**1 — SafeSight AI: Real-time hazard detection & incident reporting platform**

**One-sentence description:** A cloud + edge system that detects hazards (slips, spills, equipment faults) from live camera feeds, notifies stakeholders automatically, and provides an incident dashboard for analytics and compliance.  
**Real-world use case:** Facilities (warehouses, retail stores, hospitals) can reduce incident response time and legal exposure by automatically detecting hazards and creating auditable incident reports with photos, video clips, location, and remediation workflow.  
**Full tech stack (MVP):**

* Frontend: React + TypeScript, Vite, Tailwind CSS
* Mobile: React Native or Progressive Web App (PWA) for quick on-floor reporting
* Backend: Node.js (NestJS or Express) or Go (Gin)
* Real-time: WebSockets (Socket.IO) or WebRTC for live video + signaling
* Storage: PostgreSQL (metadata), S3 (video/images), Redis (caching, rate limiting)
* Messaging: Kafka or RabbitMQ for event processing
* ML: PyTorch/TensorFlow models, Hugging Face/ONNX for inference; Triton Inference Server for scaling; optional edge model (TensorRT or ONNX) for camera devices
* Infra: Docker, Kubernetes (EKS/GKE), Terraform, GitHub Actions for CI/CD
* Observability: Prometheus + Grafana, ELK or Loki for logs
* Optional: Edge device using Jetson/NVidia or Raspberry Pi with accelerated inference  
  **Advanced features that show technical depth**
* **Edge + Cloud hybrid inference:** Run a lightweight detector on the edge for low latency and send clips to cloud for multi-frame, higher-accuracy re-analysis (model orchestration with fallback). Demonstrates device management, model optimization, and inference scaling.
* **Automated incident workflows:** Multi-step automated workflow (detect → validate → notify → escalate → close) with role-based approvals and webhooks to integrate Slack/Teams/IMS. Build using an event-sourcing pattern and durable workflow engine (e.g., Temporal, or self-built state machine).
* **Real-time streaming + clip extraction:** Low-latency WebRTC for live view, plus server-side logic to extract short video clips around incidents and store them with timestamps and bounding boxes.
* **Explainable detection & human-in-the-loop:** Show bounding boxes, classification confidence, and a “why did this flag” panel (feature visualization or Grad-CAM thumbnails) — demonstrates ML interpretability.
* **Analytics & heatmaps:** Spatial/time visualizations (map overlay of incident density, time series of incident rates) using D3 or Apache ECharts.  
  **Resume pitch (1–2 lines):**  
  Built **SafeSight AI**, a cloud+edge incident detection MVP (React/TypeScript, Node.js, Kubernetes, PyTorch + Triton) that reduced manual incident reporting by automating real-time detection, clip extraction, and multi-step remediation workflows — implemented edge inference, event-sourcing workflows (Temporal), and an analytics dashboard for operational insights.

**Roadmap for SafeSight AI (6–8 weeks)**

**Week 1 — Foundations & Setup**

* ✅ Define requirements: hazard detection (slip/spill/fall), incident reporting workflow, dashboard.
* ✅ Choose ML model baseline (e.g., pre-trained YOLOv8 or Hugging Face object detection).
* ✅ Repo scaffolding (frontend, backend, ML service, infra).
* ✅ CI/CD pipeline setup with GitHub Actions (lint/test/build).
* ✅ Containerize services with Docker.

**Deliverable:** Working repo skeleton with Dockerized services, “hello world” React app + backend API, basic model inference script.

**Week 2 — Hazard Detection Prototype (ML service)**

* Implement ML microservice in Python (FastAPI).
* Load pre-trained YOLOv8 model (or SSD/DETR) and run inference on sample images.
* REST endpoint /detect → returns bounding boxes, labels, confidence.
* Store results in PostgreSQL (metadata only).
* Add S3-compatible bucket (e.g., MinIO locally, AWS S3 in prod) for storing uploaded images/video snippets.

