Steven Liu — San Francisco, CA | GitHub | LinkedIn | steventheliu@gmail.com |

WORK EXPERIENCE

Software Engineer — AIMdyn — Santa Barbara, CA (Remote) — (04/2022 - 08/2024)

- Architected a distributed MLOps platform to manage the full machine learning lifecycle, from experimentation to production deployment, serving as the core Python framework for all team research.
- Engineered core orchestration engine of a distributed system with Redis-based task queue and multiprocessing to decouple GUI from long running tasks, eliminating UI freezes and enabling asynchronous tasks.
- Designed and implemented a secure auth service and hybrid execution API, enabling multi-user collaboration and seamless job routing to local or remote cloud resources, improving resource utilization and scalability.
- Established platform reliability by implementing idempotent processing, sliding-window rate limiting, and circuit breakers, reducing system failures and ensuring stability under heavy load.
- Integrated MLflow for automated experiment tracking, logging parameters, metrics, and artifacts for reproducibility.
- Automated the evaluation pipeline for COVID-19 forecasting models, slashing benchmarking time against external competitors.
- Drove full-stack feature development by refactoring legacy MATLAB research code into maintainable Python APIs (FastAPI, Django) and integrating them into a production system with automated testing.
- Preprocessed data with Pandas/NumPy; worked with varied data formats (.csv, .json, .parquet, SQL).
- Championed software best practices by introducing CI/CD pipelines, containerization (Docker), and infrastructure documentation, standardizing development across cross-functional teams of ML engineers and data scientists.
- Collaborated with ML engineers, data scientists, and DevOps teams, documenting pipelines and frameworks for long-term sustainability.

EDUCATION

University of California, San Diego — Data Science B.S — GPA 3.6/ 4.0 — (09/2018 - 06/2021)

- **Data Structures & Algorithms** - Recursion, Higher-Order Functions, OOP, Time Complexity, Data Types
- **Application of Data Science** - A/B testing, Statistical Inference, ML Algorithms, Web Scraping
- **Database Management** - Relational Databases, Schema Design, SQL, Query Optimization
- **Scalable Analytics Systems** - Big Data, Memory Hierarchy, Distributed Systems, ETL, Model Deployment
- **Modeling & Machine Learning** - Deep Learning, NLP, Robotics, Model Selections, Evaluation Metrics

TECHNICAL SKILLS

- **Languages**: Python, CSS, HTML, JavaScript, TypeScript, SQL, C++, Java, MATLAB, R, Go
- **Web Development**: Flask, Django, React, FastAPI, Gin, TailwindCSS, ShadCN
- **Data Science:** Pandas, NumPy, SciPy, Scikit-Learn, TensorFlow, PyTorch, Spark, Dask, Hadoop, Matplotlib
- **DevOps:** Docker, Kubernetes, Airflow, Grafana, Prometheus
- **Data Storages:** PostgreSQL, MongoDB, Cassandra, ElasticSearch, Redis, Kafka
- **Testing:** Cypress, Selenium, Postman, GitLab CI, PyTest, JUnit, Coverage, UnitTest, Locust
- **Cloud:** AWS (S3, EC2, Elastic Beanstalk, RDS, Lambda, CloudFront), Render
- **Version Control & Web Scraping & GUI Development**: Git, GitHub, GitLab, BeautifulSoup, Tkinter

CERTIFICATIONS

- **AWS Certified Cloud Practitioner** - 11/2024

PERSONAL PROJECTS

Online Catan with Chat — Fullstack React & FastAPI (In Progress)

Stack: Python, React, Redux, FastAPI, Websockets, Node, Vite

- Built a full-stack multiplayer game platform featuring a React frontend and FastAPI backend, connected via WebSocket for real-time bidirectional communication.
- Built Catan board pieces, and physics with TypeScript with the Three.js framework
- Sped up UI development with TailwindCSS and ShadCN components
- Developed a scalable FastAPI WebSocket server managing game sessions, lobby state, and chat messages with efficient concurrency handling, enabling real-time updates for all connected clients.
- Designed REST and WebSocket APIs for lobby management, game creation/joining, and chat messaging, supporting pagination and robust error handling.
- Implemented player session management with unique player IDs issued via WebSocket handshake and persisted on the client for seamless reconnection and state synchronization.
- Integrated WebSocket middleware in React/Redux to dispatch actions and handle incoming server messages, providing a responsive, event-driven UI with minimal latency.
- Coordinated state synchronization between backend and frontend through structured message protocols, including lobby updates, game state changes, and chat communication.
- Enhanced UX with live lobby updates, real-time chat room messaging (including private and group chats), and in-app error reporting with a global error banner.
- Ensured smooth UI layout with responsive design using CSS Flexbox, keeping game area and lobby consistently sized alongside chat and sidebar components.

