Adam Ryan McDaniel

Birth	May 14, 2002
Phone	(865) 724-6783
Mail	amcdan23@vols.utk.edu
Portfolio	https://adam-mcdaniel.net
GitHub	https://github.com/adam-mcdaniel/

EDUCATION

BS in Computer Science 2020-2022
Department of Electrical Engineering and Computer Science
University of Tennessee, Knoxville
Completed in 5 semesters
Graduated with a 3.95 GPA

MS in Computer Science 2022-2024
Department of Electrical Engineering and Computer Science
University of Tennessee, Knoxville
Graduated with a 3.90 GPA

PhD Computer Science Student 2024-Present
Department of Electrical Engineering and Computer Science
University of Tennessee, Knoxville
Maintaining a 3.93 GPA

SKILLS

- **Programming Languages:** Rust, C, C++, Python, Bash, JavaScript, Java, Lisp, Haskell, RISC-V
- **Tools:** Linux, macOS, CMake, Git, Jupyter, LaTeX, VSCode, Helix
- **Web Technologies:** React, Gatsby, Hugo, Flask
- **Spoken Languages:** Spanish (B2 - Upper-intermediate proficiency), English (Native)

WORK EXPERIENCE

- **Oak Ridge National Laboratory — PhD Research:** Developed a profiler to collect millisecond level power usage statistics for GPUs on exascale supercomputers for the purpose of attributing energy use to source code. Created patches for the AMD `rocm-smi` and `amd-smi` drivers to expose the energy counters from the driver to PAPI. Collaborated with HPE and

AMD to thoroughly characterize the energy usage of widely used GPU benchmarks on the MI250X and MI300A accelerators.

- **Oak Ridge National Laboratory — Graduate Research:** Developed a highly accurate process monitoring system for anomaly detection in automated systems. Achieved a program that detects anomalies without false negatives: the structure of the model prevents the possibility of misidentifying disturbances as normal conditions. Utilized auto-encoder machine learning models for detection and used LLMs to diagnose the flagged anomalies.
- **University of Tennessee — Graduate Research:** Created HeapPulse, a heap memory profiler. The profiler primarily tracks the compressibility, access patterns, lifetimes, and physical memory usage of heap allocations. Identified that most programs can compress up to 90% of their memory during runtime with little cost, as this memory is infrequently accessed and highly compressible.
- **Oak Ridge National Laboratory — Undergraduate Research:** Contributed to ASGarD (Adaptive Sparse Grid Discretization) project, a partial differential equation solver for exascale architectures. Project poster was featured at the Supercomputing Conference.
- **University of Tennessee — Software Developer Contractor:** Helped develop Simulated Electronic Fetal Monitoring app in JavaScript, and rewrote the application in Dart to run natively on mobile and desktop.

PUBLICATIONS

- **Mahmoud Jahanshahi, David Reid, Adam McDaniel, Audris Mockus**, "OSS License Identification at Scale: A Comprehensive Dataset Using World of Code." Published on arXiv, September 2024.
- **M. Graham Lopez, Adam McDaniel, Ed F. D'Azevedo, Wael Elwasif, Hao Lu, Lin Mu, David L. Green, Diego Del-Castillo-Negrete, B. Tyler McDaniel, Timothy R. Younkin**, "Implementing an Adaptive Sparse Grid Discretization (ASGarD) for High Dimensional Advection-Diffusion Problems on Exascale Architectures." Poster presented at the *Supercomputing Conference*, Denver, Colorado.

COURSEWORK

- **Spring 2025:** COSC 690 — *Graph Algorithms, Applications, and Implementations*, COSC 540 — *Advanced Software Engineering*
- **Fall 2024:** COSC 524 — *Natural Language Processing*, COSC 594 — *Intelligent Transportation Systems*
- **Spring 2024:** COSC 581 — *Algorithms*, COSC 594 — *Contemporary Topics in Compilers and Runtime Systems*
- **Fall 2023:** COSC 530 — *Computer Systems Organization*, COSC 562 — *Operating Systems Design and Implementation*
- **Summer 2023:** MATH 231 — *Differential Equations I*

