Campus Connect: A Cloud-Native Academic Collaboration Platform

Edwin Biswas

Alex Cooper

Tyler Geiger

Kadin Matotek

December 2025

Contents

1 Project Overview

Campus Connect is a distributed, cloud-native collaboration platform designed to support class discussions, file sharing, and lightweight real-time communication. The system is deployed on raw physical machines provisioned through CloudLab and orchestrated using a fully automated Terraform → Ansible → Kubernetes toolchain. All core services run inside a production-style multi-node Kubernetes cluster with persistent storage supplied by Rook-Ceph.

At a high level, the platform is composed of:

- a **NestJS backend** providing REST APIs, authentication, and WebSocket channels;
- a **React/Vite frontend** served through Node.js;
- **MongoDB**, **Redis**, **RabbitMQ**, and **MinIO** as supporting services for state, caching, messaging, and file storage;
- a separate **LLM microservice** (implemented but not yet integrated) for future RAG-based classroom assistance;
- **ChromaDB** and a preprocessing pipeline for document embeddings and vector storage.

The cluster runs on multiple CloudLab `rawpc` nodes configured via:

- **kubeadm** for cluster bootstrap,
- **containerd** as the runtime,
- **Flannel (VXLAN)** for pod networking,
- **MetalLB** for service load balancing,
- **ExternalDNS** for hostname management,
- **cert-manager** for TLS certificates, and
- **Rook-Ceph** to provide replicated block storage for all StatefulSets.

A custom Terraform provider was developed to automate CloudLab provisioning. While independent from this project, it enables Campus Connect to allocate physical machines, configure networking, and deploy Kubernetes and Ceph entirely through declarative IaC.

The result is a functional full-stack platform that demonstrates how a modern cloud application can be deployed, persisted, scaled, and managed on bare-metal academic infrastructure under realistic production patterns.

2 System Architecture

Campus Connect is designed as a distributed, cloud-native application composed of multiple cooperating services. The architecture follows a service-oriented model, deployed on Kubernetes, with explicit separation between the frontend client, backend API layer, foundational data services, and auxiliary asynchronous and future RAG systems. This section describes the behavior, interactions, and responsibilities of each component without addressing cluster-level concerns (covered in the Cloud and Cluster Architecture sections).

2.1 Frontend Architecture

The frontend is implemented as a React + Vite single-page application (SPA). After compilation, it is packaged into an NGINX container that serves static assets and proxies API requests to the backend through the Kubernetes Ingress controller.

Key Behaviors

- **SPA Routing**: Client-side routing handles classroom navigation, user profiles, and chat views.
- **Authentication**: JWT-based session tokens are retrieved at login and stored securely in memory.
- **WebSocket Integration**: Real-time chat, presence indicators, and system events are delivered via a persistent WebSocket connection to the backend gateway.
- **File Upload Pipeline**: Users upload documents through a signed-URL flow:
    1. Backend issues a pre-signed URL for MinIO.
    2. Frontend performs a direct PUT upload to MinIO.
    3. Backend receives an AMQP notification about upload completion.
- **Role-Aware Interface**: Professors have additional controls for class management and announcements.

2.2 Backend Architecture (NestJS)

The backend is built on NestJS and structured around modules, dependency-injected services, data models, and guards. It handles authentication, user/class/channel state management, WebSocket messaging, event publication, and file integration workflows.

Core Responsibilities

- **Authentication & Sessions**: Login and session tokens managed with Redis for revocation and short-lived session caching.
- **REST API Services**: Endpoints for classes, channels, messages, file operations, and user profiles.
- **WebSocket Gateway**: A dedicated gateway streams chat messages, typing signals, and presence updates.
- **Redis Caching (Planned)**: Frequently accessed data (e.g., channel metadata) cached to reduce MongoDB load.
- **Event-Driven Flows**: RabbitMQ is used for asynchronous events, including:
    - new file uploads,
    - profile picture updates,
    - future document preprocessing for RAG pipelines.
- **MinIO Integration**: Signed URL generation, upload validation, and event consumption from AMQP notifications.

2.3 Data Services (Logical Layer)

Data services form the foundation of the platform and operate independently of the application runtime.

MongoDB

Acts as the authoritative source for all persistent data:

- Users, classes, channels, and messages.
- File metadata stored alongside class resources.
- Indexed queries to support real-time workloads.

Redis

Stores ephemeral runtime state:

- authentication sessions,
- presence information,
- WebSocket channel membership,
- caching for high-read endpoints.

MinIO

Provides object storage equivalent to AWS S3:

- Two buckets: `profile-pictures` (public) and `uploads-private`.
- AMQP notifications to RabbitMQ for every completed object upload.

RabbitMQ

Message broker used for:

- backend integration events,
- MinIO event notifications,
- planned RAG request/response mediation.

ChromaDB (Planned)

A future vector database responsible for:

- embedding storage,
- classroom-level vector indexes,
- retrieval for RAG-based question answering.

2.4 LLM Microservice (Standalone)

A standalone RAG/inference service exists in its own repository and is fully functional independently of the cluster. It will be integrated in a later phase.

Current Capabilities

- Embedding generation using a lightweight sentence-transformer.
- Document preprocessing and chunking.
- Local in-container vector store (to be replaced with ChromaDB).
- REST and gRPC interfaces for inference tasks.

Planned Integration

The backend will eventually communicate with the LLM microservice through RabbitMQ:

1. Backend publishes a preprocessing or inference request.
2. LLM service consumes the task and processes it.
3. LLM service emits a response event to a dedicated callback queue.

2.5 User Stories and Data Flows

Campus Connect's interaction model is based on two foundational workflows.

User Story 1: File Upload and Event Handling

As shown in Figures 1 and 2 (*see included PDFs in `pics/`*), a student uploads class materials which flow through:

1. frontend → backend (signed URL),
2. frontend → MinIO (direct upload),
3. MinIO → RabbitMQ (event),
4. backend consumes event and updates class metadata.

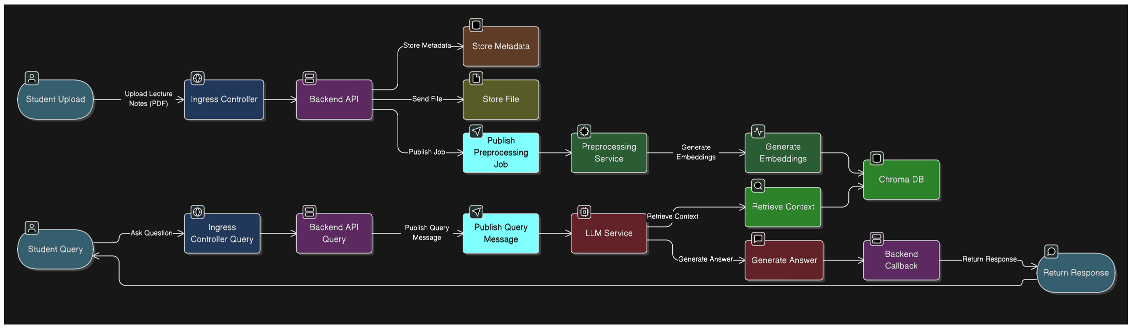

Figure 1: User Story 1: Student uploads lecture materials and triggers MinIO event pipeline.

