Neh Patel

Email: nehpatel30062004@gmail.com | Phone: 315-529-6515 | GitHub: github.com/PatelNeh2004 | LinkedIn: linkedin.com/in/neh-patel

PROFESSIONAL SUMMARY

Computer Science undergraduate seeking **Software Engineering Internship** and **Backend Developer** roles with strong expertise in **systems programming, data structures, algorithms, and full-stack development**. Proficient in **API development, database design, test-driven development**, and **software architecture**. Experienced building **production-ready applications**, implementing **object-oriented design**, and optimizing performance in **Linux/Unix environments**. Skilled in **version control, CI/CD practices, debugging**, and **Agile methodologies**.

EDUCATION

State University of New York (SUNY) at Oswego Oswego, NY

Bachelor of Arts in Computer Science *Expected May 2027*

- **Relevant Coursework:** Data Structures and Algorithms, Systems Programming, Operating Systems, Graph Algorithms, Database Systems

TECHNICAL SKILLS

- **Programming Languages:** C, C++, Java, Python, JavaScript, SQL
- **Frameworks and Libraries:** React, Next.js, Node.js, Supabase, Prisma, PostgreSQL
- **Tools and Platforms:** Git, GitHub, Linux/Unix, REST API, Postman, Docker
- **Core Competencies:** Object-Oriented Programming (OOP), Data Structures, Algorithms, Memory Management, Systems Architecture, API Development, Database Design, Caching Strategies, Performance Optimization
- **Development Practices:** Test-Driven Development (TDD), Unit Testing, Debugging, Version Control, Agile/Scrum, CI/CD

RELEVANT EXPERIENCE

Study Buddy - AI-Powered Learning Platform *April 2025*

Software Developer *Next.js, Google Gemini API, Supabase, PostgreSQL*

Winner: Best AI Hack at LakerHacks 2025 (2nd Place Overall)

- Architected and deployed a full-stack web application that transforms unstructured notes into interactive flashcards and contextual question-answering workflows.
- Integrated Google Gemini 1.5 Flash REST APIs, automating content generation and reducing manual study preparation time by **60-70%**.
- Developed RESTful backend services using **Supabase (PostgreSQL)** and **Prisma ORM**, implementing secure authentication, role-based authorization, and persistent data storage.
- Collaborated in an Agile team to deliver a production-ready MVP within a **24-hour** deadline, demonstrating rapid prototyping and software development lifecycle management.

Systems Programming Projects *September 2025 - December 2025*

Software Engineer *C, Linux, POSIX APIs, Systems Programming*
CSC 322 - Systems Programming

- Designed and implemented a POSIX-compliant Unix shell supporting job control, background process execution, and signal handling (**SIGINT, SIGTSTP, SIGCHLD**) with **O(n)** job tracking complexity.
- Engineered a configurable CPU cache simulator supporting direct-mapped, set-associative, and fully associative architectures with **LRU and LFU** replacement policies and **O(1)** average access time.
- Delivered a scalable text-based game engine simulating up to **500 concurrent entities**, enforcing strict heap memory allocation and safety guarantees to prevent memory leaks.
- Optimized memory usage and eliminated race conditions, undefined behavior, and resource leaks using disciplined debugging techniques and performance profiling tools.

Wikipedia Semantic Search and Graph Analysis System *September 2025 - December 2025*
Backend Developer *Java, Graph Algorithms, JSoup, Data Structures*
CSC 365 - Data Structures and Algorithms

- Built a content-based recommendation engine processing **200+ Wikipedia documents** into weighted term-frequency models with **O(n)** text parsing efficiency per document.
- Implemented unsupervised machine learning algorithms (**k-means and k-medoids** clustering) to classify documents into **5-10** semantic clusters for improved search relevance.
- Modeled document relationships as a weighted graph with **400+ nodes**, implementing Dijkstra's shortest-path algorithm with **O(E log V)** time complexity.
- Improved query performance by **99%+** through precomputed similarity caching, reducing lookup latency from **O(n)** to **O(1)** at runtime.