- **Spring 2023:** COSC 525 — *Deep Learning*, COSC 527 — *Biologically-Inspired Computing*, COSC 534 — *Network Security*
- **Fall 2022:** COSC 522 — *Machine Learning*, COSC 583 — *Applied Cryptography*, COSC 445 — *Fundamentals of Digital Archaeology*, COSC 462 — *Parallel Programming*
- **Spring 2022:** COSC 566 — *Software Security*, COSC 402 — *Senior Design Practicum*, COSC 365 — *Programming Languages and Systems*
- **Fall 2021:** COSC 361 — *Operating Systems*, COSC 366 — *Introduction to Cybersecurity*, COSC 461 — *Introduction to Compilers*, COSC 401 — *Senior Design Theory*
- **Spring 2021:** COSC 360 — *Systems Programming*, COSC 340 — *Software Engineering*, COSC 312 — *Algorithm Analysis and Automata*
- **Fall 2020:** COSC 302 — *Data Structures and Algorithms II*, COSC 311 — *Discrete Structures*, ECE 313 — *Probability and Random Variables*
- **Spring 2020:** COSC 140 — *Data Structures and Algorithms I*, EF 152 — *Physics for Engineers II*, MATH 251 — *Matrix Algebra I*
- **Fall 2019:** COSC 130 — *Computer Organization*, ECE 201 — *Circuits I*, EF 151 — *Physics for Engineers I*, MATH 142 — *Calculus II*
- **Spring 2019:** COSC 102 — *Introduction to Computer Science*

NOTABLE PROJECTS

- **Sage Programming Language + Operating System:** Built a compiled language with generic algebraic datatypes and deployed the compiler in a web-playground. Used this language to write the userspace for my operating system.
- **Reckon:** A prolog-like theorem prover language. Used to implement a Hindley-Milner type inferencing algorithm.
- **Verus:** A simply-typed-lambda-calculus (STLC) based language designed as a better founded alternative to formally verified languages based on C, such as MISRA-C and CompCert.
- **Dune Shell:** Built a cross-platform shell with greater scripting support, syntax highlighting, and line completion. Daily driver for all work machines.
- **HeapPulse:** A memory profiler for tracking the compressibility and lifetimes of heap allocations.
- **Harbor:** Built a compiler for code obfuscation, low-resource environments, and reduced instruction sets.
- **Code Optimization with Genetic Algorithms:** Applied genetic algorithms to optimize compiler code generation. Reduced generated code size by as much as 20%.
- **Lite Text Editor:** Created a text editor with a built-in scripting language for users to add their own features and commands. Runs on all major operating systems.

- **Parser Combinator Library for Embedded Systems:** Constructed a parser combinator library that doesn't require an allocator. Used this library to implement parsers for various data formats and custom programming languages.
- **Wisp, Bootstrapped Lisp Compiler:** Created a Lisp interpreter, then created a compiler for the language using the language itself.
- **Transformer Model for Low Resource Language:** Created a transformer language model to communicate in a low resource language, Toki Pona. Constructed a public training dataset for other models to learn the language.
- **Machine Learning Gesture Detection:** Applied machine learning to create a gesture detection device with 3 light dependent resistors. Used this to implement a game controller.
- **VPN:** Created a VPN in C using the TUN/TAP interface and communicating over TLS.
- **Thompson NFA Regex Engine:** Implemented a Regex to NFA compiler. Used Thompson's algorithm to create a Regex engine invulnerable to pathological backtracking.
- **Big Integer Library in C:** Created a library for manipulating arbitrarily large integers in C. Used this to implement RSA encryption and decryption.
- **Rusty-CI, GitHub + GitLab Continuous Integration Tool:** Created a tool for managing continuous integration over GitHub and GitLab. Uses a YAML file to specify schedulers, workers, and tests for code projects. Utilizes passwords and authorized users to approve testing pull requests.
- **Web-Assembly Chess Engine:** Created a chess engine and a web interface for playing against the engine at various settings. Hosted on my website.
- **Personal Website, Blog, and Music Player:** Deployed several websites for hosting a portfolio, a blog, and a web-player for personally created music.