota & Limits:** - 20,000 URL fetches/day (request ke API) - 6 menit/execution (timeout) - 50 MB response size - **Kesimpulan:** Cukup untuk 1000+ daily active users!

**Exercise:** 1. Buat Google Sheets baru dengan tab user 2. Setup Apps Script dengan code di atas 3. Deploy sebagai Web App 4. Test via browser (GET) 5. Test via Postman/curl (POST)

**Takeaway:** - Apps Script = backend gratis tanpa server - Belajar 5 pattern (read, write, update, delete, auth) = unlock 90% use case - Deploy dalam 5 menit, not 5 days!

### **BAB 9: STUDI KASUS: TASK MANAGER DARI NOL DALAM 30 MENIT**

**Full Timeline Build:**

**[Menit 0-10] Planning & Setup** 1. Buat Google Sheets: “Task Manager Database” 2. Setup 2 tabs: - user: username, password, role - db01: id, itemName, pic, status, tsCreated, tsCompleted, completionNote 3. Isi dummy data 4 user + 3 task 4. Freeze header row + data validation status

**[Menit 10-15] Backend (Apps Script)** 1. Extensions → Apps Script 2. Copy-paste template code (dari Bab 8) 3. Save + Deploy as Web App 4. Test GET endpoint di browser 5. Copy URL untuk frontend

**[Menit 15-25] Frontend (login.html)** 1. Buat file login.html 2. Structure: - Form: username + password input - Button: Login - Error div (hidden default) 3. JavaScript: - Fetch API\_URL dengan POST - Action: “login” - Success: Simpan user ke sessionStorage, redirect ke dashboard - Error: Tampilkan pesan

**[Menit 25-30] Frontend (dashboard.html)** 1. Buat file dashboard.html 2. Structure: - Header: User info + Logout button - Form tambah task (hidden default) - Table list tasks 3. JavaScript: - Load tasks on page load - Add task function - Complete task function - Delete task function - Auto refresh every 30s

**[Menit 30] Testing & Deploy** 1. Test login dengan user dummy 2. Test CRUD (add, complete, delete) 3. Upload ke Netlify (drag & drop folder) 4. Done! 🎉

**Code Snippet Penting:**

**login.html (Core Logic)**

const API\_URL = 'YOUR\_APPS\_SCRIPT\_URL';  
  
form.addEventListener('submit', async (e) => {  
 e.preventDefault();  
 const username = usernameInput.value;  
 const password = passwordInput.value;  
   
 const response = await fetch(API\_URL, {  
 method: 'POST',  
 body: JSON.stringify({  
 action: 'login',  
 username: username,  
 password: password  
 })  
 });  
   
 const result = await response.json();  
   
 if (result.success) {  
 sessionStorage.setItem('user', username);  
 sessionStorage.setItem('pass', password);  
 window.location.href = 'dashboard.html';  
 } else {  
 showError('Login gagal!');  
 }  
});

**dashboard.html (Load Tasks)**

async function loadTasks() {  
 const response = await fetch(API\_URL, {  
 method: 'POST',  
 body: JSON.stringify({  
 action: 'getTasks',  
 username: sessionStorage.getItem('user'),  
 password: sessionStorage.getItem('pass')  
 })  
 });  
   
 const result = await response.json();  
   
 if (result.success) {  
 renderTasks(result.data);  
 }  
}  
  
function renderTasks(tasks) {  
 const html = tasks.map(task => `  
 <tr>  
 <td>${task.itemName}</td>  
 <td>${task.pic}</td>  
 <td class="status-${task.status}">${task.status}</td>  
 <td>${formatDate(task.tsCreated)}</td>  
 <td>  
 <button onclick="completeTask('${task.id}')">✓</button>  
 <button onclick="deleteTask('${task.id}')">✗</button>  
 </td>  
 </tr>  
 `).join('');  
   
 tableBody.innerHTML = html;  
}

**Analisis Kompleksitas:**

| Component | Files | Lines of Code | Time |
| --- | --- | --- | --- |
| Database | 1 Sheets | 0 (manual setup) | 5 min |
| Backend | 1 Apps Script | ~150 lines | 5 min |
| Frontend Login | 1 HTML | ~100 lines | 10 min |
| Frontend Dashboard | 1 HTML | ~250 lines | 10 min |
| **TOTAL** | **4 files** | **~500 lines** | **30 min** |

**Fitur yang Tercover:** ✅ Authentication (login/logout) ✅ Create task ✅ Read tasks (list + auto refresh) ✅ Update task (complete with note) ✅ Delete task ✅ Auto check overdue (background job) ✅ Responsive design ✅ Error handling

**What If You Need More?**

**Tambah Fitur “Filter by Date”** (+5 menit):

function filterByDate(dateFrom, dateTo) {  
 const filtered = tasks.filter(task => {  
 const created = new Date(task.tsCreated);  
 return created >= dateFrom && created <= dateTo;  
 });  
 renderTasks(filtered);  
}

**Tambah Fitur “Search by Name”** (+5 menit):

function searchTasks(keyword) {  
 const filtered = tasks.filter(task =>   
 task.itemName.toLowerCase().includes(keyword.toLowerCase())  
 );  
 renderTasks(filtered);  
}

**Tambah Fitur “Export to CSV”** (+10 menit):

function exportCSV() {  
 const csv = tasks.map(t =>   
 `${t.id},${t.itemName},${t.pic},${t.status}`  
 ).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 = 'tasks.csv';  
 a.click();  
}

