Abishek Maharajan - Detailed Resume

Abishek Maharajan

Full-Stack Developer | AI/ML Engineer | Cloud Infrastructure Specialist

👨‍💻 **Professional Identity:** Software developer with a passion for AI-driven solutions, full-stack web development, and modern cloud infrastructure. Demonstrated ability to rapidly learn and implement emerging technologies through 24+ weekend learning projects and professional experience in building production-ready AI applications.

Contact Information

Email: maharajanabishek@gmail.com

Phone: +91 7339377594

Portfolio: abishek-maharajan.online

GitHub: github.com/abishek-m

LinkedIn: linkedin.com/in/abishek-m

Location: OMR, Chennai, India

Professional Summary

Emerging AI/ML engineer and full-stack developer with hands-on experience building production-ready applications across multiple domains including HRMS/ATS systems, conversational AI, voice agents, and enterprise automation. Currently working at Clubits Solutions developing AI-powered modules for core HR systems.

Key Differentiators:

- **AI Integration Expertise:** Built multiple LLM-powered applications using OpenAI, Gemini, LangGraph, and local models (Llama.cpp, Ollama)
- **Full-Stack Proficiency:** MERN/PERN stack with TypeScript, Next.js, and modern frontend frameworks
- **Infrastructure Knowledge:** Hands-on experience with Docker, Kubernetes, ELK stack, vector databases, and cloud platforms
- **Rapid Learning:** Completed 24+ substantial weekend projects in 2025 covering AI agents, web scraping, data engineering, and DevOps
- **Self-Directed:** Fork-and-enhance approach to learning with consistent weekend development cycles

---

Professional Experience

Software Developer Intern | Clubits Solutions

January 2025 – Present | Chennai, India

Company Context: HR technology company providing comprehensive HRMS solutions to enterprises.

Key Projects & Responsibilities:

ATSv3 Module Development

- Architected and developed a complete Applicant Tracking System module integrated into the company's React-based HRMS platform
- Implemented **automated resume parsing** using AI to extract structured data from unstructured resumes (PDF, DOCX formats)
- Built **resume scoring engine** that matches candidate qualifications against job descriptions using semantic similarity algorithms
- Integrated with **Azure Communication Services** for automated email scheduling to candidates
- Designed interview calendar system with **Microsoft Teams meeting link generation**
- Implemented **resume-JD matching** using GROQ SDK for intelligent candidate ranking

- Set up data pipeline with **Azure Blob Storage** for resume storage and **MSSQL** for structured data
- Integrated third-party job board APIs (LinkedIn, Naukri) for job posting automation
- Developed IAM-based email workflow system for multi-user approval processes
- Created RAG/embedding pipeline for JD generation and intelligent ATS scoring

Tech Stack: React.js, Node.js, Azure OpenAI, Azure Communication Services, Azure Blob Storage, MSSQL, GROQ SDK, Python (for AI/ML components)

Conversational Medical Insurance Document Bot

- Engineered an AI-powered document analysis system for medical insurance policy documents
- Implemented **OCR pipeline** supporting both image and PDF inputs using Tesseract and EasyOCR
- Built **conversational interface** using Flask backend and Gemini API for natural language Q&A about policy details
- Added **voice support** for accessibility, allowing users to ask questions verbally
- Designed database schema in MSSQL to store conversation transcripts and parsed policy data as structured JSON
- Created evaluation framework to ensure accuracy of document interpretation

Tech Stack: Python, Flask, Google Gemini API, Tesseract, EasyOCR, MSSQL, Speech-to-Text APIs

AI Voice Agents for Dental Services

- Developed end-to-end **voice automation system** using ElevenLabs for dental clinic operations
- Built **appointment booking agent** that handles natural language conversation for scheduling
- Implemented **insurance verification agent** that calls patients and collects eligibility information

- Integrated with clinic's existing CRM/scheduling system for real-time calendar updates
- Designed conversation flows with human handoff for complex scenarios
- Achieved significant reduction in administrative workload for clinic staff

Tech Stack: ElevenLabs, Python, Node.js, CRM Integration APIs, Deepgram (STT)

OCR-Based Identity Verification System

- Built automated identity card data extraction system for HRMS onboarding workflows
- Implemented multi-format ID card processing (Aadhaar, PAN, Driver's License, Passport)
- Created preprocessing pipeline for image quality enhancement and alignment
- Developed validation logic to verify extracted data against government databases
- Reduced manual data entry time by 80% during employee onboarding

Tech Stack: Python, Tesseract, EasyOCR, OpenCV, PIL, MSSQL

Impact & Metrics:

- Reduced resume screening time by 70% through automated parsing and scoring
- Processed 500+ resumes in initial testing phase with 92% accuracy in data extraction
- Automated 100% of routine appointment booking calls for pilot dental clinic
- Decreased onboarding data entry errors by 85% with OCR verification system

---

AI Developer Intern | Street League Commercials Pvt Ltd

April 2024 – June 2024 | Abu Dhabi (Remote)

Company Context: Digital marketing and e-commerce solutions provider based in UAE.

