perplexity

Goal

Extend the existing Data-Labelling CRM blueprint[1] into a dual-purpose platform that

1. Orchestrates day-to-day work for a small product team (product design, frontend, backend, QA) and
2. Operates a "coding academy" where external learners pick up coding tasks, submit solutions, and receive staff reviews—mirroring the annotator QA loop shown in the review workflow video[2].

A Super Admin role must be able to administer every resource and workflow step, similar to the "Super Admin" patterns requested in Asana[3] and delivered in Postman[4].

1. Core Architecture & Stack

Layer	Technology	Rationale
Frontend	React (Vite) + React-Admin	React-Admin already supplies role-aware routing and dashboards[1].
Backend	Node.js + Express (or Next.js API routes)	Matches the previous template and enables API-first contracts[1] [5].
Database	PostgreSQL + Prisma ORM	Type-safe models for tasks, roles, reviews, billing[1].
AuthN / AuthZ	Clerk (or Auth0)	JWTs include role claims; integrates RBAC UI gating[1].
CI/CD	GitHub Actions → Docker images	Aligns to agile workflow guidance and shared pipelines[6] [5].
Logs / Metrics	Loki + Grafana	Centralised observability; Super Admin dashboards.

2. Role-Based Access Control (RBAC)

Role	Typical Members	Key Permissions
Super Admin	CTO, Platform Owner	Full CRUD on every entity, user impersonation[3] [4].
Product Designer	UX/UI team	Create Epics, view Dev Kanban, attach designs.
Frontend Dev	FE engineers	Claim tasks, push MR, request code review.
Backend Dev	BE engineers	Same as FE but for BE repos; define APIs.
QA	Test engineers	Create test plans, reject/accept merges.
Coding Student	External learner	Accept academy tasks, push solution repositories.
Academy Reviewer	Senior devs or mentors	Review student submissions, assign grades.

The RBAC table is stored in PostgreSQL and surfaced in JWTs, exactly as proposed in the original CRM design[1].

3. Internal Development Workflow

1. Requirement Capture
   - Product Designer creates a Feature ticket with acceptance criteria[6].
   - Backend & Frontend attend joint refinement to agree on API-first contract[5].
2. Task Breakdown
   - Tickets are split into FE, BE, QA subtasks and placed on a unified Kanban board (Jira/React-Admin view)[5].
3. Development & Code Review
   - Dev pushes branch → GitHub Actions run unit tests → pull-request triggers integrated build of BE & FE to catch contract violations early[5].
   - QA reviews PR, runs Cypress end-to-end suite in pipeline.
4. Merge & Deploy
   - On PR approval, CI deploys to staging; Super Admin can hot-fix via impersonation.
5. Retrospective
   - Metrics (cycle time, defect escape rate) are visible in Grafana dashboards for team and Super Admin oversight.

4. Coding Academy Workflow

The academy mirrors the annotator "submit → review → approve/reject" loop in Superb AI's workflow demo[2].

1. Sign-up & On-boarding
   - Student registers (Clerk) and selects a learning track.
   - System assigns starter tasks (repo forks) based on skill level.
2. Task Execution
   - Student completes coding exercise and pushes to a protected branch.
   - A webhook auto-creates a "Review" ticket tagged to an Academy Reviewer.
3. Automated Checks
   - CI runs linter, unit tests; failures push task back to student automatically.
4. Human Review
   - Reviewer opens the queue (React-Admin list) → approves (green) or rejects (red) with comments—exactly the approve/reject UX from the video[2].
5. Certification & Progression
   - Approved tasks increment student XP; curriculum engine unlocks harder tasks.
   - Super Admin can override grades or reassign mentors when needed.

5. Super Admin Capabilities

Borrowing the “manage anything” scope highlighted in Postman’s Super Admin[4] and the need raised by Asana users[3]:

- Global user & org management, including academy students.
- Force-delete or re-assign any project or ticket.
- Impersonate roles to debug UX issues live.
- Edit RBAC matrix and feature flags at runtime.
- View unified analytics (team KPIs + academy metrics) on a single dashboard.

6. Key Data Models (Prisma)

```
model User {
  id           String   @id @default(cuid())
  email        String   @unique
  role         Role
  xp           Int      @default(0)         // academy progress
  tasks        Task[]   @relation("Assignee")
  reviews      Review[] @relation("Reviewer")
}

model Task {
  id           String   @id @default(cuid())
  title        String
  type         TaskType // FEATURE, BUG, ACADEMY_EXERCISE...
  status       Status   // IN_PROGRESS, SUBMITTED, APPROVED...
  repoUrl      String?
  assignee     User?    @relation("Assignee", fields:[assigneeId], references:[id])
  assigneeId   String?
  reviews      Review[]
}

model Review {
  id           String   @id @default(cuid())
  task         Task     @relation(fields:[taskId], references:[id])
  taskId       String
  reviewer     User     @relation("Reviewer", fields:[reviewerId], references:[id])
  reviewerId   String
  outcome      ReviewOutcome // APPROVED / REJECTED
  comment      String?
}
```

7. Incremental Roadmap (6 Weeks)

Week	Deliverable
1	RBAC schema, Super Admin dashboard skeleton.
2	Unified Kanban (dev tasks), API-first contracts, mock server pipeline[5].
3	CI/CD with integrated FE+BE builds and test matrices[5].

Week	Deliverable
4	Academy module: task assignment, CI auto-grading, review queue UI[2].
5	Metrics dashboards (Grafana) and Super Admin impersonation.
6	Security hardening, Cloudflare WAF, production launch[1].

8. Environment Support for Juniors

Following the “beginner-friendly environment setup” mandate in existing projects.memory[7], provide:

- Pre-built Docker dev images with VS Code Remote extensions.
- One-click starter repos for academy tasks.
- Live playground for API experimentation (Swagger UI).

Result

By layering an academy review loop onto the proven CRM scaffold[1] and enforcing modern dev-team practices drawn from industry guidance[6] [5], the platform becomes a single pane of glass where:

- Core product work moves from ideation to production with tight FE/BE/QA collaboration, and
- Learners practice real coding tasks under mentor supervision—while a Super Admin retains total control over users, resources, and workflows[3] [4].

⁂

1. Data-Labelling-CRM.txt
2. https://www.youtube.com/watch?v=L25xafJgD1E
3. https://forum.asana.com/t/super-admin-ability-to-manage-all-project-details/613463
4. https://blog.postman.com/introducing-the-super-admin-role/
5. https://www.zigpoll.com/content/what-strategies-can-a-product-lead-implement-to-ensure-effective-collaboration-between-backend-developers-and-frontend-teams-during-the-development-of-new-features
6. https://www.pluralsight.com/resources/blog/software-development/workflow
7. projects.environment_setup