Edge Safety Monitor — 8■Week Project Plan & Checklist

One-line: Real-time PPE & unsafe behavior detection (helmet, vest, phone use, drowsiness) → FastAPI inference + React dashboard.

Goal: Build an end-to-end, demo-ready safety-monitor system that runs locally (edge) and shows results on a React dashboard. This PDF gives a week-by-week checklist, estimated hours, and a daily schedule you can follow.

> **Recommended time commitment (pick one): Part-time (student-friendly)**: ~18 hours/week — *2 hours/day (Mon–Fri) + 4 hours/day (Sat–Sun)*. Good balance if you have classes. **Intensive (fast-track)**: ~40 hours/week — *6–8 hours/day (Mon–Fri)*. Finishes faster but requires focused days. **Hybrid option**: 30 hours/week — *4 hours/day (Mon–Fri) + 5 hours/day (Sat)*.

8■Week Plan (Checklist + estimated hours)

■ **Week 0 — Prep (1–2 days)** — (4–6 hours)
- ■ Create repo + README (initial), setup virtual envs
- ■ Gather datasets (helmet/vest/phone/drowsiness) and sample images
- ■ Run YOLOv8 baseline on sample images to confirm environment

■ **Week 1 — Data & Labeling** — (10–15 hours)
- ■ Assemble and clean dataset (combine public + your captures)
- ■ Annotate images or use Roboflow (200–500 images target)
- ■ Add augmentations and domain randomization (lighting, blur, occlusion)

■ **Week 2 — Train & Evaluate** — (10–12 hours)
- ■ Transfer-train YOLOv8 (yolov8n → yolov8s); log metrics
- ■ Produce validation mAP, per-class precision/recall
- ■ Iterate hyperparams & pruning if needed

■ **Week 3 — Inference Server MVP** — (12–15 hours)
- ■ Build FastAPI endpoints (/infer) and WebSocket stream (/ws/stream)
- ■ Implement drawing pipeline and a sample front-end viewer
- ■ Test end-to-end with webcam / RTSP local stream

■ **Week 4 — Frontend Dashboard MVP** — (12–15 hours)
- ■ React app: live video canvas + bounding boxes
- ■ Alerts panel and simple analytics (events list)
- ■ Connect to backend WebSocket and REST endpoints

■ **Week 5 — Edge Optimization & Docker** — (8–12 hours)
- ■ Export to ONNX, benchmark ONNX Runtime on CPU
- ■ Quantize (INT8) if needed, test speed improvements
- ■ Dockerize backend & optionally frontend

■ **Week 6 — Analytics & Persistence** — (8–10 hours)
- ■ Add DB (Mongo/Postgres) to store event logs
- ■ Implement analytics queries (events/hour, top offenders)
- ■ Show analytics on dashboard and save test results

■ **Week 7 — Tests, Polish & Deploy** — (8–10 hours)
- ■ Unit tests for API, basic CI (GitHub Actions)
- ■ Deploy frontend (Vercel) and backend (Render / VPS)
- ■ Create demo video & architecture diagram

- **Week 8 — Demo, Resume & Wrap-up** — (6–8 hours)
  - Finalize README, publish model artifacts and results
  - Add resume bullets, demo script, 2-minute video
  - Prepare interview talking points and one-page case study

Sample daily schedule (part-time, Week 1 example)

- **Mon**: 2 hours — Setup annotation tool, collect 30 images from phone/camera
- **Tue**: 2 hours — Continue collection; upload 50 images to Roboflow / label 30
- **Wed**: 2 hours — Finish labeling 100 images; start augmentation pipeline
- **Thu**: 2 hours — Create train/val split; prepare train.yaml for YOLOv8
- **Fri**: 2 hours — Quick training run on subset; evaluate results
- **Sat**: 4 hours — Bulk label remaining images; improve augmentations
- **Sun**: 4 hours — Full training run; log initial metrics; rest and plan Week 2

Demo & Resume checklist (final deliverables)

- Record 2-minute demo video (live demo + 30s results + 30s architecture).
- Publish README with quickstart, training command, evaluation metrics.
- Push trained model weights (or link) and sample test images/labels.
- Add 'Resume bullets' with measured numbers: mAP, FPS (CPU/GPU), reduction in manual work (if available).
- Create a short case-study page (README/docs) and link from GitHub/LinkedIn.

Tools & quick commands YOLOv8 training command (example): *yolo task=detect mode=train model=yolov8n.pt data=train.yaml epochs=50 imgsz=640* Export to ONNX: *model.export(format='onnx')* Start backend (development): *uvicorn app.main:app --reload --host 0.0.0.0 --port 8000* Start frontend (Vite): *npm run dev* Local Docker build: *docker build -t edge-safety-backend ./backend*

Quick tips Track progress in small sprints and mark the checkboxes each day — 2 hours focused work beats 6 hours distracted work. If you hit a blocker on model accuracy, gather more labeled images for the failing class or augment with synthetic variations. Keep a short log (README/MD) of experiments: hyperparams, dataset size, mAP — this makes your interview narrative much stronger.

Good luck — start with Week 0 tasks today. If you want, I can also create a calendar (Google Calendar) format or a Trello board from this plan.