**Exercise:** 1. Follow timeline 30 menit di atas 2. Build Task Manager dari nol 3. Test semua fitur CRUD 4. Deploy ke Netlify 5. Share URL ke teman, minta feedback

**Takeaway:** - 30 menit real, bukan clickbait (dengan template) - 500 lines code untuk production-ready app (not bad!) - Template ini bisa di-reuse untuk 10+ apps berbeda

### **BAB 10: CRUD: 4 FUNGSI YANG BISA SOLVE 90% MASALAH**

**CRUD = Create, Read, Update, Delete**

**Mengapa CRUD Itu Powerful?** - 90% aplikasi bisnis = CRUD + sedikit logic - Kalau sudah paham CRUD, bisa build: - Task Manager - Inventory System - CRM (Customer Relationship Management) - HR Absensi - Finance Tracker - dll.

**Template CRUD Universal:**

**1. CREATE (Tambah Data)**

**Apps Script:**

function addItem(data) {  
 const sheet = getSheet('db01');  
 const id = generateId();  
 const now = new Date().toISOString();  
   
 sheet.appendRow([  
 id,  
 data.name,  
 data.category,  
 data.price,  
 now  
 ]);  
   
 return { id, ...data, createdAt: now };  
}

**Frontend:**

async function create() {  
 const response = await fetch(API\_URL, {  
 method: 'POST',  
 body: JSON.stringify({  
 action: 'addItem',  
 username: sessionStorage.getItem('user'),  
 password: sessionStorage.getItem('pass'),  
 data: {  
 name: itemName.value,  
 category: category.value,  
 price: price.value  
 }  
 })  
 });  
   
 const result = await response.json();  
 if (result.success) {  
 alert('Item added!');  
 loadItems();  
 }  
}

**2. READ (Ambil Data)**

**Apps Script:**

function getItems() {  
 const sheet = getSheet('db01');  
 const data = sheet.getRange('A2:E').getValues();  
   
 return data  
 .filter(row => row[0])  
 .map(row => ({  
 id: row[0],  
 name: row[1],  
 category: row[2],  
 price: row[3],  
 createdAt: row[4]  
 }));  
}

**Frontend:**

async function loadItems() {  
 const response = await fetch(API\_URL, {  
 method: 'POST',  
 body: JSON.stringify({  
 action: 'getItems',  
 username: sessionStorage.getItem('user'),  
 password: sessionStorage.getItem('pass')  
 })  
 });  
   
 const result = await response.json();  
 if (result.success) {  
 renderItems(result.data);  
 }  
}  
  
function renderItems(items) {  
 const html = items.map(item => `  
 <div class="item-card">  
 <h3>${item.name}</h3>  
 <p>${item.category} - $${item.price}</p>  
 <button onclick="editItem('${item.id}')">Edit</button>  
 <button onclick="deleteItem('${item.id}')">Delete</button>  
 </div>  
 `).join('');  
   
 container.innerHTML = html;  
}

**3. UPDATE (Edit Data)**

**Apps Script:**

function updateItem(id, newData) {  
 const sheet = getSheet('db01');  
 const data = sheet.getDataRange().getValues();  
   
 for (let i = 1; i < data.length; i++) {  
 if (data[i][0] === id) {  
 sheet.getRange(i + 1, 2).setValue(newData.name);  
 sheet.getRange(i + 1, 3).setValue(newData.category);  
 sheet.getRange(i + 1, 4).setValue(newData.price);  
 return true;  
 }  
 }  
 return false;  
}

**Frontend:**

async function updateItem(id) {  
 const response = await fetch(API\_URL, {  
 method: 'POST',  
 body: JSON.stringify({  
 action: 'updateItem',  
 username: sessionStorage.getItem('user'),  
 password: sessionStorage.getItem('pass'),  
 id: id,  
 data: {  
 name: itemName.value,  
 category: category.value,  
 price: price.value  
 }  
 })  
 });  
   
 const result = await response.json();  
 if (result.success) {  
 alert('Item updated!');  
 loadItems();  
 }  
}

**4. DELETE (Hapus Data)**

**Apps Script:**

function deleteItem(id) {  
 const sheet = getSheet('db01');  
 const data = sheet.getDataRange().getValues();  
   
 for (let i = 1; i < data.length; i++) {  
 if (data[i][0] === id) {  
 sheet.deleteRow(i + 1);  
 return true;  
 }  
 }  
 return false;  
}

**Frontend:**

async function deleteItem(id) {  
 if (!confirm('Yakin mau hapus?')) return;  
   
 const response = await fetch(API\_URL, {  
 method: 'POST',  
 body: JSON.stringify({  
 action: 'deleteItem',  
 username: sessionStorage.getItem('user'),  
 password: sessionStorage.getItem('pass'),  
 id: id  
 })  
 });  
   
 const result = await response.json();  
 if (result.success) {  
 alert('Item deleted!');  
 loadItems();  
 }  
}

**CRUD Pattern di Real Use Cases:**

**A. Inventory Toko** - CREATE: Tambah barang baru - READ: Lihat daftar barang + stok - UPDATE: Update stok setelah terjual - DELETE: Hapus barang discontinue

**B. Absensi Karyawan** - CREATE: Check in (record timestamp) - READ: Lihat history absensi bulan ini - UPDATE: Koreksi jam masuk (salah input) - DELETE: Hapus record duplikat

