SENIOR FRONTEND ENGINEER · ARCHITECTURE & SYSTEMS

YU CHENG, CHANG

React + TypeScript | Vue.js | Monorepo | Design System | Internal Tooling | CI/CD Automation | AI-Assisted Development

Da-Yeh University / Bachelor's Degree, Department of Information Management

E-MAIL	LINKEDIN	GITHUB
tabacotaco@gmail.com	https://linkedin.com/in/tabacotaco	https://github.com/taco3064

Experience

EXPERIENCE

2026/2 – 2026/9
8 months

Sr. Frontend Engineer

Cypher Lab Sdn. Bhd.

Frontend Engineer • Malaysia

- Built a tokenized Design System that centrally manages all colors, with a custom Build Plugin and Lint rules preventing direct use of hard-coded colors.
- Refactored large monolithic components into layered, independently testable modules, with tests co-located with their implementations.
- Built an AI-assisted development workflow with reusable Agent Skills, a Design-to-Code Validation process, and replayable data-collection workflows, enabling requirements to be turned into production features more efficiently.

Note: My employment ended after the company ceased operations.

#AI Agent Workflow #React.js #Vue.js #GitHub Actions #TypeScript

2025/5 – 2025/10
6 months

Sr. Frontend Engineer

BTSE

Frontend Engineer • Xinyi District, Taipei

- Developed an AST-based CLI tool to analyze router-level dependencies and trace the impact scope of code changes.
- Worked through change impact, test scope, and delivery uncertainty in a multi-brand system heavily dependent on cross-functional processes, gaining practical experience in responsibility boundaries and process transparency.

#Vue.js #AST #WebSocket #GitLab Runner #Jira

2023/6 – 2025/4
1 year 11 months

Sr. Frontend Engineer

Clinico

Frontend Engineer • Zhonghe District, New Taipei City

- Developed internal and external systems including CRM, BPM, and LIFF, reducing maintenance costs across features through reusable architectural patterns and consistent data flows.
- Refactored the BPM architecture by extracting shared form-layer logic, improving extensibility and reducing duplicate implementation.
- Consolidated multiple frontend services into an Nx Monorepo, unified shared modules and tooling, and applied SOLID design principles across products.
- Built a shared LIFF architecture layer that standardized authentication, routing, and UI behavior across all LIFF pages.
- Drove code-quality improvements by consolidating shared patterns, establishing consistency, and continuously reducing future maintenance costs through refactoring.

#ReactJS #TypeScript #GraphQL #Monorepo

2019/10 – 2023/4

3 years 7 months

Front-end Engineer

Gorilla Technology Group

Frontend Engineer • Neihu District, Taipei

- Upgraded an existing React project from v15 to v16, reorganizing shared state and components to reduce the impact of the migration.
- Built a config-driven no-code framework that dynamically generated React pages from structured configuration instead of relying on extensive hard-coding.
- Built and maintained frontend Shared Packages to centralize shared logic and ensure consistency across products.
- Introduced an Nx Monorepo to integrate code, types, and shared tooling, improving project scalability and team collaboration.
- Built GitHub Actions and GitLab Runner automation to standardize build, test, and release workflows.
- Established development standards, including folder structure and Git branch management workflows.

#ReactJS #TypeScript #Monorepo #GitLab Runner #GitHub Actions

2016/7 – 2019/10

3 years 4 months

Software Engineer

Walsin Lihwa

Software Engineer • Xinyi District, Taipei

- Developed internal ERP and MES systems with reusable UI patterns and consistent data flows.
- Planned and delivered frontend training, turning technical concepts into development patterns that were easier to understand and pass on.

#AngularJS #GitLab #Java / Spring #MS SQL #Bootstrap

2010/8 – 2016/6

5 years 11 months

Software Engineer

Evergreen International

Software Engineer • Zhongshan District, Taipei

- Built a configurable EDI system for field transformation, reducing custom development needs and improving long-term maintainability.
- Built an Ext JS UI/UX abstraction layer that standardized component behavior and development patterns, allowing engineers to focus on business logic while maintaining overall code consistency.

#Ext.js #Java / Spring #Oracle

Capabilities

CAPABILITIES

01	Frontend Architecture	Focused on building scalable, maintainable frontend architectures that can continue to evolve. Experienced in reducing system complexity through clear module boundaries, shared abstraction layers, and consistent design, with hands-on experience in architecture design, monorepos, shared libraries, dependency governance, and engineering standards.
02	Frontend System Design	Skilled at shaping complex business requirements into clear, testable, and maintainable frontend systems. Focused on state management, component architecture, Design Systems, performance optimization, and code structures built for sustainable evolution.
03	Engineering Productivity	Improving team development efficiency and maintenance quality through tooling, automation, and engineering standards. Hands-on experience building internal CLIs, ESLint rules, code-generation workflows, Git hooks, CI quality gates, and reusable processes.
04	AI-assisted Development	Treating AI as part of engineering collaboration rather than merely a code-generation tool. Built Agent Skills, prompt workflows, Design-to-Code Validation, and engineering guardrails while preserving engineers' final judgment over quality and architecture.
05	Technical Leadership	Helping teams establish consistent development practices through architecture design, engineering standards, technical documentation, and knowledge sharing. I believe a strong engineering culture comes from clear design principles and executable processes that enable teams to consistently deliver high-quality, maintainable software.

Philosophy PHILOSOPHY

My engineering philosophy has always pursued one thing:

> Looking back, I realized that what I have consistently pursued is a Single Source of Truth. SINGLE SOURCE OF TRUTH

A piece of data, a rule, and a module responsibility should each have a clear source. When the same thing is scattered across different places and consistency depends on human memory, complexity begins to accumulate.

Early in my career at Evergreen International, I turned recurring EDI transformations and interface behaviors into configurable, reusable capabilities. At Walsin Lihwa, I further organized my implementation approaches into UI patterns and training materials that the team could adopt. I did not yet have a name for this; I only knew that duplicated rules should not exist independently.

At Gorilla Technology Group, the problem expanded to multiple brands, Shared Packages, and different delivery workflows. I began using a config-driven framework, monorepo, dependency boundaries, and CI/CD so that code, versions, and teams followed the same structure. Folder Structure was no longer merely documentation, but an architectural rule that diagrams could explain and ESLint could enforce.

At Clinico, I saw more clearly that "sharing" does not mean putting everything together. It means identifying behaviors that are fundamentally the same and allowing them to exist only once, while preserving clear override boundaries for brand differences. Whether dealing with BPM, LIFF, multi-brand services, or data models between GraphQL and REST, I was addressing the same issue: preventing identical logic from gradually diverging as products evolve.

BTSE extended this principle from software architecture into collaboration. The impact scope of a multi-brand system could not be guessed, and cross-functional requirements, delivery, and testing responsibilities should not be interpreted separately. I therefore worked to calculate dependencies, provide QA with the affected pages, and align every role around the same set of facts.

At Cypher Lab Sdn. Bhd., this problem became even more direct once AI Agents entered the development workflow. AI can generate code quickly, but if architectural rules are scattered across people's memories, documents, and tools, it only amplifies inconsistency faster. I therefore consolidated layering, boundaries, linting, CI, documentation, and the Agent Contract into Blueprint, allowing both people and AI to understand the system from the same source.

01	02	03
Reduce duplicated sources of truth	Establish clear boundaries	Make the system's real rules understandable to both people and AI

Looking back, the architectural tools I built were not unrelated efforts. From configurable systems, shared patterns, monorepos, and impact analysis to Architecture as Code, I have consistently pursued the same thing: ensuring that the system has only one true answer—one that can be understood, verified, and continuously evolved.