FoodLens – Full-Stack Food Recognition & Nutrition App

Stack: Django, React.js, Golang, AWS (S3, SNS, IAM), PostgreSQL, Redis, Celery, Docker, Kubernetes

- Built a scalable web platform to identify foods from images and return nutritional data using third-party ML APIs.
- Developed secure, rate-limited APIs with Django REST; integrated Golang services for optimized data retrieval.
- Implemented JWT authentication, IAM-based access control, XSS/CSRF protection, and SNS notifications.
- Used Celery + Redis for background task processing (e.g., image classification, nutrition lookup).
- Deployed with Docker/Kubernetes; monitored health via Prometheus/Grafana; enabled PostgreSQL read replication.
- Set up CI/CD with GitHub Actions, pre-commit hooks (linting, formatting), and automated shell scripts.
- Achieved 30–40% improvement in API response times through caching and database tuning.

Resume Chatbot – RAG-Powered LLM QA System for Personalized Resumes

Stack: LangChain, FAISS, HuggingFace Embeddings, OpenAI-compatible LLM (Ollama/Mistral), Python, Docker, React (Next.js)

- Built an interactive chatbot that accurately answers questions about resumes using Retrieval-Augmented Generation (RAG).
- Ingested and structured resume PDFs using LangChain document loaders and custom section labeling logic.
- Split and embedded documents with `bge-small-en-v1.5` (HuggingFace) and stored them in a FAISS vector database.
- Designed a strict QA prompt and integrated with Ollama via an OpenAI-compatible API using Mistral models.
- Enabled contextual question answering with section-aware retrieval, metadata tagging, and retry logic with test validation.
- Developed automated test harness with 20+ QA test cases and retries to ensure reliability and consistency of answers.
- Saved and reused vector stores for faster startup and optimized real-time response time with similarity search.

Wildlife Tracker – Geospatial & Time-Series System for Wildlife Monitoring

Stack: FastAPI, TimescaleDB, PostGIS, SQLAlchemy, React (Vite), TypeScript, Leaflet.js, Docker, Locust, Kafka, Redis

- Developed a full-stack system to track wildlife herds and families over time and geography.
- Modeled animal populations with nested relationships and enabled spatial/time-series analytics.
- Built RESTful APIs with FastAPI and PostGIS to answer complex queries
- Stored tracking data using TimescaleDB for time-series efficiency; supported geospatial filtering and joins.

- Created an interactive map UI (React + Leaflet.js) to visualize tracks, events, and metrics with real-time filtering.
- Containerized frontend, backend, and database for local orchestration with Docker Compose.
- Simulated load and verified API performance under stress using Locust.
- Implemented Kafka consumer architecture, supporting both batch and single-message processing
- Built a Dead Letter Queue (DLQ) for isolating and inspecting malformed or failing messages
- Integrated Redis to cache recent wildlife metrics and spatial summaries, improving API performance and minimizing expensive DB queries for frequently requested data.

League of Legends Jungler Pathing Predictor – ML + Computer Vision Project

Stack: Python, Scikit-learn, PyTorch, PyTesseract, PyQT, Riot API, Docker, MLflow, NumPy, Pandas

- Built a predictive system to model enemy jungler movement using historical match data and in-game visual capture.
- Implemented LSTM-based sequence model in PyTorch to learn spatial-temporal patterns from engineered game state data
- Developed historical sequence generator with timestamped input windows; trained and evaluated on 3000+ matches
- Created a PyQT overlay + Tesseract-OCR pipeline to extract real-time creep score (CS) from in-game screen regions
- Built resumable ETL pipeline using Riot API data with deduplication, preprocessing, and feature engineering
- Benchmarked LSTM against baseline regressors; visualized errors and model drift using residuals and scatter plots
- Performed EDA and preprocessing to clean raw Riot API data; normalized features and filtered leakage-prone columns
- Ongoing work includes replay parsing for higher-granularity labels and potential transformer-based models

More Projects

- **Transaction Data Pipeline** – Built a hybrid batch/streaming data pipeline using Apache Spark, Kafka, and Airflow; stored processed data in PostgreSQL and MongoDB with real-time CDC-based syncing.
- **Event Ticket Manager** – Designed a microservices-based event booking platform using Redis, Kafka, and ElasticSearch; prevented overbooking with optimistic locking and updated ticket availability in real time.
- **Real-Time Climate Monitoring System** – Used ESP32 to collect sensor data; stored over 1M records in PostgreSQL and visualized metrics via a JavaScript map dashboard.
- **NBA Team Seed Classification** – Trained GraphSAGE (88% accuracy), GCN, GCN-LPA, and Logistic Regression models to predict NBA playoff rankings; deployed models in Docker containers.
- **Payment Gateway Microservice** – Built a TypeScript/Node.js backend to process transactions through multiple 3rd-party APIs using PostgreSQL for persistence.