# Jason Yi

U.S Citizen | j.hyonyi@gmail.com | LinkedIn | GitHub | Website

## Education

**University of North Carolina at Chapel Hill** — Chapel Hill, NC
*Bachelor of Science in Computer Science, Statistics* — *August 2022 – May 2026*

- Intended **Master of Science in Computer Science**, August 2026 - May 2027
- **Coursework**: Operating Systems, Machine Learning, Algorithms, Databases, Stochastic Modeling, Probability
- **TA**: Algorithms (Spring 2025), System Fund. in C (Fall 2024), Data Structures in Java (Fall 2023, Spring 2024)

## Technical Skills

**Languages**: C/C++, Python, Java, Kotlin, TypeScript, JavaScript, HTML/CSS, Assembly, Swift/SwiftUI
**Frameworks/Libraries**: React.js, GraphQL, Angular, Node.js, PostgreSQL, NumPy, Pandas, Matplotlib, JUnit
**Developer Tools**: VSCode, Git, GitHub, Vim, Jira, Jenkins, Splunk, IntelliJ, Linux Kernel, AWS, XCode

## Experience

**Amazon Web Services** — May 2025 - August 2025
*Software Development Engineer Intern* — *Seattle, WA*

- Built a scalable mock data generation system for **AWS Compute Optimizer** in **Kotlin** with **AWS AppConfig**, **AWS Lambda**, and **S3** to reduce **Sev-2** (high-priority) alerts by decoupling canary tests from upstream data
- Accelerated bug bash validation by **20%** by integrating **TypeScript**-based workflows to automate ingestion of schema-validated mock data into existing indexing pipelines, which reduced operational delays
- Prototyped a **GenAI**-driven agentic workflow to automate bug bash processes by integrating mock data systems, schema validation, and test orchestration to streamline testing and accelerate validation timelines

**NSF RTG Networks | UNC Statistics Department** — September 2024 – December 2024
*Undergraduate Research Assistant* — *Chapel Hill, NC*

- Advised by Dr. Chudi Zhong to develop **Interpretable Machine Learning** algorithms/pipelines by refining models such as Decision Trees and Generalized Additive Models to ensure better decisions in high-stakes situations
- Optimized the TreeFARMS algorithm in **C++** and **Python**, reducing runtime by **20%** through parameter tuning and tree depth constraints, enabling faster enumeration of almost-optimal **Decision Trees**

**Fidelity Investments** — June 2024 - August 2024
*Software Engineer Intern* — *Durham, NC*

- Developed Backend services in **GraphQL** via **Experience API** for **Account Opening** which impacts **50+ million users**, and Frontend services in **Angular and TypeScript** for Crypto IRA
- Implemented customer info, address validation, and risk analysis services to prevent user fraud or illegal activity during account opening using **TypeScript and GraphQL** by matching data from multiple downstream APIs

## Projects

**CQLite** | *C, Ruby, RSpec, Bash*

- Built a persistent **B-Tree** database engine in **C** by modeling **SQLite's** internal structure, supporting **O(log n)** key lookup, in-order traversal across leaf pages, and dynamic splitting of internal and leaf nodes
- Implemented page-level memory management and cursor-based traversal, enabling range queries, recursive visualization, and structural correctness across **50+** randomized inserts
- Wrote **15+** integration tests in **RSpec** using pseudorandom insertions to validate structural integrity

**CC Compiler** | *C++, LLVM, GNU Bison, Flex*

- Built a custom toy programming language compiler in **C++** using **GNU Bison** for parsing and **Flex** for lexical analysis, generating an **Abstract Syntax Tree (AST)** and transforming it into **LLVM IR** code for execution
- Walked over the **AST** to generate byte/machine code for each node, and integrated **LLVM** to compile and execute the generated code, utilizing **llvm-config** for streamlined builds and testing

**AutoReturns** | *Python, PyTorch, Scikit-learn, NumPy, Pandas, Seaborn, Matplotlib*

- Analyzed **30M+** rows of U.S. stock return data (**CRSP, S&P 500, Russell 3000**) to document stylized facts (e.g., volatility clustering, fat tails) using PCA, autocorrelation, and return histograms
- Reduced dimensionality of stock return data by **90%** using kernel PCA in **scikit-learn** and autoencoders in **PyTorch**, preserving $\geq$**85%** variance and uncovering latent relationships across asset classes
- Clustered PCA and autoencoder embeddings with **KMeans** to identify sectoral and macroeconomic groupings