**C. Customer Database (CRM)** - CREATE: Tambah customer baru - READ: Lihat list customer + filter - UPDATE: Update status (lead → paying) - DELETE: Hapus customer inactive

**Tips Optimasi:**

**1. Bulk Operations** Kalau butuh hapus/update banyak data:

function bulkDelete(ids) {  
 ids.forEach(id => deleteItem(id));  
}

**2. Pagination** Kalau data > 100 rows:

function getItemsPaginated(page, limit) {  
 const start = (page - 1) \* limit;  
 const items = getItems();  
 return items.slice(start, start + limit);  
}

**3. Search & Filter** Combine dengan READ:

function searchItems(keyword) {  
 const items = getItems();  
 return items.filter(item =>   
 item.name.toLowerCase().includes(keyword.toLowerCase())  
 );  
}

**Exercise:** 1. Copy template CRUD di atas 2. Modifikasi untuk use case Anda (Inventory/CRM/HR) 3. Test semua 4 fungsi 4. Tambah 1 fitur bonus (search/filter/export)

**Takeaway:** - CRUD = 80% dari semua aplikasi bisnis - Template ini copy-paste ready - Kalau CRUD sudah jalan, sisanya tinggal polish UI

### **BAB 11: DEPLOY KE INTERNET (GRATIS, TANPA SERVER)**

**Mengapa Perlu Deploy?** - File lokal hanya bisa diakses di komputer Anda - Deploy = aplikasi bisa diakses dari mana saja (HP, tablet, komputer lain) - Dapat URL permanen untuk share ke user/client

**Top 3 Hosting Gratis (Recommended):**

**1. NETLIFY** ⭐⭐⭐⭐⭐

**Keuntungan:** - Drag & drop folder → instant deploy - SSL gratis (HTTPS otomatis) - CDN global (loading cepat dari mana saja) - 100 GB bandwidth/bulan (cukup untuk 10,000+ visitor) - Custom domain support

**Cara Deploy (5 Menit):** 1. Buka [netlify.com](https://netlify.com) 2. Sign up dengan email/GitHub 3. Klik “Add new site” → “Deploy manually” 4. **Drag folder public/** ke area upload 5. Tunggu 30 detik → Done! 6. URL: https://random-name-123.netlify.app

**Custom Domain:** - Domain Settings → Add domain - Gratis: yourname.netlify.app - Custom: Connect domain dari Cloudflare/Niagahoster

**Update Aplikasi:** - Drag folder baru → auto replace

**2. VERCEL** ⭐⭐⭐⭐

**Keuntungan:** - Support Next.js, React, Vue (kalau nanti upgrade) - Instant deployment - Analytics built-in - Edge functions (serverless)

**Cara Deploy:** 1. Buka [vercel.com](https://vercel.com) 2. Sign up dengan GitHub 3. New Project → Import repository (atau upload folder) 4. Deploy otomatis 5. URL: https://yourapp.vercel.app

**3. GITHUB PAGES** ⭐⭐⭐

**Keuntungan:** - Langsung terintegrasi dengan GitHub - Version control built-in - Gratis selamanya

**Cara Deploy:** 1. Buat repo GitHub baru: task-manager 2. Upload folder public/ 3. Settings → Pages → Source: main branch → folder: / (root) atau /public 4. Save → URL: https://username.github.io/task-manager/

**Bonus: Custom Domain di GitHub Pages** - Add CNAME file di root: yourname.com - Update DNS di domain provider

**Perbandingan Hosting:**

| Fitur | Netlify | Vercel | GitHub Pages |
| --- | --- | --- | --- |
| Deploy Speed | ⚡ 30s | ⚡ 30s | 🐢 2 menit |
| Bandwidth | 100 GB | 100 GB | 100 GB |
| SSL | ✅ Auto | ✅ Auto | ✅ Auto |
| Custom Domain | ✅ Gratis | ✅ Gratis | ✅ Gratis |
| Ease of Use | ⭐⭐⭐⭐⭐ | ⭐⭐⭐⭐ | ⭐⭐⭐ |

**Rekomendasi:** - **Pemula:** Netlify (paling mudah) - **Developer:** Vercel (lebih advanced) - **Open Source:** GitHub Pages (showcase project)

**Setup Homepage (index.html):**

Kebanyakan hosting otomatis baca index.html sebagai homepage.  
Kalau file Anda login.html, ada 2 opsi:

**Opsi 1: Rename**

login.html → index.html

**Opsi 2: Buat index.html Redirect**

<!DOCTYPE html>  
<html>  
<head>  
 <meta http-equiv="refresh" content="0; url=login.html">  
</head>  
<body>  
 <p>Redirecting to login...</p>  
</body>  
</html>

**Testing Checklist Setelah Deploy:**

* Buka URL di browser → halaman login muncul
* Test login dengan user dummy
* Test add task
* Test complete task
* Test delete task
* Test logout
* Buka di HP → responsive?
* Buka di browser lain (Chrome, Firefox, Safari)
* HTTPS aktif? (lock icon di address bar)

**Troubleshooting:**

**Error: “404 Not Found”** → Pastikan deploy dari folder public/, bukan root project

**Error: “Failed to fetch API”** → Cek API\_URL di file HTML sudah benar

**Error: “Mixed Content” (HTTP/HTTPS)** → Apps Script URL harus HTTPS (otomatis sudah HTTPS)

**Halaman blank** → Buka Developer Console (F12) → cek error JavaScript

**Exercise:** 1. Pilih 1 hosting (Netlify recommended) 2. Deploy aplikasi Task Manager Anda 3. Test semua fitur 4. Share URL ke 3 orang, minta feedback 5. Update berdasarkan feedback → deploy lagi

**Takeaway:** - Deploy itu mudah (5 menit, not 5 hours!) - Gratis selamanya (Netlify/Vercel/GitHub) - Update aplikasi = drag folder baru (not rocket science!)

### **BAB 12: MAINTENANCE: UPDATE TANPA CODING**

**Skenario Update Tanpa Sentuh Code:**

**1. Tambah User Baru** - Buka Google Sheets → Tab user - Tambah row baru: | john | pass123 | user | - Done! User john langsung bisa login

**2. Edit Data Task Manual** - Buka Google Sheets → Tab db01 - Edit cell langsung (ubah nama task, status, dll) - Refresh dashboard → update langsung terlihat

**3. Backup Data** - File → Download → Excel (.xlsx) - Simpan di Google Drive / Dropbox - Set reminder backup setiap bulan

**4. Restore Data (Rollback)** - Upload backup .xlsx ke Google Sheets - Ganti Sheets ID di Apps Script (kalau beda spreadsheet) - Deploy ulang

**5. Update Apps Script (Backend)** - Extensions → Apps Script - Edit code → Save - **WAJIB:** Deploy → Manage deployments → Edit → Version: New → Deploy - Frontend otomatis connect ke version baru

**6. Update Frontend (HTML)** - Edit file login.html atau dashboard.html lokal - Upload ulang ke Netlify (drag & drop) - Auto replace, URL tetap sama

**Maintenance Checklist (Bulanan):**

**✅ Data Health Check** - [ ] Cek ada row kosong/corrupt? - [ ] Cek ID duplicate? - [ ] Cek status typo? (harusnya done, malah Done atau selesai)

**✅ Performance Check** - [ ] Google Sheets < 5000 rows? (kalau > 5000, consider archive) - [ ] Apps Script quota usage < 50%? (lihat di dashboard) - [ ] Loading time dashboard < 2 detik?

**✅ Security Check** - [ ] Ganti password user default? - [ ] Apps Script masih “Who has access: Anyone”? - [ ] Tidak ada API\_URL terpublish di forum public?

**✅ Backup** - [ ] Download Sheets as .xlsx - [ ] Simpan di 2 tempat (Drive + lokal)

**Common Issues & Quick Fix:**

**Issue: “User lupa password”** **Fix:** Buka Sheets → tab user → lihat/reset password di kolom B

**Issue: “Data hilang/terhapus”** **Fix:** File → Version history → Restore versi sebelumnya (Google Sheets punya auto-backup!)

**Issue: “Task tidak bisa didelete”** **Fix:** Cek ID task di Sheets, pastikan match dengan yang di frontend

**Issue: “Overdue tidak update”** **Fix:** Overdue hanya cek saat load dashboard, tunggu auto-refresh (30s) atau refresh manual

**Issue: “Loading lambat”** **Fix 1:** Archive old data (pindah ke sheet archive)  
**Fix 2:** Add loading spinner di UI  
**Fix 3:** Pagination (load 50 task per page)

**Advanced: Auto Archive Old Data**

Apps Script (Jalankan Trigger 1x/bulan):

function autoArchive() {  
 const dbSheet = getSheet('db01');  
 const archiveSheet = getSheet('archive');  
   
 if (!archiveSheet) {  
 SpreadsheetApp.getActiveSpreadsheet().insertSheet('archive');  
 }  
   
 const data = dbSheet.getDataRange().getValues();  
 const now = new Date();  
   
 for (let i = data.length - 1; i >= 1; i--) {  
 const created = new Date(data[i][4]);  
 const daysDiff = (now - created) / (1000 \* 60 \* 60 \* 24);  
   
 if (daysDiff > 90 && data[i][3] === 'done') {  
 archiveSheet.appendRow(data[i]);  
 dbSheet.deleteRow(i + 1);  
 }  
 }  
}

Setup Trigger: 1. Apps Script → Triggers (icon jam) 2. Add Trigger → autoArchive → Time-driven → Month → Day 1 3. Save

**Exercise:** 1. Set reminder di calendar: “Maintenance Task Manager” (setiap bulan) 2. Praktek backup: Download Sheets as Excel 3. Praktek restore: Buka version history, restore 1 versi sebelumnya 4. Praktek update: Ganti judul dashboard “Task Manager” jadi “My Tasks”

**Takeaway:** - Maintenance tidak perlu coding skill - Google Sheets version history = lifesaver - Backup rutin = sleep well at night - 30 menit/bulan cukup untuk maintenance

### **BAB 13: DARI 1 APP KE 10 APPS DENGAN TEMPLATE YANG SAMA**

**Konsep Multi-App:**

Anda sudah punya **Task Manager** yang jalan.  
Dengan template yang sama, bisa bikin:

1. **Inventory System** (barang, stok, harga)
2. **CRM** (customer, status, deals)
3. **HR Absensi** (karyawan, tanggal, jam masuk/keluar)
4. **Finance Tracker** (transaksi, kategori, amount)
5. **Project Tracker** (project, milestone, progress)
6. **Lead Management** (lead, source, conversion)
7. **Equipment Booking** (alat, booker, tanggal)
8. **Complaint System** (complaint, PIC, status)
9. **Menu Ordering** (item, qty, table, status)
10. **Student Grades** (siswa, mata pelajaran, nilai)

