1. Overview

- **What** the app does (e.g., “a vehicle repair workshop management system”).
- **Who** it’s for (e.g., “for mechanics and administrators to track repairs, inventory, and client records”).
- **Tech stack**: Django (backend), MySQL (DB), React + Vite (frontend), Docker, NGINX.

---

2. Architecture

- **Container setup** with Docker Compose:
  - `frontend` (React/Vite)
  - `backend` (Django + Gunicorn)
  - `db` (MySQL)
  - `nginx` (reverse proxy/static server)
- **How traffic flows**: browser → NGINX → Django API or static React files.

---

3. Dev & Prod Environments

- Local: `http://localhost:3000`, Docker volumes for hot-reloading.
- Production: how you're deploying (e.g., VPS, NGINX config, env vars, HTTPS if applicable).

---

4. Live Demo

- Start from login or landing page.
- Show main features: CRUD operations, dashboard, filtering/search, etc.
- Bonus: demonstrate any validation, security, or role-based access.

---

5. Code Tour

- **Backend**: Django models, views, serializers, URL routing.
- **Frontend**: Component structure, API calls (using Axios/fetch), routing with React Router, state management with Zustand.
- **Docker/Deployment**: `docker-compose.yml`, NGINX config, `.env/secrets` usage.

📚 6. What I Learned / Challenges

- Architecture decisions.
- Handling CORS, static files, Docker.
- Anything clever or tricky (model normalization without breaking serializers).

🧠 7. Q&A or Discussion

Encourage feedback or suggestions:

- “Would you do anything differently?”
- “How would you scale or improve it?”
- “Do you see any bottlenecks?”