Zafeer Rangoonwala

+1-469-268-0532 | zafeer02@gmail.com | linkedin.com/in/zafeerrangoonwala | Portfolio

Education

The University of Texas at Dallas — Richardson, TX
Master of Science, Computer and Information Science — GPA: 3.5 — *May 2026*

Indus University — Ahmedabad, India
Bachelor of Science, Computer and Information Science — GPA: 3.5 — *May 2024*

Skills

Languages: Python | SQL | Java | JavaScript/TypeScript | Go | PySpark
Frameworks: Django | Flask | PyTorch | React | Next.js | Node.js | Spring Boot | GraphQL | Apache Kafka
Data & Storage: PostgreSQL | MongoDB | Amazon RDS | Hadoop | Elasticsearch
Cloud & DevOps: AWS (EC2 | S3 | Lambda | RDS) | Docker | Terraform | GitHub Actions | Kibana

Work Experience

Technology & Operations Worker — May 2025 – Present
University of Texas at Dallas - International Center

- **Coordinated** cross-department IT onboarding and device provisioning by acting as a central point of collaboration between users, administrators, and support teams, reducing communication friction and accelerating service delivery with minimal management overhead.
- **Streamlined** end-to-end asset lifecycle and support workflows (imaging, access setup, deprovisioning, and issue resolution), standardizing processes and documentation to minimize handoffs, eliminate bottlenecks, and improve operational efficiency and user experience.

Data Analyst — Jan 2024 – Jul 2024
Delta Systems

- **Architected** scalable **ETL** pipelines using Python (Pandas, NumPy) and **SQL**, consolidating data from multiple structured sources into **PostgreSQL**, reducing end-to-end processing time by **30%** and improving data reliability for downstream analytics.
- **Designed** robust **data transformation** and validation workflows, implementing schema normalization, data quality checks, and feature engineering pipelines to generate ML-ready datasets, improving **clustering** accuracy and reducing noise across **high-volume** records.
- **Optimized** complex **SQL** workloads through query **refactoring**, **indexing** strategies, and execution plan analysis, significantly reducing query **latency** and enabling near **real-time analytical** reporting for business stakeholders.
- **Implemented** unsupervised machine learning models using scikit-learn (**k-means clustering**) to segment large-scale customer datasets, driving targeted marketing strategies and measurable improvements in **operational efficiency**.

Projects

Traceline – Pipeline Anomaly Detection System | ***Python, AWS, React, ML*** | **TidalHack '26 Winner**

- **Designed** a **feature-matching** pipeline to link corrosion/anomaly records across inspections, improving **match confidence** and reducing manual review effort by **70%**.
- **Built** a change-over-time (**growth**) analysis workflow to quantify corrosion progression and **flag high-risk** segments for prioritization using **gradient boosting** trees.
- **Developed** a **web dashboard** to visualize aligned runs and **anomaly trends** in an interactive view.
- **Implemented data validation** and **quality checks** (schema rules, **outlier detection**, missing-field handling) to ensure reliable outputs across **15 years** and **11,500 rows** of inspection data.

PotionWatch – Fraud Detection System | ***Python, Flask, Pandas, NumPy***

- **Developed** an **automated fraud detection** system for EOG Resources to identify discrepancies in transport documentation, detecting **suspicious tickets** and **missing records** across transactions, preventing **revenue loss** and improving **audit compliance**.
- **Built** a **RESTful API** with statistical **outlier detection** and **real-time** ticket-matching algorithms, processing **time-series** data across **12 collection sites**, enabling the operations team to investigate anomalies.

Serverless Application using Terraform | ***AWS, Terraform, JavaScript***

- **Architected** a **full-stack serverless** to-do application on **AWS** using **Terraform** for **Infrastructure as Code**.
- **Automated** the provisioning of a **serverless backend** with **AWS Lambda** (Python) and **AWS API Gateway** to create **RESTful API** endpoints for **CRUD** operations.
- **Implemented** a **data persistence** layer using **AWS DynamoDB** to efficiently store and manage application data.
- **Configured AWS S3** for **secure** static website hosting, ensuring a fully **reproducible** and **version-controlled** infrastructure.