Key Projects & Responsibilities:

AI Image Generation Application

- Designed and built a full-stack web application integrating **ChatGPT-4 Vision** and **DALL·E 3** APIs
- Developed **React.js frontend** with modern UI/UX featuring real-time image generation preview
- Built **Node.js/Express backend** with efficient API request handling and rate limiting
- Implemented **conversation history** system storing user prompts and generated images
- Created image gallery with filtering, search, and export capabilities
- Optimized API usage to minimize costs while maintaining responsiveness
- Added user authentication and session management for multi-user support

Tech Stack: React.js, Node.js, Express.js, OpenAI API (GPT-4 Vision, DALL·E 3), MongoDB, JWT Authentication

Learning Outcomes:

- Deep understanding of OpenAI API integration patterns and best practices
- Experience with real-time streaming responses and async data handling
- Frontend state management for complex AI-powered applications
- Cost optimization strategies for production AI applications

---

Full-Stack Developer | Cognizant – CAFE Programme

February 2023 – June 2023 | Chennai (College - Remote)

Program Context: College outreach program providing hands-on training in enterprise software development.

Key Projects & Responsibilities:

Single-Page Application Development

- Developed production-grade **Angular SPA** with dynamic component loading and lazy loading

- Implemented **Angular routing** with guards and resolvers for protected routes
- Built **reactive forms** with complex validation logic and custom validators
- Created **shared component library** for consistent UI across application modules
- Integrated with RESTful backend APIs using Angular HttpClient and RxJS observables
- Implemented **state management** using NgRx for complex application state

Security Implementation

- Utilized **SonarByte** (now SonarQube) to generate comprehensive security reports
- Generated **SAST (Static Application Security Testing)** reports identifying code vulnerabilities
- Performed **DAST (Dynamic Application Security Testing)** to find runtime security issues
- Remediated critical and high-severity security findings in codebase
- Improved security score from 62% to 94% during the program
- Implemented security best practices including input sanitization, CSRF protection, and secure storage

Tech Stack: Angular, TypeScript, RxJS, NgRx, SonarQube, RESTful APIs

Skills Developed:

- Enterprise Angular development patterns and architecture
- Security-first development mindset and SAST/DAST tooling
- Professional code review and quality assurance processes

Featured Personal Projects

Production-Grade Projects

1. Conversational Insurance Bot (2025)

Problem Statement: Insurance agents spend hours answering repetitive questions about policy details from complex documents.

Solution Approach:

- Built intelligent document processing pipeline with OCR for image/PDF inputs
- Implemented RAG (Retrieval-Augmented Generation) pattern for accurate, grounded responses
- Created conversational interface with voice support for accessibility
- Stored structured data for analytics and audit trails

Technical Deep Dive:

- **OCR Layer:** Used Tesseract for typed text and EasyOCR for handwritten text, with preprocessing for image quality enhancement
- **Document Processing:** Chunked documents intelligently based on semantic boundaries for better retrieval
- **Vector Storage:** Embedded document chunks using sentence-transformers for semantic search
- **LLM Integration:** Used Gemini API with custom system prompts to ensure responses are grounded in document content
- **Voice Integration:** Implemented STT/TTS for hands-free operation
- **Database Design:** MSSQL schema storing documents, conversations, and extracted policy entities

Tech Stack: Python, Flask, Google Gemini API, Tesseract, EasyOCR, sentence-transformers, ChromaDB, MSSQL

Outcome: Successfully processed 50+ insurance policy documents with 95% accuracy in Q&A testing.

---

2. Sybil Attack Detection Model (2024)

Problem Statement: Social networks vulnerable to Sybil attacks where malicious actors create multiple fake identities.

Solution Approach:

- Built deep learning model using CNN and transformer architectures

- Implemented feature engineering pipeline extracting graph-based and behavioral features
- Optimized model performance using AdamW optimizer with learning rate scheduling
- Achieved 96% accuracy on test dataset

Technical Deep Dive:

- **Feature Engineering:**
  - Graph features: degree centrality, clustering coefficient, betweenness
  - Temporal features: account age, activity patterns, posting frequency
  - Content features: text embeddings from posts using BERT
  - Network features: friend-to-friend connection patterns
- **Model Architecture:**
  - Transformer encoder for sequential behavioral patterns
  - CNN layers for spatial feature extraction from network graphs
  - Attention mechanism to identify critical detection signals
- **Training Pipeline:**
  - Dataset: 50,000 labeled accounts (balanced)
  - AdamW optimizer with cosine annealing learning rate
  - Early stopping based on validation F1 score
  - Data augmentation using SMOTE for rare attack patterns

Tech Stack: Python, PyTorch, Transformers (Hugging Face), NetworkX, Pandas, Scikit-learn

Metrics:

- Accuracy: 96%
- Precision: 94%
- Recall: 93%
- F1 Score: 93.5%
- Inference time: <50ms per account

Research Impact: Demonstrated effectiveness of hybrid CNN-Transformer architecture for graph-based security applications.

---

3. Maaxly - Student-Employer Opportunity Platform (2025)

Problem Statement: Students lack accessible platforms to connect with employers through meaningful competitions and job opportunities, while employers struggle to find and engage with qualified student talent.