User Story 2: Class Management

Professors manage classes, make announcements, and organize files. The backend coordinates all authorization and data updates, while RabbitMQ propagates system-level events.

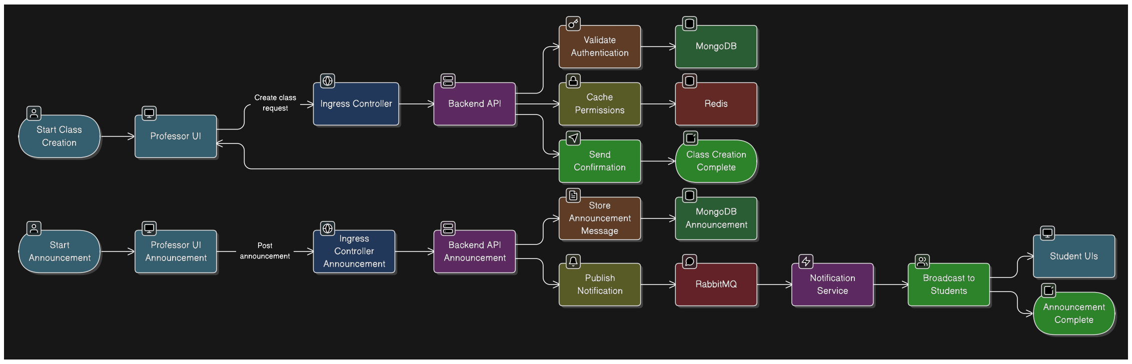

Figure 2: User Story 2: Professor management workflow (class, announcements, materials).

2.6 API Contracts

Campus Connect provides both REST and WebSocket APIs.

REST Endpoints

Examples include:

- `POST /api/auth/login`
- `POST /api/files/signed-url`
- `GET /api/classes`
- `POST /api/messages`

WebSocket Events

- `message.sent`
- `message.received`
- `user.presence`
- `typing.update`

Example Payload

```
{
  "classId": "CSC478",
  "channelId": "general",
  "content": "Is the assignment due tonight?"
}
```

3 Cloud Architecture

Campus Connect is deployed entirely on the CloudLab research testbed, using bare-metal (`rawpc`) hardware provisioned on demand. The system is designed so that an entire Kubernetes cluster can be created, destroyed, and recreated automatically through a single pipeline run. This section describes the underlying compute environment, topology design, and the Terraform-driven provisioning workflow that makes this possible.

3.1 CloudLab as the Execution Environment

CloudLab provides on-demand access to physical hardware, public IPv4 addressing, and configurable L2 networks. Unlike commercial clouds, CloudLab exposes the full nodes directly:

- No virtualization overhead.
- Full control of the kernel and container runtime.
- Complete network visibility (L2 LANs, routable public IPs).
- Multiple available aggregates (Utah, Clemson, Wisconsin, APT, Emulab).

These characteristics make CloudLab uniquely suited for educational DevOps projects that require:

- multi-node Kubernetes clusters,
- real storage systems (Ceph RBD and CephFS),
- load balancers without a cloud load balancer API,
- deterministic infrastructure provisioning across multiple runs.

3.2 Node Hardware Profile

For this project, we use the `d710` hardware profile. Each node provides:

- Intel Xeon E5530 @ 2.40 GHz (4 cores / 8 threads),
- 12 GB RAM,
- 500 GB system disk (`/dev/sda`),
- separate raw disk used for Ceph OSD (`/dev/sdb`),
- a routable public IPv4 address.

This hardware is sufficient for:

- a fully functioning Kubernetes control-plane,
- two or more worker nodes,
- Ceph RBD and CephFS storage pools,
- MinIO, MongoDB, Redis, and RabbitMQ,
- frontend + backend + future LLM microservice.

3.3 Topology and Network Model

Campus Connect uses a simple, repeatable network layout:

- **One control-plane node**: `kubeadm`
- **Two worker nodes**: `worker1`, `worker2`
- **One shared L2 LAN**: `lan0`

The LAN provides:

- full broadcast and ARP visibility (required for MetalLB L2 mode),
- direct pod-to-pod and node-to-node communication,
- stable cluster networking regardless of aggregate.

CloudLab assigns each node:

- one public IPv4 address,
- one private LAN address.

This makes it possible to:

- access any node directly for debugging,
- expose services with MetalLB LoadBalancers,
- configure ExternalDNS to point domain records at the dynamically assigned public IP.

3.4 Terraform-Defined Infrastructure

CloudLab infrastructure is provisioned using a custom Go-based Terraform provider created for this project. The provider communicates directly with CloudLab's XML-RPC API (no intermediate services or wrappers), allowing the entire cluster to be declared in HCL.

Provisioned Resources

The provider supports:

- `rawpc` nodes (bare metal),
- L2 `lan` networks,
- routed public IPs,
- blockstores and auxiliary hardware options,
- rich outputs describing experiment state.

CampusConnect Cluster Definition

The project uses a concise infrastructure specification (excerpt shown):

Listing 1: Terraform cluster definition (excerpt)

```
resource "cloudlab_portal_experiment" "CC2-Cluster" {
  name            = "CampusConnect"
  wait_for_status = "ready"

  rawpc { name = "kubeadm",  hardware_type = var.hardware_type,
          aggregate = local.aggregate_map[var.aggregate], routable_ip = true }

  rawpc { name = "worker1",  hardware_type = var.hardware_type,
          aggregate = local.aggregate_map[var.aggregate], routable_ip = true }

  rawpc { name = "worker2",  hardware_type = var.hardware_type,
          aggregate = local.aggregate_map[var.aggregate], routable_ip = true }

  lan {
    name = "lan0"
    interface { node = "kubeadm"  }
    interface { node = "worker1"  }
    interface { node = "worker2"  }
  }
}
```

Terraform abstracts CloudLab's experiment lifecycle by ensuring:

- The experiment is created.
- Nodes are allocated.
- The topology is wired.
- Public IPs and metadata are returned to the CI pipeline.

Typed Outputs

The provider returns a structured JSON object describing the instantiated experiment, including:

- node names,
- public IPv4 addresses,
- internal LAN addresses,
- expiration time,
- experiment URL.

The CI pipeline passes this map directly into Ansible, which transforms it into a live Kubernetes cluster.

