MongoDB AI Arena - Application Architecture Overview

Based on my exploration of the codebase, here's how the different components fit together:

High-Level Purpose

This is a cloud-based training platform that helps users learn MongoDB through hands-on exercises. It's themed around building an Airbnb-like short-term rental application where participants complete coding challenges, compete on a leaderboard, and learn MongoDB concepts like CRUD operations, aggregations, search, and vector search.

Component Breakdown

1. Infrastructure Layer (`utils/`)

This contains all the scripts and configuration for spinning up the cloud environment:

`utils/arena-terragrunt/` - Main Orchestration

- Purpose: Terraform/Terragrunt configuration to deploy the entire infrastructure
- Key Files:
  - `config.yaml` - Configuration per customer (MongoDB credentials, AWS settings, domain)
  - `airbnb/` - Template directory that gets duplicated per customer/event
  - `scenario.json` - Defines which exercises/modules to include for this event
- What it creates: AWS EKS cluster, MongoDB Atlas cluster, networking, SSL certificates

`utils/atlas-cluster/` - MongoDB Setup

- Purpose: Creates MongoDB Atlas cluster and populates it with sample data
- Creates:
  - Atlas cluster with M30+ tier
  - Database users for participants (from `user_list.csv`)
  - Indexes for CRUD, Search, and Vector Search exercises
  - Loads sample Airbnb dataset

`utils/eks-cluster/` - Kubernetes Resources

- Purpose: Deploys all Kubernetes services to AWS EKS
- Deploys:
  - VSCode Online (`mdb-openvscode/`) - Browser-based IDE for participants
  - Arena Portal (`mongodb-arena-portal/`) - Landing page to claim credentials
  - Documentation (`docs-nginx/`) - Hosts the Jekyll docs site
  - LiteLLM (`litellm/`) - AI proxy for Claude/GPT integration
  - Results Processor (`results-processor/`) - Validates exercise submissions
  - Nginx - Reverse proxy and load balancer
  - Redis - Caching for LLM requests

---

2. Documentation Layer (`docs/`)

This is the instructional content participants see:

- Jekyll-based static site deployed to the EKS cluster
- Structure:
  - `_pages/crud/` - CRUD operation exercises (8 exercises)
  - `_pages/pipeline/` - Aggregation pipeline exercises
  - `_pages/search/` - Atlas Search exercises
  - `_pages/vector-search/` - Vector search with AI chatbot
  - `_pages/vibe-coding/` - AI-assisted coding exercises
  - `_pages/guided/` - Guided workshop instructions
- Hosted at: `https://{customer}.mongoarena.com` (configured via nginx)

---

3. Sample Application (The "Airbnb Clone")

This is the working application participants modify during exercises:

`app/` - Frontend (Next.js)

- Purpose: User-facing Airbnb-like rental search interface
- Tech: Next.js 15, React 19, TailwindCSS
- Features:
  - Property listings with filters (amenities, beds, property type)
  - Map view (Leaflet/React-Leaflet)

- Search functionality
- Host analytics dashboard
- Exercise status indicators (shows which exercises are complete)
- Chatbot interface (vector search exercise)

- Runs on: Port 3000 in participant's VSCode Online environment

`server/` - Backend (Node.js/Express)

- Purpose: API server that participants implement exercises in
- Tech: Express.js, MongoDB Node Driver, LangChain
- Key Routes:
  - `/api/listingsAndReviews` - Property CRUD operations
  - `/api/results` - Exercise validation and leaderboard
  - `/api/chat` - AI chatbot (vector search)
- Exercise Files:
  - `server/src/lab/*.lab.js` - Incomplete exercise files participants edit
  - `utils/answers/*.lab.js` - Solution files for validation
- Testing: Mocha tests validate implementations and submit scores

---

4. Arena Portal (`utils/eks-cluster/mongodb-arena-portal/`)

This is the participant onboarding system:

- Frontend (Next.js/React/TypeScript): Landing page where participants claim their credentials
- Backend (Flask/Python):
  - Manages participant database
  - Tracks which credentials are "taken"
  - Shows available slots and leaderboard
- Database: Uses MongoDB to track participants
- Hosted at: Root domain (e.g., `https://airbnb.mongoarena.com`)

---

5. Testing & Validation System

The automatic exercise grading system:

- `server/src/saveTestResults.js`: Runs Mocha tests and posts results
- Results Processor (Java/Spring Boot in EKS):
  - Receives test results

    - Validates against answer key
    - Updates leaderboard
    - Tracks timing for "timed" mode or scores for "score" mode
- Exercise Status Component (`app/src/components/ExerciseStatus.js`): Shows green/red indicators in the frontend

---

Data Flow

```
1. SA runs `terragrunt apply`
   → Creates AWS EKS + Atlas Cluster
   → Deploys all services (VSCode, Portal, Docs, etc.)
2. Participant visits {customer}.mongoarena.com
   → Claims credentials from Arena Portal
   → Gets username/password for VSCode Online
3. Participant opens VSCode Online
   → Pre-loaded with app + server code
   → Follows documentation site for exercises
4. Participant edits server/src/lab/*.lab.js files
   → Frontend app at localhost:3000 shows results
   → Runs npm test to validate solution
5. Test results → Results Processor
   → Validates against answer key
   → Updates leaderboard
   → Shows green checkmark in frontend
```

Directory Purpose Summary

Directory	Purpose	Who Uses It
`utils/arena-terragrunt/`	Infrastructure setup scripts	MongoDB SAs only
`utils/atlas-cluster/`	MongoDB cluster config	MongoDB SAs only
`utils/eks-cluster/`	Kubernetes deployments	MongoDB SAs only
`docs/`	Exercise instructions (Jekyll)	Participants view this
`app/`	Frontend sample app	Participants see/use this
`server/`	Backend API (exercise files)	Participants edit this
`utils/answers/`	Solution keys	Validation only
`utils/eks-cluster/mongodb-arena-portal/`	Credential management portal	Participants claim credentials

---

Key Technologies

- Cloud: AWS EKS (Kubernetes), Route53 (DNS), S3 (Terraform state)
- Database: MongoDB Atlas (M30+)
- IaC: Terraform + Terragrunt
- Container Orchestration: Kubernetes + Helm charts
- Frontend: Next.js 15, React 19, TailwindCSS
- Backend: Node.js/Express, Python/Flask, Java/Spring Boot
- AI/LLM: LiteLLM proxy, LangChain, Claude/GPT

- Documentation: Jekyll (GitHub Pages style)
- IDE: VSCode in browser (OpenVSCode Server)