Solution Approach:

- Built comprehensive platform connecting students, employers, and educational institutions
- Implemented event-driven architecture for real-time updates and scalable messaging
- Designed mobile-first, security-first architecture with comprehensive testing framework
- Conducted detailed multi-cloud deployment planning with cost optimization

Technical Deep Dive:

Architecture Evolution:

- **Initial Prototype (Early 2025):** Next.js 14 with App Router, Supabase backend, TypeScript
  - Mobile-first development with progressive enhancement
  - Comprehensive RLS (Row Level Security) policies
  - Multi-provider authentication (Clerk + Firebase)
  - Shadcn/ui component library
- **Production Stack (Mid-August 2025):** Complete architectural redesign
  - **Frontend:** React 18, Vite for optimal build performance
  - **Backend:** Node.js with Express.js RESTful APIs
  - **Database:** MongoDB for flexible schema and scalability
  - **Message Queue:** Apache Kafka (3 partitions, 5 brokers) for event-driven architecture

- **Caching:** Redis (150MB allocation) for session management and real-time data
- **Containerization:** Docker with Docker Compose for local development
- **Cloud Strategy:** Evaluated AWS, GCP, Azure with detailed cost analysis

Core Features Implemented:

- **Multi-Role System:** Student, Employer, Admin dashboards with role-based access control
- **Competition Management:**
  - Competition creation, editing, and lifecycle management
  - Real-time registration and application tracking
  - Category-based filtering and advanced search
- **Job Board:**
  - Job posting with rich descriptions
  - Application tracking system
  - Company profile integration
  - Location and type-based filtering
- **Messaging System:**
  - Real-time conversations between students and employers
  - Kafka-powered message delivery
  - Read receipts and typing indicators
- **Admin Panel:**
  - User management and content moderation
  - Platform analytics with engagement metrics
  - Audit trail for compliance

Event-Driven Architecture with Kafka:

- **Partition Strategy:** 2 partitions for live updates, 1 for backup/analytics
- **Use Cases:**
  - Real-time notifications for applications and messages

    - Event sourcing for audit trails
    - Analytics event processing
    - Asynchronous job processing

Testing & Quality Assurance:

- **Comprehensive Test Suite:** 60+ test cases across all system areas
- **Test Categories:**
    - **Integration Tests:** Backend API endpoints, Kafka message flow, database operations
    - **E2E Tests:** User workflows from registration to application submission
    - **Unit Tests:** Individual component and function testing
    - **Performance Tests:** Load testing for concurrent users
    - **Security Tests:** Input validation, authentication flows, authorization checks
- **Test Coverage Areas:**
    - Frontend (UI/UX, responsive design, accessibility)
    - Backend (API endpoints, business logic, data validation)
    - Kafka (message delivery, partition handling, consumer groups)
    - Redis (caching strategies, session management)
    - Search (Elasticsearch integration for full-text search)
    - Profiles (student and employer profile management)
    - Messaging (real-time conversations, notifications)
    - Auth (multi-factor authentication, OAuth flows)
    - DevOps (Docker builds, deployment pipelines)

Deployment Planning & Cost Optimization:

- **Multi-Cloud Analysis:** Evaluated 9 different deployment scenarios
- **Plan 1 - Managed Services:** Full managed Kafka, Redis, MongoDB
    - AWS: $1,729/month (MSK premium)
    - Azure: $1,172/month (most cost-effective for managed)

    - GCP: $1,446/month (Confluent Cloud dependency)
- **Plan 2 - Self-Hosted:** Docker Compose on VMs
    - GCP: $252/month (lowest compute cost)
    - AWS: $299/month
    - Azure: $337/month (highest for self-hosted)
- **Plan 3 - Hybrid Containerized (Selected):** VM + Managed Containers
    - **AWS: $231/month (optimal choice)** - Fargate for containers
    - Azure: $267/month - Azure Container Instances
    - GCP: $301/month - Cloud Run (less optimal for persistent workloads)
- **Cost Breakdown (AWS Plan 3):**
    - Application VM (4 vCPU, 16GB): $121.47
    - Fargate containers (Kafka/Redis): $85.06
    - MongoDB Atlas M5: $25.00
    - S3 Storage (10GB): $0.23
- **Resource Allocation:**
    - 4 vCPU, 8GB RAM for application server
    - 2 vCPU, 8GB RAM for Dockerized services
    - 5GB MongoDB storage for user data
    - 10GB blob storage for resumes and intro videos

Security Implementation:

- **Zero Trust Architecture:** Never trust, always verify approach
- **Defense in Depth:** Multiple security layers
- **Input Validation:** Client and server-side with comprehensive sanitization
- **Authentication:** JWT-based with secure session management
- **Authorization:** Role-based access control at API and database levels
- **Data Protection:** Encrypted at rest and in transit

Performance Optimizations:

- **Redis Caching:** Sub-millisecond response times for frequently accessed data
- **Lazy Loading:** Component-level code splitting for faster initial load
- **Image Optimization:** Compressed resumes and profile images
- **Database Indexing:** Optimized MongoDB queries with proper indexes
- **CDN Strategy:** Static asset delivery planned

Project Challenges & Solutions:

- **Challenge 1 - Event-Driven Complexity:** Managing message ordering and exactly-once delivery
  - **Solution:** Implemented Kafka consumer groups with idempotency keys
- **Challenge 2 - Real-time Synchronization:** Keeping UI updated across multiple users
  - **Solution:** WebSocket connections with Redis pub/sub for instant updates
- **Challenge 3 - Cost Management:** Balancing managed services vs operational overhead
  - **Solution:** Detailed 9-scenario analysis leading to optimal hybrid approach
- **Challenge 4 - Type Safety at Scale:** Maintaining consistency across large TypeScript codebase
  - **Solution:** Comprehensive Zod schemas, ESLint rules, and code review processes

Tech Stack:

- **Frontend:** React 18, Vite, TypeScript, Tailwind CSS, Shadcn/ui, React Hook Form, Zustand
- **Backend:** Node.js, Express.js, JWT authentication, RESTful APIs
- **Database:** MongoDB (primary data), Redis (cache, sessions)
- **Message Queue:** Apache Kafka (event streaming)
- **DevOps:** Docker, Docker Compose, GitHub Actions (CI/CD planned)
- **Cloud:** AWS (primary target), multi-cloud deployment capability

- **Testing:** Jest, React Testing Library, Cypress/Playwright (E2E)

Current Status:

- **MVP Complete:** All core features implemented and tested
- **Deployment Ready:** Infrastructure planned and costed
- **Resource Allocation:** Server sizing and budget finalized
- **Next Steps:** Production deployment, monitoring setup, user beta testing

Project Metrics:

- **Development Time:** 8+ months (including prototype iterations)
- **Test Cases:** 60+ comprehensive tests across all areas
- **Code Quality:** ESLint compliant, TypeScript strict mode
- **Architecture Documents:** Complete deployment plans with cost analysis
- **Deployment Cost:** $231/month (AWS hybrid containerized approach)

Business Impact (Projected):

- Reduces student-employer connection friction by 70%
- Provides structured competition framework for skill demonstration
- Enables data-driven hiring decisions through analytics
- Scalable to 10,000+ concurrent users with current architecture

---

Weekend Learning Projects (2025)

> 🎯 **Learning Philosophy:** Dedicated weekends to hands-on technical exploration through the "fork & enhance" methodology. Each project represents not just code written, but technologies mastered and problems solved.

AI & Machine Learning Projects

1. Agents From Scratch - LangGraph Email Assistant

Objective: Master AI agent architecture by building a production-ready ambient email assistant.

Project Scope:

- Implemented complete email triage and management system using LangGraph for workflow orchestration
- Built human-in-the-loop approval system with Agent Inbox for sensitive operations
- Created memory system using LangGraph Store enabling agent to learn from user feedback
- Developed comprehensive evaluation framework using LLM-as-a-judge metrics
- Integrated with Gmail API for production deployment

Technical Achievements:

- **Agent Architecture:** Multi-step reasoning with tool calling (email read, send, archive, label)
- **Memory Implementation:** Vector-based retrieval of past user preferences and decisions
- **Evaluation Pipeline:** Automated testing using LangSmith with custom metrics
- **Background Processing:** Ambient agent pattern running periodic inbox scans

Key Learnings:

- LangGraph state management and workflow orchestration
- Production AI deployment patterns and monitoring
- Human-in-the-loop design for safe autonomous agents
- Prompt engineering for reliable tool use
- Agent evaluation methodologies and metrics

Tech Stack: Python, LangGraph, LangSmith, OpenAI API, Gmail API, Pytest, ChromaDB

Project Outcome: Functional ambient agent capable of managing 80% of routine email tasks with human approval for critical actions.

---

2. Agentic OS v0 - Local LLM Infrastructure

Objective: Build completely offline AI system for privacy-sensitive environments.

Project Scope:

- Compiled and optimized Llama.cpp for local CPU inference
- Implemented GGUF model loading and management system
- Created command-line interface for AI-powered system operations
- Optimized quantization strategies (4-bit, 5-bit, 8-bit) for different hardware

Technical Achievements:

- **Build Process:** Successfully compiled Llama.cpp with CPU optimizations (AVX2, OpenBLAS)
- **Model Management:** Created abstraction layer for loading different GGUF models (TinyLlama 1.1B, Llama 2 7B)
- **Performance:** Achieved 25 tokens/sec on consumer CPU with 4-bit quantization
- **Memory Efficiency:** Implemented context caching and KV cache optimization

Key Learnings:

- Low-level LLM inference optimization
- C++ compilation and cross-platform builds (CMake)
- Quantization techniques and accuracy/speed tradeoffs
- Memory management for large model inference
- CPU optimization techniques (SIMD, cache-friendly algorithms)

Tech Stack: Python, Llama.cpp, C++, CMake, GGUF format, TinyLlama/Llama 2 models

Project Outcome: Fully functional local AI system running on commodity hardware with no internet dependency.

---

3. Voice Agents with OpenAI - AIE Workshop Implementation

Objective: Implement production-ready voice-enabled AI agents using OpenAI's latest SDK.

Project Scope:

- Built progression from basic text agents to sophisticated voice interactions
- Implemented real-time voice streaming using WebRTC
- Created Next.js frontend with modern UI for voice interactions
- Added type-safe implementation with Zod schemas

Technical Achievements:

- **Voice Processing:** Real-time bidirectional audio streaming with <300ms latency
- **Voice Activity Detection:** Custom VAD algorithm to detect speech start/end
- **Conversation Flow:** State machine for managing multi-turn voice dialogues
- **Error Handling:** Robust fallback strategies for audio processing failures

Key Learnings:

- OpenAI Realtime API for voice interactions
- WebRTC for browser-based audio streaming
- Audio processing in JavaScript (Web Audio API)
- Real-time state synchronization between client and server
- UX patterns for voice interfaces

Tech Stack: Node.js, TypeScript, OpenAI Agents SDK, Next.js, WebRTC, Zod, Web Audio API

Project Outcome: Voice agent handling natural conversations with 95% transcription accuracy and natural-sounding responses.

---

4. YouTube Video Summarizer

Objective: Build intelligent video content processing system with dynamic topic extraction.

Project Scope:

- Extracted transcripts from YouTube videos using official API
- Implemented intelligent content segmentation based on topic boundaries
- Built NLP processing pipeline with topic modeling

- Generated structured summaries with timestamps
- Created batch processing capability for channel analysis

Technical Achievements:

- **Topic Segmentation:** Implemented TextTiling algorithm for finding topic boundaries
- **Keyword Extraction:** TF-IDF with custom domain-specific stopwords
- **Summarization:** Extractive summarization using sentence scoring
- **Quality Validation:** Automated coherence checking of generated summaries

Key Learnings:

- NLP techniques: tokenization, stemming, topic modeling
- Text segmentation algorithms
- Extractive vs abstractive summarization
- YouTube API integration and rate limiting
- Batch processing patterns for large-scale content

Tech Stack: Python, YouTube Transcript API, BeautifulSoup4, NLTK, Scikit-learn, Selenium, Playwright

Project Outcome: Processed 100+ videos across multiple channels with average summary quality score of 4.2/5.

---

5. AI Model PowerShell Integration

Objective: Automate Windows system administration using AI-powered scripts.

Technical Achievements:

- Built PowerShell modules for AI API integration
- Implemented system automation workflows (user management, log analysis, backup automation)
- Created conversation memory using MongoDB

Tech Stack: PowerShell, REST APIs, JSON, MongoDB

---

6. DOCS Scraper - Vector Database Integration

Objective: Create local AI-powered documentation search system.

Technical Achievements:

- Scraped technical documentation and indexed in vector databases
- Implemented semantic search using embeddings
- Built RAG pipeline for question answering

Tech Stack: Python, ChromaDB, Qdrant, Ollama, sentence-transformers

---

Web Development Projects

7. Project Management System - MERN Stack

Objective: Build production-grade task management application.

Project Scope:

- Complete CRUD operations for projects, tasks, and users
- Real-time updates using WebSocket connections
- User authentication and role-based access control
- Dashboard with analytics and reporting

Technical Achievements:

- **Backend API:** RESTful Express.js with 15+ endpoints
- **Database Design:** Normalized MongoDB schema with efficient indexing
- **Real-time:** Socket.io integration for live task updates
- **Frontend:** React with Context API for state management
- **Responsive Design:** Mobile-first approach with breakpoints

Tech Stack: MongoDB, Express.js, React, Node.js, Socket.io, JWT, Font Awesome

---

8. NextJS Portfolio Website

Objective: Build personal portfolio with optimal performance and SEO.

Project Scope:

- Static site generation for fast loading

- Shadcn/ui component library for consistent design
- Automated CI/CD deployment to GitHub Pages
- Lighthouse score: 98+ across all metrics

Technical Achievements:

- **Performance:** First Contentful Paint <1s
- **SEO:** Complete meta tags, structured data, sitemap
- **Animations:** Framer Motion for scroll-triggered effects
- **Deployment:** GitHub Actions workflow for automated builds

Tech Stack: Next.js 14, TypeScript, Tailwind CSS, Shadcn/ui, Framer Motion, GitHub Pages

Live URL: abishek-maharajan.online

---

9. NotionClone - Collaborative Document Editor

Objective: Build Notion-like editor with real-time collaboration.

Project Scope:

- Block-based editor with drag-and-drop functionality
- Real-time synchronization across multiple users
- Rich text editing with markdown support
- Hierarchical document structure with navigation
- File uploads with cloud storage integration

Technical Achievements:

- **Real-time Sync:** Convex for live data synchronization with conflict resolution
- **Editor:** Custom block-based editor with slash commands
- **Collaboration:** Live cursor tracking and presence indicators
- **File Management:** EdgeStore integration for media uploads
- **Authentication:** Clerk for secure user management

Tech Stack: Next.js, TypeScript, Convex, EdgeStore, Clerk Auth, Tailwind CSS, Tiptap Editor

---

10-12. Additional Web Projects

- **React Dashboard Projects:** Multiple admin interfaces with data visualization
- **ChatGPT Clone:** Full conversational AI app with streaming responses
- **Z - Community ATS Platform:** Enterprise ATS with AI-assisted development

---

Infrastructure & Data Engineering Projects

13. ELK Stack - Log Analysis Infrastructure

Objective: Deploy complete observability stack for application monitoring.

Project Scope:

- Deployed Elasticsearch 8.17.0 for log storage and search
- Configured Logstash pipelines for log ingestion and transformation
- Built Kibana dashboards for visualization and analysis
- Set up alerting for critical events

Technical Achievements:

- **Elasticsearch:** Cluster setup with 3 nodes, custom index templates, ILM policies
- **Logstash:** Custom grok patterns for application log parsing
- **Kibana:** 10+ dashboards for different use cases (performance, errors, security)
- **Monitoring:** Elastic Stack monitoring for infrastructure health

Use Cases:

- Application performance monitoring (APM)
- Security event analysis and SIEM
- Business intelligence from application logs
- Infrastructure capacity planning

Tech Stack: Elasticsearch 8.17.0, Logstash 8.17.0, Kibana 8.17.0, Beats, Nginx

---

14. Minio Object Storage

Objective: Self-hosted S3-compatible storage for file management.