3.5 Lifecycle Model

Campus Connect follows a reproducible "full lifecycle" workflow:

1. Terraform provisions the hardware and topology.
2. Ansible configures the OS, installs Kubernetes tooling, and bootstraps the cluster.
3. Helm deploys MinIO, MongoDB, Redis, RabbitMQ, Ingress, MetalLB, and the application services.
4. ExternalDNS binds domain names to dynamically assigned CloudLab IP addresses.

This approach guarantees that every deploy (whether for development, testing, or demonstration) begins from:

- clean bare-metal hardware,
- an empty Ceph cluster,
- a freshly bootstrapped Kubernetes installation.

Since CloudLab experiments expire automatically, the automation ensures that recreating the platform is fast and deterministic.

4 Cluster Architecture

Once CloudLab provisions the raw hardware, the CI pipeline transforms the machines into a fully functioning Kubernetes cluster using Ansible and kubeadm. This section details the configuration of the control plane, worker nodes, CNI networking, load balancing, ingress, DNS automation, and the storage infrastructure that supports the Campus Connect application stack.

4.1 Cluster Bootstrapping with Ansible and Kubeadm

The cluster is initialized through a multi-phase Ansible playbook (`ansible/playbooks/bootstrap.yml`). The workflow is fully automated:

1. Configure OS prerequisites on all nodes.
2. Install containerd with systemd cgroup support.
3. Install Kubernetes tooling: `kubeadm`, `kubelet`, `kubectl`.
4. Initialize the control plane via `kubeadm init`.
5. Distribute and execute the worker join command.
6. Deploy CNI, Ingress, MetalLB, ExternalDNS, and Ceph.

OS Configuration (roles/base)

Each node undergoes uniform preparation:

- swap is disabled and removed from `/etc/fstab`,
- required kernel modules (`overlay`, `br_netfilter`) are enabled,
- sysctl settings ensure proper packet forwarding and bridge networking,
- `git`, Python tooling, and dependencies are installed,
- the CC2-Cluster repository (with Helm charts) is cloned to the node.

Container Runtime (roles/containerd)

The container runtime is configured as follows:

- containerd installed from the official package repository,
- `SystemdCgroup = true` in `/etc/containerd/config.toml`,
- CRI socket availability waited on before kubeadm is invoked.

This ensures compatibility with the Kubernetes control-plane and predictable cgroup behavior.

Installing Kubernetes Tooling

Via `roles/kube-tools`, each node receives:

- `kubeadm` for initializing and joining clusters,
- `kubelet` for runtime orchestration,
- `kubectl` for management,
- `cri-tools` providing `crictl`.

Pinning package versions guarantees the cluster remains reproducible.

Control Plane Initialization

The control-plane node executes:

```
kubeadm init --pod-network-cidr=10.244.0.0/16
```

This yields:

- Admin kubeconfig for both root and the CloudLab user,
- A join command containing a token and CA hash,
- Certificates and cluster metadata stored under `/etc/kubernetes`.

Ansible distributes the join script to each worker node.

Worker Node Join

Workers run the exact join command generated by kubeadm. Once registered, they become schedulable nodes for workloads, Ceph OSDs, and infrastructure services.

4.2 Networking Architecture

Campus Connect uses a layered approach to cluster networking to replicate a real cloud environment while running on raw hardware.

CNI: Flannel

Flannel is applied immediately after kubeadm initialization. The choice of Flannel is intentional:

- Simple, stable overlay networking for educational clusters.
- Zero cloud-provider dependencies.
- Reliable pod network CIDR matching the kubeadm configuration.

Deployment finishes only once all DaemonSet pods are reporting Ready across `kubeadm`, `worker1`, and `worker2`.

4.3 Load Balancing: MetalLB

Because CloudLab does not provide a cloud load balancer API, Kubernetes LoadBalancer services would fail without an alternative. Campus Connect uses MetalLB in L2 mode.

Dynamic IP Pool Generation

The Ansible `roles/metallb` role performs:

1. Detect the node's primary network interface.
2. Parse the IPv4 subnet (e.g., `10.10.1.0/24`).
3. Generate a VIP address pool at the end of the subnet: `.200--.220`.

This ensures compatibility across CloudLab aggregates where subnets vary.

L2 Advertisement Behavior

MetalLB speakers:

- announce VIPs via ARP,
- migrate VIPs across nodes on failure,
- provide real load-balanced endpoints for Ingress-NGINX,
- support external exposure of application services.

The result is functionally equivalent to a cloud load balancer.

4.4 Ingress: NGINX Ingress Controller

The Ingress controller is installed via the upstream Helm chart. It receives a MetalLB-assigned LoadBalancer IP and exposes multiple hostname-based routes:

- `app.campusconnectwcu.com` ⟶ frontend service,
- `api.campusconnectwcu.com` ⟶ backend service,
- `minio.campusconnectwcu.com` ⟶ MinIO console,
- `prom.campusconnectwcu.com` ⟶ Prometheus,
- `grafana.campusconnectwcu.com` ⟶ Grafana.

4.5 DNS Automation: ExternalDNS + Cloudflare

Since CloudLab assigns new public IPs with every experiment, DNS automation is critical. ExternalDNS monitors the cluster and updates Cloudflare records.

How It Works

- Watches Ingress and LoadBalancer resources.
- Reads a Cloudflare API token stored in Ansible Vault.
- Creates or updates DNS A records under: `campusconnectwcu.com`.
- Uses `txtOwnerId` to avoid conflicts or stale records.

Why It's Required

Without ExternalDNS, each redeployment would require manual DNS changes. With automation:

- URLs remain stable,
- public endpoints update instantly,
- grading, demos, and development environments are fully reproducible.

4.6 Storage System: Rook-Ceph

Campus Connect uses a real distributed storage backend rather than hostPath volumes.

Rook-Ceph Operator

The CI pipeline installs the Rook operator from the official Helm chart. This provides the controllers necessary to:

- create Ceph clusters,
- manage block devices,
- provision RBD and CephFS volumes,
- monitor cluster health.

Ceph Cluster Deployment

A custom Helm chart (under `helm/ceph/`) configures:

- Monitors (MONs) — one per node for redundancy.
- Manager (MGR) — monitoring + administration.
- OSDs — one per node, using `/dev/sdb`.
- Rook-Ceph block pool — replicated size 3.
- CephFS filesystem — metadata + data pools.
- StorageClasses:
  - `rook-ceph-block` (RBD)
  - `rook-cephfs` (CephFS)

This ensures that StatefulSets such as MongoDB, Redis, MinIO, and RabbitMQ have highly durable storage.

