Expectations & Goals

Before diving into the codebase, developers should understand the following expectations:

- Modular Architecture: Follow clean separation of concerns—entities, repositories, services, controllers, and DTOs.
- Security First: Implement JWT-based authentication and role-based access control for two user types: Uploader and Reviewer.
- Entity Relationships: Model realistic relationships including @ManyToOne, @OneToMany, @ManyToMany, and @Embeddable fields.
- Custom Queries: Write JPQL or Criteria-based queries to support flexible search and filtering logic.
- Pagination & Sorting: Ensure all list endpoints support pagination and sorting via Spring Data.
- Caching: Use Spring Cache to optimize frequently accessed endpoints.
- Testing Discipline: Write unit tests for services and controllers using JUnit 5 and Mockito.
- Scalability & Extensibility: Design with future enhancements in mind—DTO mapping, Swagger documentation.

Developer Mindset

Approach this POC not just as a coding task, but as a **real-world simulation** of building a secure, scalable, and testable backend system. Think critically about:

- How data flows across layers
- How to isolate and test logic
- How to optimize queries and caching
- How to enforce security without compromising usability

Business & Feedback API

Table of Contents

1. Project Overview
2. Entity Design
3. API Endpoints
4. Search & Filter Logic
5. