# iStock: RAG for Livestock - 5-Milestone Development Plan

*This plan is designed to meet or exceed the "Exceeds Expectations" criteria of the Hytel AI Coding Bootcamp Rubric, prioritizing Quality & Testing (≥80% coverage) and thoughtful Architecture.*

## Milestone 1: Foundational Architecture & User Authentication (Weeks 1-2)

**Goal:** Establish the monorepo structure, deploy the initial web app, and secure user authentication.

| **Task Category** | **Specific Tasks (TDD & Rubric Focus)** | **Success Criteria** |
| --- | --- | --- |
| **DevOps/Infra** | Set up PNPM monorepo with apps/web, functions/, packages/shared, and **packages/rag-service**. | Turbo-aware CI/CD pipeline passes **lint & typecheck**. |
| **Auth/Security** | Implement Firebase Auth and **Firestore Security Rules** to enforce basic user ownership (Principle-of-Least-Privilege). | Users can register/login. Auth flow is covered by **Playwright E2E tests** (Happy Path). |
| **Frontend** | Implement basic layout using **Tailwind/shadcn/ui** (Clean, consistent UI). Build the basic FarmProfile creation form. | Form uses **React Hook Form + Zod validation** (Meets/Exceeds Frontend Implementation). |
| **Backend** | Create user tRPC router (getById, updateProfile). Define and share **Zod schemas** for FarmProfile (Architecture & Code Organization). | All API requests are **typed** end-to-end. |

## Milestone 2: RAG Knowledge Base & Core API Integration (Weeks 3-4)

**Goal:** Integrate the RAG service and enable the first AI-powered functionality.

| **Task Category** | **Specific Tasks (TDD & Rubric Focus)** | **Success Criteria** |
| --- | --- | --- |
| **RAG/Data** | Select and deploy Vector DB (e.g., ChromaDB). Index initial **veterinary manuals** into the vector store via a seeding script. | The **rag-service** package successfully performs vector search and retrieves context chunks. |
| **AI/Backend** | Implement the health.askRag tRPC procedure. This procedure must handle the **Gemini API call** with the retrieved context. | Successful generation of a basic, cited response to a test query ("Cow fever causes") is logged via **Cloud Logging**. |
| **Frontend** | Build the basic **RAG Chatbot UI** (input box, history display). | Chatbot displays responses with the **citation/source text** from the RAG service (Meets Expectations UI/UX - Traceability). |
| **Quality** | Write **Unit tests** for the rag-service function that formats the final prompt to the LLM (≥60% coverage target). |  |

## Milestone 3: Precision Nutrition Module (Weeks 5-6)

**Goal:** Deliver the second core feature: the least-cost feed formulation tool.

| **Task Category** | **Specific Tasks (TDD & Rubric Focus)** | **Success Criteria** |
| --- | --- | --- |
| **Backend/Math** | Implement the **least-cost linear programming solver** in a pure function within functions/nutrition router. | The solver accurately calculates the optimal feed ratio based on mock data inputs. |
| **Data Model** | Define the Ingredient and FeedRation Zod schemas and Firestore models. | **Firestore rules** are updated to allow user-specific ingredient list creation/editing. |
| **Frontend** | Build the **Feed Optimizer UI** (form for ingredient input, display for calculated ratio). | Form uses **React Hook Form** for complex array field management. |
| **Quality** | Achieve **≥80% Unit Test coverage** on the linear programming solver function (Quality & Testing). |  |

## Milestone 4: Multimodal & Error Handling (Weeks 7-8)

**Goal:** Enhance the RAG system with image input and establish robust production monitoring.

| **Task Category** | **Specific Tasks (TDD & Rubric Focus)** | **Success Criteria** |
| --- | --- | --- |
| **AI/Multimodal** | Update the health.askRag procedure to handle **image uploads** (e.g., using Firebase Storage and then passing the base64 image to Gemini). | The system can accurately diagnose a disease from a picture of a skin lesion or symptom (validated against test data). |
| **Monitoring** | Implement front-end error capture via **Crashlytics/Sentry** and structured logging on Cloud Functions (Monitoring & Logging). | A failed API call triggers a logged event with full stack trace and user context. |
| **Error Handling** | Implement graceful failure mechanisms for when the RAG returns low-confidence results (e.g., "I cannot find a confident answer, consult a vet," instead of hallucination). | API includes standard **tRPC error conventions** (e.g., AUTH\_ERROR, BAD\_REQUEST). |
| **Product Mgmt** | Stakeholder demo and backlog grooming. Define tasks for Milestone 5. |  |

## Milestone 5: Polishing, Performance, & Final Audit (Weeks 9-10)

**Goal:** Achieve "Exceptional" status for Design/UI/UX and ensure high-quality production readiness.

| **Task Category** | **Specific Tasks (TDD & Rubric Focus)** | **Success Criteria** |
| --- | --- | --- |
| **Quality/Test** | Final full E2E Playwright run. Verify coverage is **≥80% overall**. | Zero regressions in E2E tests. Code coverage report confirms target achieved. |
| **UI/UX** | Refine UI with **custom theming** and motion polish (Pixel-perfect, branded design). Conduct **WCAG 2.1 AA audit** (Accessibility). | All components pass the Storybook a11y addon checks. App is visually impressive and fast. |
| **Scalability** | Tune **TanStack Query caching** and prefetching on key animal record data. Implement simple denormalization logic (if required). | App feels responsive; no unnecessary network calls observed. |
| **Documentation** | Finalize **TDD** and add an **ADR** (Architectural Decision Record) for the RAG architecture choice. | All documentation is complete and up-to-date (Architecture & Code Organization). |