4.7 Namespace and Service Organization

To maintain clarity and isolation, workloads are organized as follows:

- **rook-ceph** — storage subsystem (operator, MON, MGR, OSDs).
- **metallb-system** — MetalLB speakers and controllers.
- **ingress-nginx** — ingress controller and admission components.
- **kube-system** — core cluster infrastructure.
- **services** — application layer:
  - backend,
  - frontend,
  - MinIO,
  - MongoDB,
  - Redis,
  - RabbitMQ.
- **monitoring** — Prometheus, Grafana, exporters.

This namespace layout mirrors industry practices and simplifies debugging and observability.

4.8 Cluster Behavior and Scheduling Model

Stateful vs Stateless Workloads

- **Stateful**: use RBD-backed PVCs, scheduled via StatefulSets. (MongoDB, MinIO, Redis, RabbitMQ)
- **Stateless**: deployed as Deployments with rolling updates. (backend, frontend, future LLM service)

Pod Health Probes

Application Deployments define:

- `livenessProbe` for crash detection,
- `readinessProbe` for rollout gating.

This ensures:

- rolling updates via Keel do not interrupt users,
- broken Pods are removed from service,
- startup ordering is enforced where required.

4.9 Summary

This architecture replicates a real-world Kubernetes environment on bare metal by adding the cloud-like functionality that CloudLab does not provide natively: MetalLB for load balancing, ExternalDNS for dynamic public DNS, Rook-Ceph for persistent storage, and ingress routing for multi-service exposure.

The result is a fully automated, reproducible, production-style cluster suitable for deploying Campus Connect and future extensions such as the RAG microservice.

5 Persistence Architecture

Persistent data in Campus Connect is backed by a full Rook–Ceph storage cluster running on the raw hardware provisioned through CloudLab. The system uses Ceph RBD volumes for StatefulSets, CephFS for optional shared filesystem workloads, and dynamically provisioned PersistentVolumeClaims (PVCs) for all stateful services. This section describes the storage backend, configuration, and per-service storage design.

5.1 Rook–Ceph Storage Backend

Rook–Ceph provides a distributed, fault-tolerant storage substrate suitable for databases, caches with persistence, object storage layers, and future embedding stores.

The storage subsystem is deployed in two stages:

1. **Rook-Ceph operator**: installed via the official Helm chart to manage CRDs and controllers.
2. **Custom Ceph cluster chart**: configures monitors, OSDs, block pools, and filesystems.

This arrangement mirrors production-ready clusters where application teams depend on a dedicated storage layer with dynamic PVC provisioning.

5.2 Ceph Cluster Layout

Each CloudLab node includes a secondary disk (`/dev/sdb`), unused by the OS. Rook consumes this disk as a Ceph OSD.

Core Ceph Components

- **Ceph MONs**: one monitor per node to maintain strong quorum.
- **Ceph MGR**: provides cluster telemetry and orchestrates OSD operations.
- **OSDs**: one per node, each consuming `/dev/sdb`.
- **RBD Block Pool**: replicated with size = 3 for fault tolerance.
- **CephFS Filesystem**: includes a metadata pool and a replicated data pool.

This topology ensures:

- even distribution of data and recovery load,
- durability across node failures (within the limits of a 3-node cluster),
- dynamic PVC provisioning for all Kubernetes StatefulSets.

5.3 StorageClasses

Two StorageClasses are created as part of the Ceph deployment:

- **`rook-ceph-block`** — default RBD-backed block volumes.
- **`rook-cephfs`** — POSIX-compliant CephFS filesystem volumes.

Campus Connect uses the RBD class exclusively for StatefulSets, due to:

- strong durability guarantees,
- high performance for database workloads,
- fault tolerance through replication.

Every StatefulSet in the system declares a `volumeClaimTemplate` that references `rook-ceph-block`, enabling fully dynamic provisioning.

5.4 Persistent Volume Usage by Service

The table below summarizes all persistent data components and their corresponding volumes.

MongoDB — 20 Gi RBD PVC

- Mounted at `/data/db`.
- Stores user profiles, classes, channels, and message documents.
- Single-replica StatefulSet to ensure consistent write semantics.

Redis (AOF Persistence) — 10 Gi RBD PVC

- Mounted at `/data`.
- Append-only file (AOF) enabled to protect session and cache integrity.
- Stores:
  - session tokens,
  - WebSocket presence and group membership,
  - rate-limits and ephemeral caches with optional persistence.

MinIO — 50 Gi RBD PVC

- Mounted at `/data`.
- Stores all uploaded files, including:
  - profile pictures,
  - class attachments,
  - temporary objects used for signed URL workflows.
- Buckets initialized via an in-cluster Job using the MinIO client (mc).
- AMQP event hooks send MinIO events to RabbitMQ:
  - currently used for profile image update workflow,
  - designed for future RAG preprocessing triggers.

RabbitMQ — 2–5 Gi RBD PVC (Optional)

- Used if durable queues are enabled.
- Stores on-disk message journals and queue state.
- Facilitates:
  - backend event fanout,
  - MinIO event consumption,
  - future request/response inference workloads.

ChromaDB — 10–20 Gi RBD PVC (Future)

- Dedicated volume for persistent embedding storage.
- Intended for the full RAG microservice integration.
- Will store:
  - document embeddings,
  - search indexes,
  - vector metadata.

5.5 PVC Templates

Each StatefulSet uses a `volumeClaimTemplate` describing its storage requirements.

Example: MongoDB:

Listing 2: MongoDB PVC template

```yaml
volumeClaimTemplates:
  - metadata:
      name: data
    spec:
      accessModes: ["ReadWriteOnce"]
      storageClassName: rook-ceph-block
      resources:
        requests:
          storage: 20Gi
```

This structure ensures that every deployment of Campus Connect receives a fresh, independent set of Ceph-backed volumes without manual provisioning.

5.6 Reclaim Policies and Data Safety

All Rook–Ceph StorageClasses used by the project are configured with:

- **reclaimPolicy**: **Retain**

This ensures that PersistentVolumes are never deleted automatically, even if:

- a StatefulSet is removed,
- a PVC is accidentally deleted,
- the cluster is reinstalled.

This is essential for academic settings, where:

- student files must not be lost during upgrades,
- debugging often involves redeploying Helm charts,
- CI/CD may roll out multiple backend updates.

Retaining the underlying RBD images provides strong safety guarantees while preserving operational flexibility.

5.7 Summary

Campus Connect uses a production-grade persistence strategy built around Rook–Ceph, enabling durable, scalable, dynamically provisioned storage for all stateful services. Databases, caches, object storage, queueing systems, and future RAG services all operate on top of Ceph RBD volumes, ensuring consistent data integrity across experiment resets and cluster lifecycle changes.

6 CI/CD Pipeline Architecture

The Campus Connect deployment workflow is fully automated end-to-end using GitHub Actions for continuous integration, Keel for in-cluster continuous deployment, and the custom Terraform provider for CloudLab to provision the underlying experiment hardware. The pipeline ensures that every experiment can be reproduced deterministically: cluster creation, bootstrap, workload deployment, and rolling updates all occur without manual intervention.

