Yusuf Lawal

Software & ML Engineer
Email | Github | Portfolio | Nigeria

Education

Osun State University, Nigeria **Sep. 2021 - Sep. 2025**
Bachelor of Science in Computer Science

Experience

Independent Projects **Dec 2024 — Present**
Software & ML Engineer

- Designed and developed a Node.js command-line tool that crawls a website; local or live and identifies broken links across its pages
- Built end-to-end solutions that encompass data preprocessing, model training, and deployment to production environments.
- Designed and integrated an n-gram language model trained on WikiText-103 datasets and engineered probabilistic fallback system calculating $P(w_3|w_1,w_2) = Count(w_1,w_2,w_3)/Count(w_1,w_2)$ across 500K+ pattern database, ensuring 100% prediction availability through 3-tier degradation strategy
- Developed an end-to-end facial surveillance and crowd analytics system. Leveraged YOLO for high-accuracy object detection and implemented SORT (Simple Online and Realtime Tracking) for robust temporal tracking. Built the deployment pipeline to handle video streams with minimal latency

IMTC, Osun State University **Dec 2024 — Aug 2025**
Software & ML Engineer

- Architected a QR-based automated attendance system to streamline data collection, which served as the foundation for a predictive model that forecasts student performance with 85% accuracy.
- Integrated these insights into a custom web application for educators and collaborated with senior engineers to scale these solutions across the university system.
- Extensive experience managing Linux-based environments, leveraging containerization (Docker) and virtualization to architect scalable, performance server infrastructures for model training and deployment

OIC HUB LIMITED **Jun 2023 — Nov 2024**
Software Engineer

- Develop full-stack web applications and intelligent systems, ranging from predictive engines to voice-controlled navigation.
- Designed and developed end-to-end web applications using the MERN stack (MongoDB, Express.js, React, Node.js), focusing on scalability, performance, and maintainability
- Gained strong working experience with the Linux operating system, including file organization, system configuration, debugging, and workflows

Work

Checker: A Node.js command-line tool that crawls a website; local or live and identifies broken links across its pages. You point it at a URL, and it systematically fetches each page, extracts every link, pings them for a valid response, and reports back what's broken, redirected, or dead all in real-time directly in your terminal. It's built for developers who want fast, scriptable link validation without opening a browser.

Approach - Built a Node.js CLI crawler from scratch, implementing BFS link traversal with concurrent HEAD/GET requests, robots.txt compliance, and binary asset detection to identify broken links across

local and live websites with full redirect and error classification, wrapped in a terminal interface with live output and structured JSON reporting.
Outcome - A fast, scriptable broken link checker that works on localhost before you push or as a CI/CD pipeline step to block deployments with dead links.
Source code - npm · github

Text-autocomplete: A next-word prediction interface built around an N-gram language model trained on WikiText-103 dataset. Trained trigram and bigram statistical models on WikiText-103 dataset (100M+ tokens), implementing Maximum Likelihood Estimation with conditional probability calculations. Engineered intelligent fallback chain ensuring graceful degradation from trigram (67% confidence) to bigram to common word suggestions. The goal was to understand how the autocomplete system works.
Approach - Trained N-gram language models on 100M+ Wikipedia tokens, implementing trigram/bigram statistical inference with Maximum Likelihood Estimation to predict next words with 35% top-3 accuracy and wrapped it in a lightweight interface that reacts as the user types.
Outcome - The result is a fast predictive typing interface that makes the underlying model legible, interactive, and easy to evaluate.
Source code - github

Arsenal-forecast: A probabilistic machine learning system that forecasts Arsenal's 2025/26 season outcomes across Premier League and UEFA Champions League using Monte Carlo simulation. Transform colloquial notions of "winning vs bottling" into rigorous, data-driven probabilities while accounting for fixture difficulty, current form, and multi-competition dynamics.
Approach - Built a gradient boosting classifier trained on 2022-2024 EPL data with features including team strength, rolling form, and home advantage. Ran 10,000 Monte Carlo simulations per competition to estimate title probabilities, then combined EPL and UCL forecasts for multi-trophy scenario analysis.
Outcome - Delivered 65% EPL title probability for Arsenal vs 24% for Man City, 32% UCL win probability, and 21% chance of winning the double. The system reveals that Arsenal can afford one slip-up while City needs near-perfection.
Source code - github

Crowd-density-analysis: A real-time computer vision system that monitors crowd density, detects bottlenecks, and identifies reverse flow patterns to prevent crowd disasters at events, stadiums, and public gatherings. The system uses a combination of object detection and tracking algorithms to analyze video feeds and provide actionable insights for crowd management and safety. Crowd risk builds quickly, so the system needed to surface meaningful movement signals fast enough to help with intervention.
Approach - Utilized OpenCV, YOLOV8; SUPERVISION for object detection and tracking to monitor flow patterns across live footage, then structured the outputs around safety-relevant events instead of raw detections.
Outcome - The project translates dense scene analysis into actionable signals for crowd management in stadiums, events, and public spaces.
Source code - github

Facial-surveillance: A real-time surveillance system that scans video/live RTSP footage to detect and track a specific person. Built with InsightFace for accurate face recognition. The system had to stay responsive while identifying a specific person accurately enough for real-world monitoring scenarios.
Approach - Built the pipeline around InsightFace recognition and real-time video processing so the target identity and tracking loop stay tightly connected.
Outcome - The result is a focused surveillance prototype that shows how face recognition can drive targeted tracking on live streams.

Source code - github

Technical skils

Languages: Python, JavaScript (TypeScript)
Technologies and Frameworks: Node.js, FastAPI, PyTorch, Scikit-learn, OpenCV, YOLO, Docker
AI Engineering: RAG, Model fine-tune, Prompt engineering, Guardrails, Adversarial testing, Hugging Face, Langchain