Technical Achievements:

- Deployed Minio server with multi-tenant bucket configuration
- Implemented bucket policies for fine-grained access control
- Integrated with applications using S3 SDK
- Set up versioning and lifecycle policies

Tech Stack: Minio Server, S3 API, Docker

---

15. BET ARCHIVE - Elasticsearch Analytics

Objective: Time-series betting data analytics platform.

Technical Achievements:

- Ingested betting data into Elasticsearch with time-series indexing
- Built statistical models for pattern detection
- Created Jupyter notebooks for exploratory data analysis
- Implemented real-time data pipeline

Tech Stack: Elasticsearch 8.17.0, Node.js, Jupyter Notebooks, Pandas, NumPy

---

16. n8n Self-Hosted AI Starter Kit

Objective: Complete workflow automation platform with local AI.

Project Scope:

- Deployed n8n with 400+ integration nodes
- Integrated Ollama for local LLM processing
- Set up Qdrant vector database for semantic search
- Created PostgreSQL backend for workflow storage

Technical Achievements:

- **Docker Compose:** Multi-container orchestration
- **AI Workflows:** Created 10+ workflow templates combining n8n and local AI
- **Vector Search:** Qdrant integration for RAG workflows
- **Custom Nodes:** Built custom n8n nodes for specific use cases

Tech Stack: n8n, Docker Compose, Ollama, Qdrant, PostgreSQL, Redis

---

Automation & Scraping Projects

17. AI WebScraper - Intelligent Content Extraction

Objective: Build production-grade scraping system with AI-powered content cleaning.

Project Scope:

- Implemented modern async web crawling framework
- Built intelligent content extraction with AI-based cleaning
- Created markdown generation for documentation sites
- Handled both static and JavaScript-rendered sites

Scraping Targets:

- Apache Superset complete documentation
- Medium articles with paywall bypass
- Metabase API documentation
- Custom business application content

Technical Achievements:

- **Async Crawling:** Concurrent requests with rate limiting and retry logic
- **Content Cleaning:** AI-based removal of navigation, ads, and boilerplate
- **Sitemap Discovery:** Automatic URL discovery from sitemap.xml
- **Format Conversion:** HTML to clean markdown with preserved structure
- **Session Management:** Browser session reuse for efficiency

Tech Stack: Python, Crawl4ai, Selenium, BeautifulSoup4, Playwright, aiohttp

Project Outcome: Successfully scraped 1000+ pages across multiple sites with 98% content quality.

---

18. AWS Scraper

Objective: Specialized scraper for AWS documentation.

Technical Achievements:

- Puppeteer-based browser automation
- Complex authentication handling
- Download management for PDF resources

Tech Stack: Node.js, Puppeteer, Axios

---

19. TMUX Orchestrator - Multi-Agent Coordination

Objective: Build 24/7 autonomous AI agent system.

Technical Achievements:

- Terminal multiplexer for persistent agent sessions
- Claude API integration for intelligent task routing
- Multi-agent coordination with shared memory

Tech Stack: tmux, Claude API, Bash, Python, Google Cloud SDK

Enterprise & Business Applications

20. SRE Projects - DevOps Infrastructure

Objective: Implement site reliability engineering best practices.

Project Scope:

- Container orchestration with Kubernetes
- Monitoring stack with Prometheus and Grafana
- CI/CD pipelines for automated deployment
- Health checks and circuit breakers

Tech Stack: Python Flask, Docker, Kubernetes, Prometheus, Grafana, Helm

---

21. EOM Maaxly Development

Objective: Structured development with comprehensive progress tracking.

Technical Achievements:

- Multi-cloud architecture comparison (AWS/GCP/Azure)
- Cost optimization strategies

- Progressive development milestones

Tech Stack: Google Cloud Platform, Supabase, Kafka, Redis, MongoDB Atlas

---

22. Accenture Hackathon

Objective: Rapid enterprise AI solution development.

Available Use Cases:

- Inventory optimization for retail (AI-driven stock management)
- Personalized e-commerce recommendations (ML-based)
- AI for sustainable agriculture (smart farming)
- AI for elderly care and support (healthcare tech)
- AI-powered job application screening (HR automation)
- AI-driven customer support (multi-agent systems)

Tech Stack: Multiple based on use case selection

System Programming & Learning

23. ZED Editor - Rust Development Environment

Objective: Explore systems programming with Rust.

Key Learnings:

- Rust language fundamentals
- GPU acceleration for editor rendering
- Real-time collaboration architecture

Tech Stack: Rust, Cargo, GPU Acceleration

---

24. MTW2025 - Tech Conference Experience

Objective: Industry insights and professional networking.

