Lakshay Bansal

Albany, 12203 | +1 (518)-229-0731 | lbansal@albany.edu | LinkedIn

OBJECTIVE

Seeking an intern role where I can leverage expertise in back-end engineering, coding, and debugging to build impactful applications.

EDUCATION

State University of New York, Albany| Bachelor of Science in Computer Science **January 2024 - Expected May 2026**

Courses: DBMS, Data Structures and Algorithms, Computer Communication Networks, Software Engineering, Design and Analysis of Algorithms, Operating systems, System Fundamentals, Automata and Formal Languages, Assembly Programming

SKILLS

Languages	Python, Java, C++, JavaScript, TypeScript, SQL, PHP (Laravel), R, Bash
Frameworks & Libraries	Node.js, Express.js, React, Angular, Next.js, .NET, Spring Boot, Flask
Databases	MongoDB, MySQL, PostgreSQL, SQLite, Firebase, Redis
Testing & DevOps	JUnit, Pytest, Selenium, Mocha, Chai, Jenkins, GitHub Actions
Cloud & Deployment	Google Cloud Platform (GCP), AWS (EC2, S3, Lambda), Heroku, Docker, Kubernetes, Vercel
Data & ML Tools	NumPy, Pandas, Matplotlib, Scikit-Learn, TensorFlow, Keras,
Miscellaneous Tools	Git, Figma, Trello, VS Code, IntelliJ IDEA, Android Studio, REST APIs, GraphQL

PROJECTS

Java CPU & Cache Simulator GitHub

- Designed a cycle-accurate 32-bit RISC CPU simulator in Java with a complete fetch-decode-execute pipeline, including a custom bitlevel ALU (Adder, Shifter, Multiplier), and JUnit 5 tests for validating each instruction.
- Created a two-pass Assembler that translates assembly language to binary and an autoloader to set up memory, allowing end-to-end program execution with cycle-count reporting.
- Developed a two-level cache hierarchy (direct-mapped L1 instruction cache above unified L2) with 8-word blocks modeled and simulated hits/misses providing (~5× cycle savings for benchmark workloads).
- Created over 100 JUnit tests (for ALU, Memory, Cache, Processor, and Assembler) to test correctness of instruction executions, memory access and cache operations.

Tran Language Interpreter GitHub

- Made a high-performance interpreter pipeline (Lexer → Parser → AST → Interpreter) for a new Tran language that can run complex scripts in real time. It can recognize blocks, keywords, operators, literals, and comments based on indentation. It also uses SyntaxErrorException to keep track of line and column for accurate error reporting
- Implemented a TokenManager to facilitate lookahead and consumption of tokens for simple recursive descent parsing that turns token streams into composite AST nodes (classes, interfaces, methods, control structures, expressions).
- Implemented the various types of AST nodes (ClassNode, MethodHeaderNode, VariableDeclarationNode, StatementNode), which applies the Tran syntax, to support future semantic analysis and code generation.
- Included a full JUnit test suite that tests lexing edge cases, parsing ambiguities, and runtime exceptions to make sure the code is reliable enough for production.

Word-Guessing Game (C Web Server) GitHub

- Used a multi-threaded POSIX sockets and pthreads-based HTTP server to host a live word-guessing game at port 8000.
- Used a custom lexical/URL parser that retrieves and validates the player's guesses and imposes letter-frequencies constraints by probing dynamically letter distributions.
- Generated HTML/CSS responses-including word-progress display, letter inventory, and interactive forms-in real-time to provide a browser-based, professional game-play experience.
- Incorporated signal-handler code (SIGINT) and thorough memory-cleanup routines to enable graceful shutdown and prevent resource leaks.

CERTIFICATIONS

Algorithms – **Stanford University (Coursera),** Introduction to Artificial Intelligence – IBM, **Stanford University (Coursera)** – Introduction to Statistics, **IBM** – Introduction to Hardware and Operating Systems. **AWARDS & HONORS** **2nd Place Winner,** AI2EM Hackathon (Harnessing Artificial Intelligence to Manage Future Emergencies), University at Albany, April 2024.