6.1 Repository Structure

The CI/CD pipeline coordinates four GitHub repositories:

- **cc2-backend** — NestJS service (REST + WebSocket). Builds and publishes a backend image to GHCR.
- **cc2-frontend** — React/Vite web client. Produces a static distribution image served by the frontend Pod.
- **cc2-llm** — Standalone microservice providing inference endpoints. Built and pushed like other services, but not yet wired into the main application.
- **cc2-cluster** — Infrastructure repository. Contains:
  - Terraform configuration using the CloudLab provider,
  - Ansible bootstrap playbooks,
  - Helm charts for backend, frontend, MinIO, MongoDB, Redis, RabbitMQ,
  - manifests for Ingress, ExternalDNS, and MetalLB.

Each repository emits its own image version into GHCR under a unique, immutable tag format. Keel monitors these tags and updates the running workloads automatically.

6.2 Pipeline Overview

Figure 3 shows the high-level flow of build, publish, and in-cluster deployment.

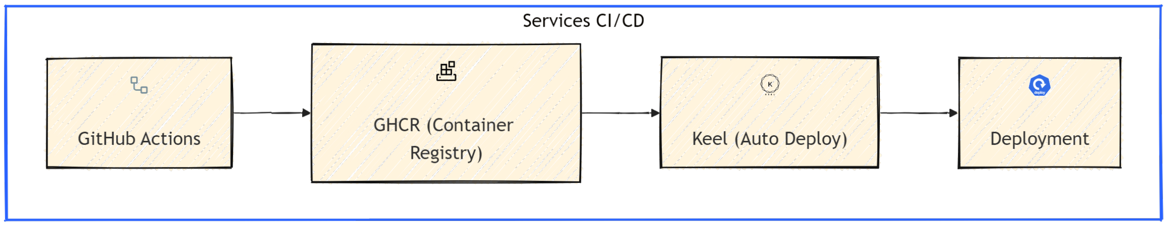

Figure 3: Logical CI/CD flow from services to the running cluster.

The workflow is driven by three stages:

1. **Build** (per-service repos) GitHub Actions build container images for backend, frontend, and LLM.
2. **Publish** Images are pushed to GHCR with versioned and `:latest` tags.
3. **Deploy**
   - The `cc2-cluster` pipeline provisions hardware, deploys Kubernetes, and installs Helm charts.
   - Keel watches GHCR and triggers rolling updates.

6.3 CloudLab Infrastructure Provisioning

All hardware allocation and low-level bootstrap are performed through the custom Terraform provider for CloudLab. Rather than writing Python scripts or manually interacting with the CloudLab GUI, Terraform produces:

- one **kubeadm control plane** node,
- two **kubeadm workers**,
- preconfigured network interfaces for MetalLB,
- attached disks on `/dev/sdb` for Rook–Ceph OSDs.

A GitHub Actions workflow in `cc2-cluster` performs:

1. **terraform init/plan/apply** using the CloudLab provider,
2. SSH key injection (via ephemeral GitHub-generated keys),
3. publication of node IPs as job artifacts for follow-up stages.

Figure 4 shows this stage of the pipeline.

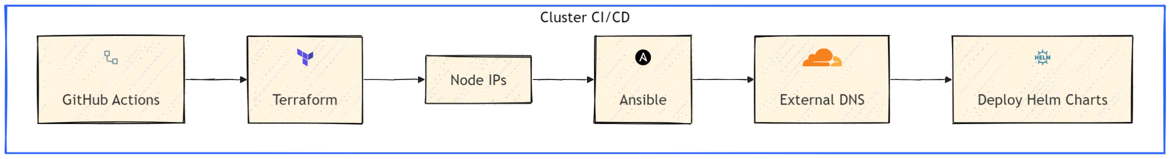

Figure 4: Cluster provisioning stage using the Terraform CloudLab provider.

6.4 Bootstrap Phase (Ansible)

Once Terraform exposes reachable IPs, an Ansible playbook executes the complete bootstrap sequence:

- install container runtime (Docker),
- run `kubeadm init` on the control plane,
- join workers to the cluster,
- install CNI (Flannel),
- deploy Rook–Ceph operator,
- deploy custom Ceph cluster and StorageClasses,
- install MetalLB with the CloudLab-assigned address pool,
- install ExternalDNS and register DNS records through Cloudflare.

The end result is a fully functional cluster with ingress, DNS, and storage before any application workloads are installed.

6.5 Application Deployment (Helm)

After bootstrap, the pipeline installs all application components using Helm charts stored in the `cc2-cluster` repository:

- Redis (AOF persistence)
- MongoDB
- MinIO + bucket initialization Job
- RabbitMQ
- Backend (NestJS)
- Frontend (React/Vite)
- LLM service (standalone, not yet connected)
- Ingress rules and TLS configuration

Each chart uses RBD-backed PVCs provisioned dynamically by Ceph.

6.6 Keel Automatic Rollouts

Keel runs inside the cluster as a controller monitoring GHCR image tags referenced in Deployment and StatefulSet manifests.

When the backend or frontend pushes a new image to GHCR:

1. Keel receives a webhook or performs scheduled polling.
2. It compares the running digest with the available digest.
3. If newer, Keel performs:
    - rolling Deployment update for stateless services,
    - in-place StatefulSet restart for stateful workloads.

This enables rapid iteration during development without re-running the entire cluster deployment pipeline.

6.7 Secrets and Credential Handling

GitHub Actions uses encrypted repository secrets to provide:

- **CloudLab PEM certificate** for authenticating Terraform API calls,
- **Cloudflare API token** for ExternalDNS management,
- **GHCR PAT** for pushing container images,
- **Ansible Vault password** for decrypting sensitive bootstrap tasks.

No sensitive values are baked into Terraform, Helm, or application code.

6.8 End-to-End Workflow Summary

Figure 5 summarizes the coordination of all pipeline stages.

