# **Learning Path Praktis: Vanilla JS (Frontend) + Go (Backend) Tanpa Framework**

## **Stage 1 — Fundamental Setup (1–2 minggu)**

### **Frontend JS (Vanilla)**

1. Setup project sederhana (gunakan npm init + Vite atau cukup HTML + JS)
2. Buat struktur folder minimal:  
   * /src  
     + /components
     + /pages
     + /services
     + /utils
3. Buat halaman HTML dengan JS modular menggunakan ES Modules (import/export)
4. Implementasi interaksi sederhana, misalnya tombol tambah todo
5. Pahami penggunaan module scope dan ES6 import/export

### **Backend Go (Tanpa Framework)**

1. Setup workspace Go dengan go mod init
2. Buat struktur folder project:  
   * /cmd/app
   * /internal/handlers
   * /internal/services
   * /internal/repositories
   * /internal/models
3. Buat server HTTP sederhana dengan package net/http
4. Buat route /todos dengan method GET dan POST (CRUD minimal)
5. Simpan data sementara pakai map di memory (tanpa database dulu)
6. Pahami konsep handler function dan struct di Go

## **Stage 2 — Feature-Based Project (2–4 minggu)**

### **Frontend JS**

1. Buat folder /features/todo dan pindahkan logic todo ke sana
2. Pisahkan komponen UI, service (business logic), dan utils
3. Implementasi CRUD todo lengkap: add, edit, delete, mark done
4. Buat shared utilities (misal format tanggal, helper localStorage)
5. Gunakan event delegation dan DOM manipulation yang efisien

### **Backend Go**

1. Buat folder /internal/features/todo berisi handler, service, dan repository
2. Implementasi CRUD API todo lengkap (GET, POST, PUT, DELETE)
3. Pisahkan service logic dari HTTP handler
4. Gunakan interface untuk repository sebagai abstraksi database
5. Tambahkan validasi data input sederhana

## **Stage 3 — Architecture Mastery (4–6 minggu)**

### **Frontend JS**

1. Terapkan struktur folder clean dengan feature dan shared folder
2. Pisahkan logic state management sederhana menggunakan objek JS
3. Buat module untuk state management (contoh: todoState.js)
4. Implementasi observer pattern atau pub/sub sederhana untuk update UI
5. Refactor kode menjadi modul-modul kecil yang reusable

### **Backend Go**

1. Terapkan Clean Architecture:  
   * /cmd/app
   * /internal/domain (entity/model bisnis)
   * /internal/usecase (business logic)
   * /internal/repository (interface repository)
   * /internal/handler (HTTP API)
2. Pisahkan entity dan DTO (Data Transfer Object)
3. Implementasi interface repository dan mock untuk testing
4. Buat unit test sederhana untuk usecase dan repository
5. Tambahkan logging standar menggunakan package log

## **Stage 4 — Scaling Up (6–12 minggu)**

### **Frontend JS**

1. Implementasi lazy loading modul menggunakan dynamic import
2. Optimasi DOM manipulation dengan batching update
3. Implementasi caching data sederhana dengan localStorage/sessionStorage
4. Tambahkan form validation modular dan reusable

### **Backend Go**

1. Tambahkan koneksi database (Postgres/MySQL) dengan driver resmi (tanpa ORM)
2. Implementasikan connection pooling dan error handling yang baik
3. Tambahkan fitur pagination dan filtering di API
4. Tambahkan caching layer sederhana (misal Redis atau in-memory)
5. Dockerize aplikasi Go dengan membuat Dockerfile

## **Stage 5 — Production Ready**

### **Frontend JS**

1. Setup build tools seperti Rollup, esbuild, atau Vite (tanpa framework)
2. Minify dan bundling file JS dan CSS
3. Setup environment variables untuk konfigurasi (misal API URL)
4. Implementasi error handling UI (toast, modal error)
5. Setup basic unit test menggunakan Jest atau testing-library

### **Backend Go**

1. Setup configuration management dengan environment variables atau config files
2. Implementasi structured logging (misal pakai zap atau logrus jika mau)
3. Setup graceful shutdown server agar aplikasi berhenti dengan baik
4. Setup health check endpoint untuk monitoring
5. Setup monitoring metrics sederhana, misal expose Prometheus metrics

## **Catatan Penting**

* Fokus menggunakan vanilla JS dan Go murni tanpa framework atau ORM.
* Gunakan interface dan kode modular untuk loose coupling dan maintainability.
* Dokumentasikan struktur folder dan arsitektur di README tiap project.
* Gunakan Git dengan commit rutin setiap ada fitur atau perubahan besar.

Project

# **🎯 Fokus Pembelajaran Arsitektur Aplikasi dengan Project "TaskMaster"**

## **Pendekatan Utama:**

* Mulai dari **struktur folder dan layer aplikasi** (presentation, service, data, dll)
* Pahami **tanggung jawab setiap layer** dan bagaimana mereka berkomunikasi
* Terapkan **best practice pemisahan concern** (separation of concerns)
* Gunakan **pattern arsitektur sederhana dulu** (layered architecture, service pattern)
* Jangan hanya coding, tapi **refleksi setiap modul kenapa dan bagaimana** dibuat
* Sediakan **documentasi dan diagram sederhana** tiap tahapnya

## **Struktur Layer / Folder yang Akan Dipelajari**

text

SalinEdit

/backend

├── cmd/ # Entry point aplikasi (main.go)

├── internal/

│ ├── handlers/ # HTTP handler (controller)

│ ├── services/ # Business logic

│ ├── repositories/ # Data access layer (DB queries)

│ └── models/ # Entity/domain models

├── pkg/ # Utility/helper functions (logging, config)

└── migrations/ # DB migration scripts

/frontend

├── components/ # UI components (reusable)

├── services/ # API calls & state management

├── utils/ # Helpers & utilities

├── styles/ # CSS/animations

└── index.html / main.js

## **Step-by-step Pembelajaran Arsitektur**

| **Step** | **Fokus** | **Tujuan** | **Deliverable** |
| --- | --- | --- | --- |
| 1 | Definisi domain & model | Pahami konsep domain model dan entitas aplikasi | Sketsa model domain (Task, User, dll) |
| 2 | Repositori & data access | Abstraksi akses data (mock dulu, lalu DB) | Repository interface + mock implementation |
| 3 | Business service layer | Implementasikan logic bisnis (validasi, aturan, flow) | Service layer yang memanggil repo |
| 4 | HTTP handler (controller) | Terima request, validasi input, panggil service | Handler terpisah untuk API endpoint |
| 5 | API routing & middleware | Routing, error handling, middleware (logging, auth dasar) | Router sederhana dan middleware |
| 6 | Frontend modular structure | Pisah UI, service, utils, style | Struktur folder frontend, contoh komponen |
| 7 | Frontend service & state mgmt | Sinkronisasi data dari API dan state management | Service fetch + state simple observer/pubsub |
| 8 | Integrasi FE-BE & testing | Hubungkan frontend dan backend, buat unit test sederhana | Todo app sederhana full stack |

## **Tips belajar efektif:**

* Setelah buat kode, **tuliskan dokumentasi singkat**:  
    
    
   “Kenapa buat layer service? Apa fungsinya?”
* Gunakan diagram sederhana (draw.io / paper) untuk visualisasi
* Fokus pada **clean code** dan **pemahaman alur data**
* Review dan refactor, diskusikan flow setiap modul

Kalau kamu setuju, aku siap bantu bikin:

* Template folder dan file tiap layer
* Contoh kode minimal tiap layer dengan penjelasan
* Checklist tugas dan refleksi harian
* Diagram sederhana setiap minggu