**Cara Clone Template:**

**Step 1: Duplikat Folder**

task-manager/ (original)  
 ├── login.html  
 └── dashboard.html  
  
inventory-system/ (clone)  
 ├── login.html (sama persis)  
 └── dashboard.html (ubah UI + field)

**Step 2: Update Database Structure**

**Inventory System:** | id | itemName | category | stock | price | lastUpdated | |—-|———-|———-|——-|——-|————-|

**CRM:** | id | customerName | email | phone | status | dealValue | lastContact | |—-|————–|——-|——-|——–|———–|————-|

**HR Absensi:** | id | employeeName | date | checkIn | checkOut | totalHours | |—-|————–|——|———|———-|————|

**Step 3: Update Apps Script**

Ganti nama fungsi + field:

// Task Manager  
function addTask(itemName, username) { ... }  
  
// Inventory System  
function addItem(itemName, category, stock, price) { ... }  
  
// CRM  
function addCustomer(name, email, phone) { ... }

**Step 4: Update Frontend UI**

**Dashboard Inventory:**

<div class="add-form">  
 <input type="text" id="itemName" placeholder="Nama barang" required>  
 <input type="text" id="category" placeholder="Kategori" required>  
 <input type="number" id="stock" placeholder="Stok" required>  
 <input type="number" id="price" placeholder="Harga" required>  
 <button onclick="addItem()">Simpan</button>  
</div>

**Dashboard CRM:**

<div class="add-form">  
 <input type="text" id="customerName" placeholder="Nama customer" required>  
 <input type="email" id="email" placeholder="Email" required>  
 <input type="tel" id="phone" placeholder="Phone" required>  
 <select id="status">  
 <option>Lead</option>  
 <option>Prospect</option>  
 <option>Customer</option>  
 </select>  
 <button onclick="addCustomer()">Simpan</button>  
</div>

**Template Universal (Starter Kit):**

**1. File Structure**

app-name/  
├── public/  
│ ├── login.html  
│ ├── dashboard.html  
│ └── style.css (optional)  
├── SETUP.md  
└── README.md

**2. Google Sheets Template**

Sheet: user (sama untuk semua app)  
Sheet: db01 (customize per app)  
Sheet: logs (optional)

**3. Apps Script Template**

// ===== CONFIG =====  
const SHEET\_NAME = 'db01'; // ganti per app  
  
// ===== HELPERS ===== (sama semua app)  
function getSheet(name) { ... }  
function success(data) { ... }  
function error(message) { ... }  
function validateUser(username, password) { ... }  
  
// ===== CRUD ===== (customize per app)  
function getItems() { ... }  
function addItem(data) { ... }  
function updateItem(id, data) { ... }  
function deleteItem(id) { ... }  
  
// ===== MAIN ===== (sama semua app)  
function doPost(e) { ... }

**Rekomendasi Build Order:**

**Bulan 1:** Task Manager (belajar CRUD dasar)  
**Bulan 2:** Inventory System (tambah kategori + stok)  
**Bulan 3:** CRM (tambah status pipeline + email)  
**Bulan 4:** Finance Tracker (tambah sum/total calculation)  
**Bulan 5:** HR Absensi (tambah date range filter)

Setiap bulan: 1 app baru (30 menit build + 30 menit polish)

**Multi-Database di 1 Spreadsheet:**

Kalau mau 10 apps, tapi 1 Sheets:

Google Sheets: "Business Hub"  
 ├── Tab: user (shared login untuk semua app)  
 ├── Tab: tasks (Task Manager)  
 ├── Tab: inventory (Inventory System)  
 ├── Tab: customers (CRM)  
 ├── Tab: attendance (HR Absensi)  
 ├── Tab: transactions (Finance)  
 └── Tab: logs (activity tracking)

**Keuntungan:** - 1 Apps Script untuk semua (switch tab aja) - 1 login untuk semua app (seamless UX) - Relasi data mudah (contoh: Task → Customer)

**Code:**

function doPost(e) {  
 const params = JSON.parse(e.postData.contents);  
   
 switch(params.app) {  
 case 'tasks':  
 return handleTasks(params);  
 case 'inventory':  
 return handleInventory(params);  
 case 'crm':  
 return handleCRM(params);  
 default:  
 return error('Unknown app');  
 }  
}

**Exercise:** 1. Clone Task Manager → jadi Inventory System 2. Update database structure (4 kolom: item, category, stock, price) 3. Update Apps Script (fungsi addItem, getItems) 4. Update dashboard UI (form + table) 5. Deploy dan test

**Takeaway:** - 1 template → 10 apps (copy-paste + customize) - 30 menit per app baru (karena sudah punya template) - Multi-database di 1 Sheets = super efficient

### **BAB 14: MULTI-DATABASE: FINANCE, INVENTORY, HR DALAM SATU SISTEM**

**Konsep Hub System:**

**Google Sheets: “Business Hub”**

Tab 1: user (Username, Password, Role, Apps Access)  
Tab 2: tasks (Task Manager data)  
Tab 3: inventory (Barang, Stok, Harga)  
Tab 4: finance (Transaksi, Amount, Kategori)  
Tab 5: customers (CRM data)  
Tab 6: employees (HR data)  
Tab 7: attendance (Absensi data)  
Tab 8: logs (Activity tracking semua app)