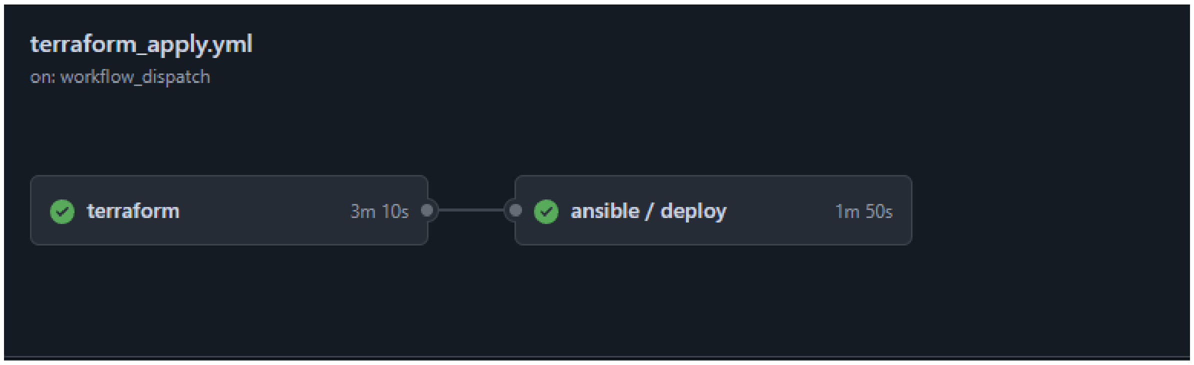

Figure 5: End-to-end CI/CD workflow across Terraform, Ansible, Helm, and Keel.

This architecture provides:

- deterministic cluster creation,
- reproducible deployments for grading and experimentation,
- safe rolling updates without downtime,
- complete automation from code commit to running container.

The system is designed so that a single GitHub Action run can:

1. allocate CloudLab hardware,
2. deploy Kubernetes and Rook–Ceph,
3. install ingress, DNS, and monitoring,
4. deploy every microservice,
5. roll out future image updates automatically.

7 Team Charter

Roles

- **Edwin Biswas — Project Lead / PM**: Oversees schedule, deliverables, and coordination.
- **Alex Cooper — Backend/Inference Engineer**: Develops APIs and integrates RAG inference.
- **Tyler Geiger — DevOps Engineer**: Manages CI/CD, Terraform, Helm, cluster lifecycle, and deployment tooling.
- **Kadin Matotek — QA & Documentation**: Designs testing scenarios, validates system behavior, and maintains documentation consistency.

Definition of Done (DoD)

A feature is considered complete when:

- Code passes the CI/CD pipeline without errors.
- The resulting Pod deploys successfully in the Kubernetes cluster.
- Documentation, diagrams, API specifications, and repository notes are updated.
- QA verifies correct behavior under expected and edge-case conditions.

8 Conclusion

Campus Connect demonstrates a fully realized, cloud-native architecture built on top of raw CloudLab hardware. The platform integrates Kubernetes orchestration, Ingress routing, distributed object and block storage via Rook–Ceph, observability through Prometheus and Grafana, and a reproducible CI/CD pipeline that automates the entire deployment lifecycle. The system is architected using the same principles found in modern production environments: declarative infrastructure, immutable container builds, automated rollout mechanisms, and stateless microservices supported by persistent backing stores.

A key outcome of this work is the creation of a standalone Terraform provider for CloudLab. Although used here to automate Campus Connect deployments, the provider is designed as an independent project: general-purpose, reproducible, and usable by any CloudLab researcher or student without modification. This significantly improves the tooling available for teaching, academic research, and infrastructure experimentation.

The project establishes a foundation on which additional features—such as full RAG integration, distributed inference workers, expanded monitoring, and multi-node deployments—can be added without architectural changes. The system's design emphasizes clarity, maintainability, and extensibility, ensuring that future teams or researchers can build upon this work as CloudLab and classroom requirements evolve.

Team Resumes

EDWIN PRITOM BISWAS

Brookhaven, PA 19015

6102569965

Edo30025@gmail.com

PROFESSIONAL SUMMARY

Profile/Professional Summary: Diligent worker with eight years of experience in the fast-food industry, seeking new and exciting opportunities. Works well both in groups and independently.
Adaptive under pressure, good at problem-solving, and interested in learning new things.

WORK HISTORY

Crew Member/Store Manager, 02/2017 - Current
Dunkin Donuts, 2705 Edgemont Avenue, Brookhaven, PA 19015
Dunkin Donuts, 5101 Pennell Rd, Media, PA 19063

- Provided excellent customer service by greeting customers and meeting quality expectations.
- Took orders, prepared meals, and collected payments.
- Kept food preparation area, equipment, and utensils clean and sanitary.
- Worked front counter, drive-thru, and other areas.
- Assisted other team members in achieving goals.

WEBSITES, PORTFOLIOS, PROFILES

- https://www.linkedin.com /in/edwi n-biswas-

SKILLS

- Proficient in Customer Service.
- Well-versed in Individual and Mass communication.
- Experienced in Information management and modern technology.
- Experience with Windows in File Management and Networking.
- Documentation using Adobe and Microsoft Word, Excel, and PowerPoint.
- Experienced in managing money as a manager during the pandemic of 2020 and at current times.
- Product Promotion
- Time Management
- Workplace Efficiency

EDUCATION

Bachelor of Science, Computer Science, Expected in 12/2025

West Chester University of Pennsylvania - 700 S High St, West Chester, PA 19383

GPA: 3.47

Associate degree, Computer Science, 12/2021

Delaware County Community College - Marple Campus 901 S. Media Line Road Media, PA 19063

GPA: 3.79

Alex M. Cooper Jr.

U.S. Citizen | cooperjralex@gmail.com | (717) 693-9424 | Github: https://github.com/AlexCooperJr

Education

West Chester University of Pennsylvania **West Chester, Pennsylvania**
B.S. in Computer Science Expected Graduation: Spring 2026

- **Related Coursework:** Computer Science III, Calculus, Discrete Mathematics, Statistics II, Data Structures and Algorithms, Computer Security, Computer Systems, Cloud Computing
- Placed second in the 2025 PACISE Cyber Security Competition
- Participated in the 7th and 10th Annual West Chester Programming Contest
- Spring 2024 Dean's List, Fall 2024 Dean's List, Spring 2025 Dean's List

STEM Work Experience

Booz Allen Hamilton **Annapolis Junction, Maryland**
Software Engineering Intern (Data Science) June 2025-August 2025

- Implemented an AI model to perform knowledge graph(KG) completion methods to output inferences about entities and their relationships from text data, alongside other metrics from the graph.
- Developed a user interface(UI) to streamline the implemented processes of KG construction, modification, visualization, and completion.
- Utilize containerization best practices to make a complete deployable product for cloud or other devices. Containized the backend separate from the front for utilization of the model without the established UI.
- Presented research and implemented results to a group of interns, full time employees, leadership, and executives on a national stage.

Statistics Tutor **West Chester, Pennsylvania**
Peer Tutor August 2024-Current

- Assisted students in either one on one or small group (less than 4) settings to work through topics from the Statistics I course at the West Chester University Learning Assistance and Resource Center (LARC)
- Covered topics including probability, confidence intervals, tests of hypothesis', and regression
- Reviewed material and developed individualized plans to ensure each student was able learn effectively and communicated between faculty, students, and peers to ensure continuity.
- Participated in training with the College Reading and Learning Association to be certified.

