# TECHNICAL DESIGN DOCUMENT: iStock - RAG for Precision Livestock

*Based on the Hytel AI Coding Bootcamp TDD Template and Rubric.*

## MY BENEFIT

**One-sentence pitch:** iStock is an AI-powered livestock health and nutrition application that provides farmers with immediate, citable diagnostic and treatment advice by grounding large language models in verified veterinary science and local regulatory data.

## 1. OVERVIEW

* **Goal:**
  + To reduce livestock loss by enabling farmers to diagnose and initiate treatment for common diseases 24/7.
  + To increase farm efficiency by optimizing feed formulations based on real-time animal metrics and local ingredients.
  + To provide decision support that is verifiable and grounded in localized, trustworthy information, thus addressing the risk of LLM 'hallucination'.
* **Key features (MVP):**
  + **RAG Chatbot:** Conversational interface for diagnostic questions (symptom input) with source citations.
  + **Feed Optimizer Tool:** Calculator that takes animal type/stage, ingredient price/nutritional values, and outputs a least-cost ration.
  + **Auth & User Management:** Secure login and profile creation (user role: Farmer or Vet/Consultant).
* **Target users & success criteria:**
  + **Target Users:** Small to Medium Commercial Livestock Farmers (Dairy/Beef Cattle).
  + **Success Criteria:** **(Quality & Testing Rubric Target: ≥80% coverage)**, **(Product Management Rubric Target: Defined MVP accepted against criteria)**, 90% accuracy on RAG-generated treatment protocols (verified by human veterinary review), and a 15% demonstrated reduction in feed costs for pilot users.

## 2. TECH STACK (GOLDEN PATH)

* **Runtime:** Node (Firebase Gen 2 Cloud Functions)
* **Language:** TypeScript (strict)
* **Front-end:** React + Vite
* **UI kit:** shadcn/ui (Radix + Tailwind source-copy model)
* **State / data fetching:** TanStack Query
* **Forms & validation:** React Hook Form + Zod resolver
* **Shared validation:** Zod (client & server)
* **API layer:** tRPC (typed RPC)
* **Backend services:** Firebase Auth · Firestore · Functions
* **AI Service:** Gemini API (for RAG/Multimodal processing)
* **RAG Infrastructure:** External Vector Database (e.g., Pinecone/ChromaDB) for knowledge indexing.

## 3. MONOREPO LAYOUT (PNPM)

*(Structure remains compliant with Rubric standards for Architecture & Code Organization)*

.  
├── apps/  
│ └── web/ ← React front‑end, UI components, Chatbot UI  
├── functions/ ← Cloud Functions / tRPC routers (Health, Nutrition, User)  
├── packages/  
│ ├── shared/ ← Zod schemas (FeedIngredient, Diagnosis), common types  
│ ├── rag-service/ ← Logic for querying the Vector DB and calling Gemini  
│ └── seeding/ ← Initial vector embedding scripts and Firestore data  
├── docs/ ← Project docs (this TDD, ADRs, API notes)  
└── .github/ ← CI workflows

## 4. ARCHITECTURE

* **Client (React):** User interacts with the Chatbot or Feed Optimizer form. Data submitted via tRPC.
* **tRPC Handlers (Functions):**
  + **Health Router:** Calls the rag-service/ package logic.
  + **Nutrition Router:** Runs the least-cost linear programming solver (a pure TypeScript function) with inputs.
* **rag-service Package:** Encapsulates the logic to turn a user query into a vector search, retrieve context from the Vector DB, and send the final prompt to the Gemini API for generation.
* **Data Flow:** Client ⇄ tRPC Function ⇄ RAG Service ⇄ Vector DB + Gemini API

## 5. DATA MODEL (Firestore)

|

| Entity | Key fields | Notes |

| User | uid, email, role (Farmer/Vet) | Auth via Firebase |

| FarmProfile | userId, location (geo), mainLivestockType | Links user to farm context (for RAG localization) |

| Ingredient | name, unitPrice, nutritionalValues (map) | User-defined or retrieved local feed ingredients |

| HealthRecord | animalId, diagnosis, protocol (RAG generated) | Stores RAG chat history and treatment protocol for traceability |

* **Security rules:** Enforce Principle-of-Least-Privilege: Users can only read/write their own FarmProfile and HealthRecords. Vets can read shared, anonymized public health data.
* **Index strategy:** Composite indexes on HealthRecord for rapid retrieval by animalId and diagnosisDate.

## 6. API DESIGN (tRPC)

| Router | Procedure | Input (Zod schema) | Output |

| health | askRag | query (string), context (FarmProfile ID) | RagResponse (text, sources, confidence) |

| health | saveRecord | HealthRecord data | success (boolean) |

| nutrition | optimizeFeed | targetAnimal, ingredients (array) | FeedRation (ingredient percentages, cost) |

## 7. TESTING STRATEGY

*(Adhering to Quality & Testing Rubric target: ≥80% coverage)*

| Level / focus | Toolset | Scope |

| Unit | Vitest | Pure functions, especially feed formulation linear algebra solver in nutrition router. |

| Component | Vitest + Testing Library | React components (Feed Optimizer Form, Chat Interface). |

| End-to-end | Playwright | Auth flows, successful feed optimization calculation, and a full RAG query/response cycle. |

* **Fixtures / Seeding:** pnpm seed will include loading initial vector embeddings into the RAG Vector DB for testing.

## 8. CI / CD PIPELINE (GITHUB ACTIONS)

*(Pipeline is Turbo-aware and adheres to Dev Experience Rubric standards)*

1. Setup PNPM and restore Turbo remote cache
2. pnpm exec turbo run lint typecheck – **Ensure all Zod schemas and tRPC routes are type-checked.**
3. pnpm exec turbo run test – Run Vitest unit/component tests.
4. pnpm exec turbo run e2e – Playwright suite (headless).
5. Deploy preview (Firebase Hosting channel).
6. Changesets release & promote to prod on merge to main.

## 9. ENVIRONMENTS & SECRETS

| Env | URL / target | Notes |

| local | localhost:5173 | .env + Firebase emulators; Validated by T3 Env. |

| preview-\* | Firebase Hosting channel | Auto-created per PR. |

| prod | https://istock-app.com | Promote via CI workflow. |

Secrets include GEMINI\_API\_KEY, VECTOR\_DB\_URL, etc., handled securely with firebase functions:config:set and GitHub repo secrets.