**Frontend:**

https://yourapp.com/  
 ├── /login.html (shared login)  
 ├── /hub.html (pilih app)  
 ├── /tasks/ (Task Manager)  
 ├── /inventory/ (Inventory System)  
 ├── /finance/ (Finance Tracker)  
 ├── /crm/ (Customer Management)  
 └── /hr/ (HR & Absensi)

**Implementasi Hub System:**

**1. hub.html (App Selector)**

<div class="hub-container">  
 <h1>Business Hub</h1>  
 <div class="apps-grid">  
 <a href="tasks/dashboard.html" class="app-card">  
 <h3>📋 Task Manager</h3>  
 <p>Manage team tasks</p>  
 </a>  
 <a href="inventory/dashboard.html" class="app-card">  
 <h3>📦 Inventory</h3>  
 <p>Track stock & products</p>  
 </a>  
 <a href="finance/dashboard.html" class="app-card">  
 <h3>💰 Finance</h3>  
 <p>Income & expenses</p>  
 </a>  
 <a href="crm/dashboard.html" class="app-card">  
 <h3>👥 CRM</h3>  
 <p>Customer management</p>  
 </a>  
 </div>  
</div>

**2. Apps Script (Route per App)**

function doPost(e) {  
 try {  
 const params = JSON.parse(e.postData.contents);  
   
 if (!validateUser(params.username, params.password)) {  
 return error('Invalid credentials');  
 }  
   
 // Route based on app  
 switch(params.app) {  
 case 'tasks':  
 return handleTasks(params);  
   
 case 'inventory':  
 return handleInventory(params);  
   
 case 'finance':  
 return handleFinance(params);  
   
 case 'crm':  
 return handleCRM(params);  
   
 default:  
 return error('Unknown app');  
 }  
 } catch (err) {  
 return error(err.toString());  
 }  
}  
  
// Tasks Handler  
function handleTasks(params) {  
 switch(params.action) {  
 case 'getTasks':  
 return success(getTasks());  
 case 'addTask':  
 return success(addTask(params.itemName, params.username));  
 default:  
 return error('Unknown action');  
 }  
}  
  
// Inventory Handler  
function handleInventory(params) {  
 switch(params.action) {  
 case 'getItems':  
 return success(getInventoryItems());  
 case 'addItem':  
 return success(addInventoryItem(params.data));  
 case 'updateStock':  
 return success(updateStock(params.id, params.qty));  
 default:  
 return error('Unknown action');  
 }  
}  
  
// Finance Handler  
function handleFinance(params) {  
 switch(params.action) {  
 case 'getTransactions':  
 return success(getTransactions());  
 case 'addTransaction':  
 return success(addTransaction(params.data));  
 case 'getBalance':  
 return success(calculateBalance());  
 default:  
 return error('Unknown action');  
 }  
}

**3. Frontend Call (dengan app parameter)**

// Di tasks/dashboard.html  
const response = await fetch(API\_URL, {  
 method: 'POST',  
 body: JSON.stringify({  
 app: 'tasks', // identifier  
 action: 'getTasks',  
 username: sessionStorage.getItem('user'),  
 password: sessionStorage.getItem('pass')  
 })  
});  
  
// Di inventory/dashboard.html  
const response = await fetch(API\_URL, {  
 method: 'POST',  
 body: JSON.stringify({  
 app: 'inventory', // identifier  
 action: 'getItems',  
 username: sessionStorage.getItem('user'),  
 password: sessionStorage.getItem('pass')  
 })  
});

**Relasi Antar Database:**

**Contoh: Finance → Inventory (Auto Update Stok)**

**Skenario:** - User input transaksi “Jual Barang A, Qty: 5” - Finance tab: record transaksi - Inventory tab: kurangi stok Barang A otomatis

**Code:**

function addTransaction(data) {  
 const financeSheet = getSheet('finance');  
 const inventorySheet = getSheet('inventory');  
   
 // 1. Add transaction  
 financeSheet.appendRow([  
 generateId(),  
 data.type, // 'sale' or 'purchase'  
 data.item,  
 data.qty,  
 data.amount,  
 new Date().toISOString()  
 ]);  
   
 // 2. Update inventory stock  
 if (data.type === 'sale') {  
 updateStock(data.item, -data.qty); // kurangi stok  
 } else {  
 updateStock(data.item, data.qty); // tambah stok  
 }  
   
 return { success: true };  
}  
  
function updateStock(itemName, qtyChange) {  
 const sheet = getSheet('inventory');  
 const data = sheet.getDataRange().getValues();  
   
 for (let i = 1; i < data.length; i++) {  
 if (data[i][1] === itemName) {  
 const currentStock = data[i][3];  
 sheet.getRange(i + 1, 4).setValue(currentStock + qtyChange);  
 return true;  
 }  
 }  
 return false;  
}