**Deliverable:** Upload image → API returns detections + stores result in DB.

**Week 3 — Backend API & Incident Workflow**

* Build Node.js (NestJS/Express) backend with PostgreSQL + Redis.
* Define entities: Incident, Detection, User, WorkflowStep.
* Expose endpoints:
  + POST /incident → create new incident from ML detection.
  + GET /incident/:id → incident details.
  + PATCH /incident/:id/status → update workflow step.
* Integrate background jobs (BullMQ/Redis) for notification + escalation.
* Add RBAC (role-based access: Worker, Manager, Admin).

**Deliverable:** Full incident workflow API with role-based controls.

**Week 4 — Frontend Dashboard**

* React + TypeScript + Tailwind CSS.
* Pages:
  + **Login/Signup** (Auth0/OAuth)
  + **Live Feed** (WebRTC preview or upload demo video)
  + **Incidents List** (sortable table)
  + **Incident Details** (image/video preview, bounding boxes overlay, status workflow).
* Connect to backend APIs.

**Deliverable:** Usable dashboard with incident reporting + status tracking.

**Week 5 — Real-Time Streaming + Notifications**

* Integrate WebSockets (Socket.IO) to push real-time hazard alerts to dashboard.
* Video ingestion: support simulated live stream (FFmpeg → WebRTC or HLS).
* Clip extraction service: auto-cut 10s before/after detection, store in S3.
* Notifications: Slack/Email integration for escalations.

**Deliverable:** Live dashboard auto-updates when new hazards are detected.

**Week 6 — Analytics & Visualization**

* Add Grafana/Prometheus for monitoring (optional for MVP, but good for resume).
* Frontend D3.js charts:
  + Incident counts over time.
  + Heatmap of hazards (aggregate bounding box positions).
* Export incidents CSV/PDF.

**Deliverable:** Analytics dashboard with visual insights.

**Week 7 — Infra & Edge Deployment**

* Deploy ML service to GPU-enabled environment (AWS EC2 G4, or run inference locally with ONNX).
* Container orchestration with Docker Compose → upgrade to Kubernetes (EKS/GKE).
* Set up Terraform for cloud infra (VPC, DB, S3 bucket).
* Optimize model for edge (ONNX/TensorRT on Jetson Nano or local CPU).

**Deliverable:** Running in cloud (K8s) with reproducible infra setup.

**Week 8 — Polish & Demo**

* Add unit + integration tests.
* Record demo video:
  + Show live detection on camera feed.
  + Incident auto-logging + dashboard update.
  + Workflow escalation + analytics.
* Write resume bullet points and LinkedIn post.

**Deliverable:** Polished demo-ready MVP.

**🛠️ Tech Choices Recap**

* **Frontend:** React + TypeScript + Tailwind
* **Backend:** Node.js (NestJS) + PostgreSQL + Redis + BullMQ
* **ML Service:** Python (FastAPI) + YOLOv8/DETR → ONNX/TensorRT optimization
* **Real-time:** WebRTC + Socket.IO
* **Storage:** PostgreSQL (incidents), S3/MinIO (media), Redis (cache/queue)
* **Infra:** Docker, Kubernetes (EKS/GKE), Terraform
* **Notifications:** Slack API + SMTP (email)
* **Analytics:** D3.js (frontend), Prometheus/Grafana (infra)

**Useful Docker commands (beginner-friendly)**

* **Follow logs** (all services): docker compose logs -f
* **Follow logs** (single service): docker compose logs -f backend
* **Run a one-off shell in backend container**:  
  docker compose exec backend sh
* **Run a SQL command inside Postgres**:
* docker compose exec postgres psql -U appuser -d safesight -c "SELECT now();"
* **Rebuild only one service**:  
  docker compose build backend && docker compose up -d backend
* **Stop & remove containers (and networks) but keep volumes**:  
  docker compose down
* **Stop & remove everything including volumes** (will delete DB data):  
  docker compose down -v