Projects

Embedded Retrieval Augmented Generation (RAG) System **Lancaster, Pennsylvania**
August 2025

- Utilized an embedding model and vector database to handle the conversion of text data from different pipelines into vectors then storage for future retrieval

- Leveraged prompt engineering to create a prompt which allowed a Large Language Model to answer questions with a higher accuracy and more relevant information by retrieving relevant documents from the database.
- Developed a simple user interface to allow for a non command line way to prompt the LLM and receive answers

Thermostat Project **Lancaster, Pennsylvania**
July 2024

- Programmed an Arduino UNO to take in input from a thermistor and display the temperature onto an Liquid Crystal Display(LCD), change a Tri-color LED to a corresponding color based on range, and start a DC motor depending on the registered temperature.
- Utilized the Arduino IDE to write code for communication between the arduino and components.
- Designed the circuit and implemented the appropriate techniques and components to regulate current outputs for protection of all components.

Technical Skills

Languages: Python, Java, HTML, JavaScript, CSS, C, C++, Haskell, React
Libraries: Pandas, Numpy, Matplotlib, scikit-learn, Java.util, next.JS, node.JS, networkX
Environments/Tools: Thonny, BlockPy, jGRASP, Visual Studio Code, GitHub,Arduino IDE,CloudLab, Docker, fastAPI, wireshark

Non-Technical Work Experience

Valvoline Instant Oil Change **Lancaster, Pennsylvania**
Senior Technician May 2022-August 2024

- Supervised a team of employees to ensure speed of service, quality work, and safety.
- Interacted with suppliers to ensure supplies and resources were properly managed and available.
- Used a variety of new technologies, and industry standards to perform a number of roles during vehicle servicing.
- Kept track of and issued disciplinary files and ensured accountability for team members.

Resident Assistant Internship and Leadership Exploration(RAILE) **Newark, Delaware**
Intern/Mentee February 2023-May 2023

- Attended a variety of professional workshops that addressed issues commonly seen in work places and resident halls.
- Received on-site training and met with a mentor to build and develop characteristics of a leader.
- Helped coordinate and host events to engage with members of the community.

Tyler Geiger

West Chester, PA | tygg513@outlook.com | 267-500-7205 | linkedin.com/in/tyler-geiger

github.com/TylerGeiger513

Profile

Cloud-focused software engineer with experience deploying applications and infrastructure on AWS, Azure, and research-grade platforms. Skilled in Kubernetes, Terraform, and CI/CD automation, with a focus on building reproducible, scalable environments. Experienced in Agile teams, taking ownership of technical challenges, and contributing to both enterprise and academic projects.

Education

West Chester University of Pennsylvania, BS in Computer Science, Minor in Applied Statistics, Certificate in Cloud Engineering — Sept 2022 – May 2026

- GPA: 3.52/4.0
- **Deans List:** Spring 2024, Fall 2024, and Spring 2025
- **Relevant Coursework:** Cloud Computing I & II, Software Engineering, Operating Systems, Modern Web Applications, Data Communications & Networking, Database Management Systems, Computer Security & Ethics, Programming Languages & Paradigms, Data Structures & Algorithms, Experimental Design, Applied Statistics

Experience

Software Developer Intern, iPipeline – King of Prussia, PA — June 2025 – Aug 2025

- Migrated six carriers from Jenkins to GitHub Actions by upgrading frameworks, modernizing package references, and validating builds.
- Developed scripts to automate NuGet updates and accelerate local workspace setup and testing with parallelization.
- Transitioned multiple proprietary web services from TRX servers to AWS Lambda Functions and contributed to Terraform-based deployments for consistent Sandbox, QA, UAT, and Production environments.
- Worked within an Agile team, collaborating across roles to troubleshoot issues, implement requested features, and ensure reliable delivery for clients.

Copyright Office – Temporary Employee, ASTM International – Conshohocken, PA — Dec 2022 – Present

- Successfully registered over 5000 of ASTM's published literary works with the U.S Copyright Office through the ECO system.
- Developed an automation for the process above using Node.js, the XLSX library, and a Copyright Public Records API. This reduced a previously week-long, sometimes multi-week manual process into an automated process that can be configured and executed in a single day, significantly improving efficiency.
- Developed numerous ease-of-use automations and shortcuts utilizing Excel Office scripts and Windows PowerShell, significantly improving workflow efficiency, and simplifying tasks for my supervisor.

IT Operations – Temporary Employee, ASTM International – Conshohocken, PA — June 2023 – Aug 2023

- Populated the Application Portfolio Management (APM) system by conducting interviews with IT team members and documenting application dependencies, subject matter experts, and infrastructure components across DEV, QA, and Production environments.
- Standardized and organized metadata for dozens of enterprise applications to improve visibility, support planning, and enhance future maintenance efforts.

Projects

Campus Connect – Full-Stack Kubernetes Application on Research Cloud Infrastructure — `github.com/campusconnectwcu/cluster`

- Collaborated with a team to design and deploy a full-stack academic communication platform using NestJS and

MongoDB, containerized with Docker and orchestrated via Kubernetes and Helm.

- Configured a production-grade environment on the CloudLab research testbed, implementing DNS, SSL certificate management, and NGINX ingress routing for secure, scalable access.
- Automated image builds, designed repeatable infrastructure, and cluster deployments through GitHub Actions, Docker registry workflows, and Keel for continuous delivery and image version tracking.
- Engineered Helm charts, Kubernetes secrets management, and repository split strategies to optimize CI/CD pipelines and development workflows.
- Authored documentation to ensure reproducibility, scalability, and future academic research reference.
- **Tools Used:** NestJS, MongoDB, React, Redis, Docker, Kubernetes, Helm, Keel, GitHub Actions, Terraform, Skaffold, NGINX, CloudLab.

Portalctl – Go CLI for CloudLab/Emulab XML-RPC (WIP)

`github.com/CSC478-WCU/portalctl`

- Implemented a Go command-line utility to manage CloudLab experiments via Emulab XML-RPC, covering full lifecycle operations: `start`, `status`, `modify`, `terminate`, `extend`, `manifests`, `reboot`, `connect`, and `disconnect`.
- Added reproducible parameterization with `-bindings`/`-bindings-file` and repeatable `-param k=v` flags, plus JSON status output (`-j`) and configurable timeouts for CI usage.
- Built TLS client-auth support (PEM cert/key) and cross-platform builds; documented usage and examples for fast onboarding.
- Designed to pair with Terraform/CI pipelines for research testbed deployments (e.g., parameterized profile launches, profile modification, extends, node reboots).
- **Tools Used:** Go, XML-RPC, TLS/PEM, CloudLab/Emulab.