**Advanced: Dashboard Analytics**

**Cross-App Stats:**

function getDashboardStats() {  
 const tasksCount = getSheet('tasks').getLastRow() - 1;  
 const inventoryValue = calculateInventoryValue();  
 const monthlyRevenue = calculateMonthlyRevenue();  
 const activeCustomers = getSheet('customers').getLastRow() - 1;  
   
 return {  
 tasks: tasksCount,  
 inventoryValue: inventoryValue,  
 revenue: monthlyRevenue,  
 customers: activeCustomers  
 };  
}  
  
function calculateInventoryValue() {  
 const sheet = getSheet('inventory');  
 const data = sheet.getRange('A2:D').getValues();  
   
 let total = 0;  
 data.forEach(row => {  
 if (row[0]) {  
 total += row[2] \* row[3]; // stock \* price  
 }  
 });  
   
 return total;  
}  
  
function calculateMonthlyRevenue() {  
 const sheet = getSheet('finance');  
 const data = sheet.getRange('A2:F').getValues();  
 const now = new Date();  
 const thisMonth = now.getMonth();  
   
 let revenue = 0;  
 data.forEach(row => {  
 if (row[0]) {  
 const date = new Date(row[5]);  
 if (date.getMonth() === thisMonth && row[1] === 'sale') {  
 revenue += row[4]; // amount  
 }  
 }  
 });  
   
 return revenue;  
}

**Frontend Dashboard:**

async function loadDashboard() {  
 const response = await fetch(API\_URL, {  
 method: 'POST',  
 body: JSON.stringify({  
 app: 'hub',  
 action: 'getStats',  
 username: sessionStorage.getItem('user'),  
 password: sessionStorage.getItem('pass')  
 })  
 });  
   
 const result = await response.json();  
   
 if (result.success) {  
 document.getElementById('tasksCount').textContent = result.data.tasks;  
 document.getElementById('inventoryValue').textContent = `$${result.data.inventoryValue}`;  
 document.getElementById('revenue').textContent = `$${result.data.revenue}`;  
 document.getElementById('customers').textContent = result.data.customers;  
 }  
}

**Access Control (Role-Based):**

**Tab: user** | username | password | role | appsAccess | |———-|———-|——|————| | admin | admin123 | admin | all | | staff1 | pass1 | staff | tasks,inventory | | finance | pass2 | staff | finance |

**Apps Script:**

function validateAccess(username, app) {  
 const sheet = getSheet('user');  
 const data = sheet.getDataRange().getValues();  
   
 for (let i = 1; i < data.length; i++) {  
 if (data[i][0] === username) {  
 const access = data[i][3];  
   
 if (access === 'all') return true;  
 if (access.includes(app)) return true;  
   
 return false;  
 }  
 }  
 return false;  
}  
  
function doPost(e) {  
 const params = JSON.parse(e.postData.contents);  
   
 if (!validateUser(params.username, params.password)) {  
 return error('Invalid credentials');  
 }  
   
 if (!validateAccess(params.username, params.app)) {  
 return error('Access denied to this app');  
 }  
   
 // ... rest of code  
}

**Exercise:** 1. Setup Hub System dengan 3 apps (Tasks, Inventory, Finance) 2. Implement relasi: Finance → Inventory (update stok) 3. Buat dashboard stats (cross-app analytics) 4. Test access control (buat user dengan limited access)

**Takeaway:** - 1 Sheets → 10 apps (multi-database) - Relasi antar tab mudah (same spreadsheet) - Hub system = professional look + efficient management

### **BAB 15: KAPAN WAKTUNYA HIRE PROGRAMMER (HINT: NANTI BANGET)**

**Red Flags: Waktunya Hire Programmer**

**1. Data > 10,000 Rows** - Google Sheets mulai lambat di > 10k rows - Solution sebelum hire: Archive old data, pagination - Kalau masih lambat: Consider Firebase/Supabase (masih bisa DIY)

**2. Concurrent Users > 100** - Apps Script quota: 20,000 calls/day - Kalau traffic tinggi, mulai butuh database real - Solution sebelum hire: Optimize query, caching

**3. Complex Business Logic** - Contoh: Algoritma pricing dinamis, ML prediction, payment gateway - Kalau logic > 500 lines, mulai ribet maintain sendiri

**4. Security Critical** - Payment processing, data sensitif (KTP, medical record) - Butuh audit, encryption, compliance - Jangan main-main, hire expert!

**5. Mobile App Native** - Kalau butuh iOS/Android app (bukan web app) - Butuh push notification, GPS, camera native - Consider hire atau pakai no-code (FlutterFlow, Adalo)

**Green Flags: Masih Bisa DIY**

✅ **User < 100, Data < 5000 rows**  
✅ **CRUD sederhana, tidak ada complex logic**  
✅ **Web app cukup (tidak butuh native mobile)**  
✅ **Tidak handle payment/data sensitif**  
✅ **Budget terbatas (< $500)**

**Kesimpulan: 90% bisnis kecil-menengah masih bisa DIY!**

**Alternative Sebelum Hire:**

**1. Upgrade Stack (Masih DIY)** - Dari Sheets → Firebase (gratis up to 50k reads/day) - Dari Apps Script → Supabase (free tier generous) - Dari HTML → Next.js (belajar 1 minggu)

**2. No-Code Tools** - Airtable (database visual) - Zapier/Make (automation) - Bubble.io (build app tanpa code)

**3. Hire Freelancer (1x Project)** - Budget: $100-500 untuk setup Firebase + refactor - Platform: Upwork, Fiverr, Sribulancer (Indonesia) - Hasil: Code clean + dokumentasi

**4. Outsource Maintenance** - Hire VA (Virtual Assistant) untuk data entry - Budget: $50/bulan - Mereka maintain Sheets, Anda fokus bisnis

**Kapan Invest ke Programmer Full-Time?**

**Kondisi Ideal:** - Revenue > $10,000/bulan - User > 500 active - Tech debt menghambat growth - Ada budget $3,000-5,000/bulan (gaji + tools)

**Sebelum hire:** - Pastikan product-market fit (user happy dengan versi simple) - Dokumentasi lengkap (apa yang sudah jalan, apa yang butuh improve) - Roadmap jelas (hire untuk apa? Scaling atau new features?)

