Ali Alemami

Systems & Backend Developer

Amman, Jordan | ali.alemami01@gmail.com | +962 780 020 203 | github.com/ali-alemami | ali-alemami.github.io | linkedin.com/in/ali-alemami

Computer Science student (The Hashemite University '26 & 42 Amman) specializing in low-level systems and backend engineering. Strong foundation in **C/C++** (POSIX APIs, multithreading, network I/O) and **C#/.NET** (layered architecture, SQL Server). Seeking Backend / Systems Engineering Internships or Junior roles.

TECHNICAL SKILLS

Programming Languages:	C (POSIX/C99), C++ (98/11/17), C# (.NET), SQL (T-SQL), Bash
Systems & OS Concepts:	POSIX APIs, Multithreading (pthreads), Concurrency, Sockets & epoll, IPC, Dynamic Memory Management
CS Fundamentals:	Data Structures & Algorithms, OOP & Design Patterns, 3-Tier Architecture, Computer Networks
Tools & Infrastructure:	GDB, Valgrind, Docker, Docker Compose, NGINX, Git, Linux, Makefiles, SQL Server
Languages (Spoken):	Arabic (Native), English (Fluent / Professional)

EXPERIENCE

42 Amman — Software Engineering Cadet / Peer Developer 2025 – Present

Structured, deadline-driven software engineering program — 22/24 Common Core projects complete

- **Multithreading & Concurrency** (C — philo): Engineered a race-free simulation of Dijkstra's Dining Philosophers using pthreads and mutex locks; built async monitoring routines with microsecond-precision timers preventing thread starvation and deadlocks.
- **HTTP Web Server** (C++ — webserv): Building a non-blocking HTTP/1.1 server using epoll I/O multiplexing, with stateful request parsing, NGINX-style config parsing, and CGI execution.
- **DevOps & Container Infrastructure** (inception): Architected a multi-container infrastructure with Docker Compose on Alpine Linux; isolated containers for NGINX (TLS/SSL), WordPress, and MariaDB with bind mounts and internal bridge networking.
- **Unix Shell & AST Engine** (C — minishell): Implemented a POSIX-compliant shell with an AST parser enforcing operator precedence, nested subshell forks, and quote-aware tokenization; zero leaks via Valgrind.
- **Unix Process Pipelines & IPC** (C — pipex): Engineered a Unix pipeline engine replicating shell redirection (< file1 cmd1 | cmd2 > file2) using fork, pipe, dup2, and execve; implemented POSIX process synchronization, multi-fd stream redirection, PATH resolution, and error handling.
- **Algorithm Optimization & Dual-Stack Sorting** (C — push_swap): Built a dual-stack radix sort on array stacks with coordinate compression, bounding passes to log2(N); added lookahead bit-checking in Stack B to cut redundant transfers, with solvers for N ≤ 5.
- **C++ Modules & Ford-Johnson Algorithm** (C++ — cpps): Completed 42 C++ Modules 00–09 (OOP, polymorphism, templates, STL); implemented Ford-Johnson Merge-Insert Sort optimizing comparisons via Jacobsthal sequence pairing.
- **Foundations & Algorithms:** mini_rt (simple raytracer), fract-ol (fractal rendering), minitalk (UNIX signal IPC), get_next_line, ft_printf, libft.
- **Peer Review & Collaboration:** Conducted 50+ technical evaluations and collaborated in pair-programming teams (minishell, mini_rt).

SOFTWARE & DATABASE ARCHITECTURE PROJECTS

- **Bank & ATM Platform** (C++ — Bank-and-ATM-System-CPP): Built a modular terminal-based banking platform in C++ featuring role-based bitmask permissions, transaction audit logging, and persistent file storage.
- **Contacts Management Architecture** (C#, .NET, SQL Server — Contacts-App-3Tier): Engineered a layered N-tier architecture isolating business logic and data access; enforced data integrity with parameterized T-SQL queries and ACID transaction management.
- **Relational Database Schemas** (SQL Server — Programming-Advices-Core): Designed 5 normalized relational database schemas (3NF/BCNF) with indexed foreign keys, constraints, and optimized multi-table JOIN queries across complex business domains.
- **Algorithms & Problem Solving** (C++ — 225+ Problems): Implemented solutions across 5 difficulty levels covering data structures, matrix operations, string parsing, and combinatorial algorithms.
- **Event-Driven Desktop Applications** (C#, .NET — CSharp-WinForms-Collection): Developed a suite of 10+ desktop applications implementing event-driven architectures, custom UI controls, and object-oriented state management.

EDUCATION

The Hashemite University — BSc in Computer Science 2022 – 2026 (Expected)