# Mohamad Jad **Chaker**

**Computer Programmer** · Systems, low-level computing, and developer tooling

BARRIE & GREATER TORONTO AREA, ON · MJCHAKER19@GMAIL.COM · GITHUB.COM/MJCHAKER

## THE SHORT VERSION

I learned computing from the bottom up — memory layout, pointers, evaluators, the machine underneath the framework. That is why a new stack takes me days rather than months: modern frameworks are abstractions, and I already understand what they abstract over. Hand me React, Rust or a Kubernetes manifest and I am reading a new interface to problems I have solved a layer down.

I am not a prompt-and-pray developer — I write C by hand and I don't ship code I can't explain line by line. What I want from the work is leverage for other people: most of the people around me aren't short on intelligence, they're short on time and patience for the machine. Building the tool so they never have to think about it is the part of engineering I care about.

## SELECTED WORK

### Scheme Interpreter

C · FLAGSHIP

LANGUAGE IMPLEMENTATION, WRITTEN FROM SCRATCH

A working Lisp in hand-written C, in the SICP tradition: reader and tokenizer, an **eval/apply** core, and frame-chained lexical environments where closures capture their defining frame. Parsing, symbol interning, recursive evaluation and manual memory ownership, with no runtime to catch mistakes. Writing the evaluator made every other language's semantics legible to me.

### Notion MCP Server

AI AGENT TOOLING · MODEL CONTEXT PROTOCOL

A Model Context Protocol server exposing Notion to LLM agents as callable tools: protocol implementation, tool-schema design, third-party REST integration and auth. Built before Notion's official Claude integration existed.

### Universal Remote

SOFTWARE-DEFINED DEVICE CONTROL · APPLE FRAMEWORKS

A software remote built on Apple's native frameworks — device and command modelling, transport, and interface, specified in a full technical design document before any code.

### Original music, released commercially

SELF-PRODUCED · DISTRIBUTED VIA DISTROKID

Writing, producing, mixing and mastering original records, then shipping them to streaming platforms independently — where the signal-processing fluency at right was earned, and evidence I finish things nobody assigned me.

## EDUCATION

### Mathematical Physics — coursework toward BSc

UNIVERSITY OF WATERLOO · WATERLOO, ON

Honours mathematics coursework — analysis and multivariable calculus (MATH 137, MATH 237), linear algebra, formal proof. Degree not completed.

### BEng, Software Engineering

ONTARIO TECH UNIVERSITY · 2025 – PRESENT

Concurrent study — formal software engineering practice alongside the mathematics and physics background.

### Tutoring — mathematics & STEM

Ongoing informal tutoring. Same instinct as the tooling work: make the impenetrable usable by the person in front of me.

## CAPABILITIES

**LANGUAGES**

**C** (deep — K&R, Modern C, GNU internals) · **Python** · Java · Scheme / Lisp · Swift · C++ · zsh & shell scripting

**SYSTEMS & INFRASTRUCTURE**

Linux administration · Git · **Docker** · nginx · Ansible · OpenStack · architecture and systems programming at CS:APP depth

**AI & AGENT ENGINEERING**

**Model Context Protocol** servers · LLM API integration · tool-calling design · agentic workflow tooling

**FOUNDATIONS**

Automata & computability · machine learning (CS229) · data mining · distributed systems · MATLAB, R, LaTeX

**BUILDING TOWARD**

Modern web — JavaScript, the browser runtime, API-driven front ends · Solidity · Idris and dependent types

## DIGITAL SIGNAL PROCESSING

Working fluency, applied in production rather than studied once: **sample rate** and Nyquist limits, **bit depth**, quantisation and dither, **linear PCM**, **codecs versus container formats**, lossy/lossless trade-offs, and **non-linear processing** — compression, saturation, limiting — and why it resists clean closed-form description. Daily tools: **Logic Pro** and the full **FabFilter** suite.

## APPLE PLATFORMS

A hobby, held with clear eyes: **macOS**, **Xcode**, **Swift**, **XNU / Darwin** internals. Most of the world doesn't run Apple hardware, largely on price — so this is curiosity, not a bet. The transferable habit is reading the kernel and framework source instead of only the docs. Set up to build and **publish** — Xcode on Apple Silicon.

## HOW I WORK

> Every abstraction I use, I can open. That's the difference between someone who assembles software and someone you can hand an unfamiliar system to.