Key Takeaways:

- Industry leadership and market dynamics
- Innovation processes in organizations

- Technology landscape and emerging trends
- Professional networking strategies

Technical Skills (Comprehensive)

Programming Languages

Proficient: Python, JavaScript, TypeScript, SQL

Intermediate: C, PowerShell, Bash

Learning: Rust (via ZED editor project)

Web Development

Frontend Frameworks & Libraries:

- React.js (Hooks, Context API, custom hooks)
- Next.js 14 (App Router, Server Components, Static Export)
- Angular (SPA, NgRx, RxJS)
- Shadcn/ui, Material-UI (MUI), DaisyUI, Tailwind CSS

Backend Frameworks:

- Node.js/Express.js (RESTful APIs, middleware, authentication)
- Python Flask (microservices, API development)
- FastAPI (async Python, automatic API documentation)

Databases:

- **SQL:** PostgreSQL, MSSQL (T-SQL, stored procedures, indexing)
- **NoSQL:** MongoDB (aggregation pipelines, indexing), Redis (caching, pub/sub)
- **Vector:** ChromaDB, Qdrant (embeddings, semantic search)
- **ORM:** Sequelize, Mongoose, SQLAlchemy

API Development:

- RESTful API design and implementation
- GraphQL (queries, mutations, subscriptions)

- WebSocket (Socket.io, native WebSocket)
- API authentication (JWT, OAuth 2.0, API keys)

AI/ML & LLMs

LLM Platforms & APIs:

- OpenAI API (GPT-4, GPT-4 Vision, DALL·E 3, Whisper)
- Google Gemini API (Gemini Pro, Gemini Pro Vision)
- Anthropic Claude API
- Azure OpenAI Service

AI Frameworks:

- LangChain (chains, agents, memory, callbacks)
- LangGraph (state machines, workflows, multi-agent)
- LangSmith (evaluation, monitoring, tracing)
- Ollama (local model deployment)
- Llama.cpp (quantization, optimization)

ML Libraries:

- PyTorch (neural networks, custom models, training loops)
- Transformers (Hugging Face - BERT, GPT, T5)
- Scikit-learn (classical ML, preprocessing, evaluation)
- NLTK (tokenization, POS tagging, sentiment analysis)
- spaCy (NLP pipelines, entity recognition)

Voice & Vision:

- ElevenLabs (TTS, voice cloning)
- Deepgram (STT, real-time transcription)
- Tesseract, EasyOCR (OCR, document processing)
- OpenCV (image preprocessing, computer vision)

Vector Databases & RAG:

- ChromaDB (local vector store)

- Qdrant (production vector search)
- FAISS (Facebook AI Similarity Search)
- Sentence-transformers (embedding generation)
- RAG architecture patterns (retrieval, reranking, generation)

Data & Analytics

Data Storage & Processing:

- Elasticsearch (full-text search, aggregations, EQL)
- Apache Kafka (streaming, event-driven architecture)
- Redis (caching, session management, pub/sub)

Analytics & BI:

- Apache Superset (dashboards, SQL Lab, charts)
- Metabase (self-service BI, questions, dashboards)
- Kibana (log analysis, visualizations, Canvas)
- Jupyter Notebooks (exploratory analysis, reporting)

Data Processing:

- Pandas (data manipulation, time-series)
- NumPy (numerical computing)
- ETL pipeline design and implementation

DevOps & Cloud

Cloud Platforms:

- **AWS:** EC2, S3, Lambda, RDS, CloudWatch (Certified Cloud Practitioner)
- **Google Cloud:** Compute Engine, Cloud Storage, Cloud Functions
- **Azure:** Azure AI Foundry, Communication Services, Blob Storage

Containerization & Orchestration:

- Docker (Dockerfile, multi-stage builds, networking)
- Docker Compose (multi-container apps, volumes, networks)
- Kubernetes (deployments, services, ingress, helm)

- Helm (chart development, templating)

CI/CD:

- Git (branching strategies, rebase, cherry-pick)
- GitHub Actions (workflows, matrix builds, secrets)
- GitHub Pages (static site deployment)

Infrastructure & Monitoring:

- ELK Stack (Elasticsearch, Logstash, Kibana)
- Prometheus (metrics collection, PromQL)
- Grafana (dashboards, alerting)
- Nginx (reverse proxy, load balancing)

Infrastructure as Code:

- Docker Compose for local development
- Kubernetes YAML manifests
- Helm charts for application deployment

Automation & Integration

Workflow Automation:

- n8n (visual workflows, 400+ integrations)
- Power Automate (RPA, cloud flows)

Web Scraping:

- Selenium (browser automation, complex interactions)
- Puppeteer (headless Chrome, PDF generation)
- Playwright (cross-browser testing, screenshots)
- BeautifulSoup4 (HTML parsing)
- Scrapy (large-scale scraping)
- Crawl4ai (modern async crawling)

API Integration:

- RESTful API consumption

- Webhook handling
- OAuth 2.0 flows
- Rate limiting and retry strategies

System Integration:

- Gmail API (email automation)
- Google Calendar API (scheduling)
- Microsoft Teams API (meetings, messaging)
- Azure Communication Services (email, SMS)
- Job board APIs (LinkedIn, Naukri)

Development Tools & Practices

Version Control:

- Git (advanced workflows, rebasing, conflict resolution)
- GitHub (PRs, code review, project management)

Code Quality:

- ESLint, Prettier (code formatting)
- SonarQube (SAST, code quality metrics)
- TypeScript (type safety, interfaces, generics)
- Pytest (unit testing, fixtures, mocking)
- Jest (JavaScript testing, snapshots)

Development Environment:

- VS Code (extensions, debugging, SSH remote)
- Postman (API testing, collections, environments)
- ZED Editor (Rust-based, GPU-accelerated)

Documentation:

- Markdown (technical writing)
- OpenAPI/Swagger (API documentation)
- JSDoc, Pydoc (code documentation)