**Cost Comparison:**

| Solusi | Cost | Maintenance | Scalability |
| --- | --- | --- | --- |
| Google Sheets + Apps Script | $0 | DIY (30 min/bulan) | < 100 users |
| Firebase/Supabase DIY | $0-25/bulan | DIY (1 hr/bulan) | < 1000 users |
| No-Code (Bubble, Airtable) | $25-100/bulan | Low | < 5000 users |
| Freelancer (1x setup) | $500 (1x) | DIY | < 1000 users |
| Hire Programmer | $3,000+/bulan | Handled | Unlimited |

**Real Case Study:**

**Bengkel Motor (Owner: Pak Budi)** - User: 3 mekanik + 1 admin - Data: 200 customer, 50 task/bulan - Stack: Google Sheets + Apps Script - Cost: $0 - Revenue: Naik 20% (karena tracking lebih baik) - **Kesimpulan:** No need hire programmer (yet!)

**Warung Makan (Owner: Bu Siti)** - User: 5 kasir + 1 owner - Data: 100 transaksi/hari - Stack: Google Sheets → Firebase (DIY upgrade) - Cost: $10/bulan (Firebase) - Revenue: Naik 30% (inventory real-time) - **Kesimpulan:** Upgrade stack, masih no hire!

**SaaS Startup (Owner: Team 3 orang)** - User: 500+ paying customers - Data: 50,000+ rows - Stack: Firebase → PostgreSQL + Node.js (hire freelancer) - Cost: $500 setup + $50/bulan hosting - Revenue: $15,000/bulan - **Kesimpulan:** Hire freelancer untuk refactor, worth it!

**Takeaway:** - Jangan buru-buru hire (optimize dulu yang ada) - Upgrade stack bertahap (Sheets → Firebase → Custom) - Hire saat revenue sudah support cost (bukan karena hype) - Fokus: User happy? Revenue naik? Kalau ya, stack saat ini cukup!

## 🎉 PENUTUP: YOUR NEXT STEPS

**Anda Sudah Belajar:** - ✅ Planning aplikasi dari nol (5 langkah) - ✅ Database design (Google Sheets) - ✅ Backend API (Apps Script) - ✅ Frontend (HTML + JavaScript) - ✅ CRUD (Create, Read, Update, Delete) - ✅ Deployment (Netlify/Vercel gratis) - ✅ Maintenance (update tanpa coding) - ✅ Multi-app system (10 apps, 1 template)

**Challenge 30 Hari:**

**Week 1:** Build Task Manager (follow Bab 9)  
**Week 2:** Build Inventory System (clone Task Manager)  
**Week 3:** Build Finance Tracker (tambah sum calculation)  
**Week 4:** Build CRM (integrasi ketiga app jadi Hub System)

**Hasil:** 4 apps production-ready, 0 framework, 0 biaya server!

**Resources:**

**Template Download:** - Task Manager Template (HTML + Apps Script) - Inventory Template - CRM Template - Hub System Template

**Community:** - Discord: [link] - Forum: [link] - GitHub: [link]

**Author Contact:** - Email: [email] - Twitter: [twitter] - LinkedIn: [linkedin]

**Closing Statement:**

“Aplikasi terbaik bukan yang paling canggih,  
tapi yang paling dipakai dan solve masalah real.

You don’t need to be a programmer.  
You just need to be a problem solver.

AI akan coding.  
Anda yang mikir.

Selamat membangun solusi digital Anda! 🚀”

**P.S.**

Kalau buku ini membantu, share ke teman yang juga punya masalah bisnis tapi takut coding.

Karena dunia butuh lebih banyak Builder, bukan hanya Coder.

**#BuilderMindset #NoCodeMovement #SolutionOverSyntax**

## 📚 BONUS: QUICK REFERENCE

**CRUD Cheat Sheet:**

// CREATE  
sheet.appendRow([id, name, value, timestamp]);  
  
// READ  
const data = sheet.getDataRange().getValues();  
  
// UPDATE  
sheet.getRange(row, col).setValue(newValue);  
  
// DELETE  
sheet.deleteRow(rowNumber);

**Fetch API Cheat Sheet:**

const response = await fetch(API\_URL, {  
 method: 'POST',  
 body: JSON.stringify({ action: 'getData', ...params })  
});  
const result = await response.json();

**Deployment Commands:**

# Netlify CLI  
npm install -g netlify-cli  
netlify deploy --prod  
  
# Surge  
npm install -g surge  
cd public && surge

**Google Sheets Shortcuts:** - Ctrl+S: Save - Ctrl+Z: Undo - Ctrl+Shift+V: Paste values only - View → Freeze → 1 row

**END OF DRAFT**

Total Pages: ~250-300 pages (estimate)  
Total Words: ~50,000 words  
Reading Time: ~8 hours  
Build Time (follow all exercises): ~30 hours

**Format:** PDF, ePub, Kindle  
**Price:** $19-29 (self-published)  
**Target:** Non-technical business owners, UMKM, aspiring entrepreneurs