Web Development Volunteer, Chester County Association for the Blind and Visually Impaired (CCABVI) Sept 2024 – Dec 2024

- Collaborated wtih a team on a website accessibility enhancement project, implementing UI/UX improvements and features designed to meet ADA Section 508 compliance standards for blind and visually impaired users.
- Diagnosed and resolved technical issues, including a broken PayPal donation integration and misconfigured WordPress hosting setup.
- Took a leadership role in front-end development and coordinated team communication to ensure timely delivery of improvements.

Technologies

Languages: JavaScript/TypeScript, Go, Python, SQL, Bash, PowerShell

Cloud & Infrastructure: AWS (Lambda, ParamStore, CloudWatch), Azure (CLI, AKS, ACR), Kubernetes, Docker, Helm, Skaffold, NGINX, Keel, CloudLab/Emulab

DevOps & IaC: Terraform, Terragrunt, GitHub Actions, Octopus Deploy, CI/CD Pipelines, Secrets Management

Frameworks & Tools: .NET, NestJS, Node.js, Postman, Jest, MySQL, MongoDB, Git/Github

Specialties: Infrastructure-as-Code, Automation Scripting, API Development, Accessibility/ADA 508 Compliance, Cloud-Native Deployment Strategies

Kadin Matotek

302-723-2182 | kadinmatcs@gmail.com | linkedin.com/in/kadin-matotek | github.com/kmatotek

Education

Strath Haven High School Wallingford, PA
High School Diploma *Graduation: 5/2022*

- Clubs and Organizations: Ice Hockey, Track and Field, and Soccer.
- Relevant Coursework: Computer Science Principles

West Chester University of Pennsylvania West Chester, PA
B.S. in Computer Science, Minor in Mathematics *Expected Graduation: 5/2026*

- GPA: 3.94/4.0
- Clubs and Organizations: Upsilon Pi Epsilon, Omicron Delta Kappa, Club Ice Hockey
- Relevant Coursework: Artificial Intelligence, Data Structures & Algorithms, Operating Systems, Computer Security, Software Engineering, Cloud Computing, Applied Statistics, Linear Algebra, Discrete Mathematics, Multivariable Calculus

Projects

Email Classifier | *Java* Oct 2023

- Achieved 89% prediction accuracy in distinguishing legitimate emails from spam across a dataset of 5,000 emails by calculating distances between individual emails and clusters.
- Implemented Java GUI enabling users to assess email authenticity and calculate distances between individual emails and/or email groups.
- Integrated a range of classification techniques, including nearest neighbors and Euclidean distance calculations, to improve adaptability and strengthen analytical precision.

Happy Programming Language | *Java* Nov 2024

- Designed and developed "Happy", a custom programming language featuring support for variable assignments, conditional expressions, loops, functions, list operations, and string formatting, aimed at providing a clean and concise syntax compared to other common programming languages.
- Documented the development and syntax through knowledge base documentation, allowing for anyone to learn the syntax, built in functions, and everything needed to know to start developing with this language.

2D Unity Game | *C#, Unity* Oct 2023

- Developed a 2D video game inspired by *The World's Hardest Game*, incorporating challenging level design and precise player controls.
- Designed and implemented multiple levels with increasing difficulty to test players' reflexes and problem-solving skills.
- Conducted play-testing to refine game mechanics and enhance user experience based on feedback.

3D Unity Game | *C#, Unity* Nov 2023

- Developed a 3D game incorporating custom graphics, custom player scripts, and realistic physics simulations in outer space.
- Developed multiple levels, presenting players progressively challenging obstacles to overcome, ensuring continuous skill progression.
- Conducted user testing to get feedback and implemented improvements through updates.

Facial Recognition | *Python* Feb 2024

- Engineered a real-time face recognition program achieving over 90% accuracy on a structured dataset, providing efficient visual identification.
- Integrated advanced facial detection and recognition algorithms, utilizing libraries to detect and recognize faces in diverse environments.

Experience

Research Assistant Feb 2025 – Present
West Chester University of Pennsylvania *West Chester, PA*

- Awarded $2,500 research grant as the only Computer Science student accepted into SURI
- First author of an AI research paper submitted to CCSCE, rated highest among all conference submissions
- Built multi-model pipelines using Ollama and Hugging Face, leveraging reprompting and chain-of-thought techniques to enhance AI model performance
- Leveraged the ACCESS supercomputing center to run large-scale jobs efficiently, enabling rigorous testing of modern LLMs beyond typical hardware constraints
- Collaborated with Dr. Ngo on research initiatives aimed at evaluating AI model performance and optimizing testing methodologies

D.P. Dough West Chester Mar - Aug 2023
Delivery Driver *West Chester, PA*

- Executed over 1,000 customer deliveries with a strong focus on optimizing delivery routes, resolving logistical challenges, and ensuring timely, accurate order fulfillment through advanced problem-solving techniques.
- Utilized problem-solving skills to streamline order management processes and provide customer service.
- Took orders, served customers, addressed inquiries, resolved any issues to ensure the best customer experience.

Vicky's Place Jun - Jul 2021
Dish Washer *Swarthmore, PA*

- Assisted kitchen staff by ensuring a steady supply of clean cookware and dishware during peak hours.
- Maintained a fast-paced and organized dishwashing station to support smooth kitchen operations.
- Trained new employees on proper dishwashing procedures, safety protocols, and kitchen cleanliness standards to ensure consistency and efficiency.

Fort Delco Gym Jun - Jul 2020
Gym Custodian *Morton, PA*

- Maintained cleanliness and sanitation of workout areas, locker rooms, and equipment to ensure a safe and hygienic environment for both members and staff.
- Conducted routine inspections to identify and address maintenance or cleanliness issues promptly.
- Collaborated with staff to uphold gym standards and provide a positive experience for members.

Open Sky Energy Aug - Nov 2019
Social Media Manager *Swarthmore, PA*

- Collaborated with the team to develop marketing strategies and improve social media outreach.
- Developed and scheduled posts to promote solar energy solutions, company updates, and customer success stories.
- Managed and created content for the company's social media platforms, increasing engagement and brand visibility.

Volunteering

Media Food Bank Mar - Aug 2023
Volunteer *Media, PA*

- Assisted customers by taking food orders and providing a welcoming, supportive environment.
- Retrieved and organized food items based on customer orders to ensure accuracy and efficiency.
- Worked collaboratively with volunteers and staff to maintain smooth food distribution operations.

Publications

K. Matotek, H. Cassel, M. Amiruzzaman, L. B. Ngo. *Evaluating the Limitations of Local LLMs in Solving Complex Programming Challenges.* arXiv preprint, 2025. arxiv.org/abs/2509.15283

Technical Skills

Proficient Languages: Java, Python
Familiar Languages: JavaScript, C#, R, SQL, Haskell, LaTeX, HTML, CSS
Developer Tools: Docker, Git, Linux, Zsh, Redis, MongoDB, Spring Boot, Unity, Ollama, Hugging Face