Education

Bachelor's in Computer Science

St. Joseph's Institute of Technology

August 2022 – Present | OMR, Chennai

- Cumulative GPA: **7.55 / 10**
- Relevant Coursework: Data Structures, Algorithms, Database Systems, Web Technologies, Machine Learning, Cloud Computing, Software Engineering
- Academic Projects: Sybil Attack Detection Model (96% accuracy), Multiple web applications
- Extra-curricular: Technical club participation, hackathon involvement

Diploma in Computer Science and Engineering

Apollo Polytechnic College

June 2019 – May 2022 | Chennai

- Percentage: **92 / 100** (Distinction)
- Strong foundation in programming fundamentals, database concepts, and web technologies
- Early exposure to software development and project-based learning

---

Certifications

Cloud & Infrastructure

- **AWS Certified Cloud Practitioner (CLF-C02)** - Score: 914/1000 (Active)
- **AWS re/Start Graduate** - Comprehensive cloud training program
- **Introduction to AI and Machine Learning** - Google Cloud

Development & APIs

- **Postman API Fundamentals Student Expert** - API testing and development
- **MongoDB for Students** - MongoDB University
- **N8n Community Course Level 1** - Workflow automation

AI & Machine Learning

- **Llama3** - DataCamp (Meta's LLM fundamentals)

Achievements & Recognition

Technical Achievements

- **24+ Weekend Projects Completed** (2025) - Comprehensive exploration of AI, web dev, and infrastructure
- **96% Model Accuracy** - Sybil Attack Detection using deep learning
- **914/1000 AWS Score** - Top 10% of test takers globally
- **Multiple Production Deployments** - Portfolio site, dashboards, and web applications live
- **Open Source Contributions** - Fork and enhance methodology with documentation contributions

Professional Impact

- **70% Reduction** in resume screening time through automated ATS at Clubits
- **85% Decrease** in onboarding errors with OCR verification system
- **100% Automation** of routine appointment calls with voice agents
- **80% Task Automation** with ambient email agent (personal project)

Learning & Development

- **MTW2025 Conference Participation** - Technology trends and networking
- **Consistent Weekend Learning** - Year-long commitment to skill development
- **Fork & Enhance Strategy** - Practical learning through open-source adaptation
- **AI-Assisted Development** - Early adopter of AI coding assistants

---

Project Statistics & Portfolio Metrics

GitHub Activity (2025)

- **Public Repositories:** 25+
- **Total Commits:** 500+ across personal and work projects
- **Languages:** Python (40%), JavaScript/TypeScript (35%), HTML/CSS (15%), Others (10%)
- **Active Projects:** 10+ in various stages of development

Learning Metrics

- **Total Projects:** 24+ major implementations
- **Technologies Learned:** 50+ tools, frameworks, and platforms
- **Weekend Hours:** ~400+ hours of focused learning
- **Documentation:** Comprehensive project documentation for all major projects

Project Distribution

- AI & Machine Learning: 6 projects (25%)
- Web Development & Full-Stack: 6 projects (25%)
- Data Engineering & Infrastructure: 4 projects (17%)
- Automation & Scraping: 3 projects (12%)
- Enterprise & Business Applications: 3 projects (12%)
- System Programming & DevOps: 2 projects (9%)

Interests & Additional Information

Professional Interests

- **AI Agent Architecture:** Multi-agent systems, workflow orchestration, autonomous agents
- **Full-Stack Development:** Building end-to-end applications with modern frameworks
- **Cloud Infrastructure:** Scalable, reliable systems design and deployment
- **DevOps & SRE:** Observability, monitoring, and production reliability
- **Open Source:** Contributing to and learning from open-source projects

Personal Interests

- **Photography** 📸 - Creative outlet and visual storytelling
- **LLM Enthusiast** 🤖 - Following latest research and model releases
- **Technical Writing** ✍️ - Documenting learning journeys and project details
- **Conference Attendance** - Staying current with industry trends
- **Weekend Learning** - Continuous skill development through hands-on projects

Languages

- **English:** Professional working proficiency
- **Tamil:** Native proficiency
- **Hindi:** Limited working proficiency

References

Professional references available upon request.

Additional Resources

Portfolio Website: abishek-maharajan.online

GitHub Profile: github.com/abishek-m

LinkedIn: linkedin.com/in/abishek-m

Email: maharajanabishek@gmail.com

💡 **What Sets Me Apart:**

1. **Rapid Learning Ability:** 24+ substantial projects in 2025 demonstrates ability to quickly master new technologies
2. **Production Experience:** Not just tutorials - built real systems deployed in production at Clubits Solutions
3. **Full-Stack Breadth:** Comfortable across frontend, backend, AI/ML, and infrastructure layers
4. **AI-First Mindset:** Early adopter integrating AI into every aspect of development workflow
5. **Self-Directed:** Weekend learning philosophy shows initiative and passion for continuous growth
6. **Documentation Culture:** Comprehensive documentation of all learning and projects
7. **Problem-Solving:** Real-world projects solving actual problems, not just toy examples
8. **Modern Stack:** Focus on latest tools and frameworks (Next.js 14, LangGraph, Ollama, etc.)

Last Updated: October 2025

This detailed resume showcases the comprehensive technical journey, hands-on learning approach, and production experience gained through professional work and dedicated weekend learning projects.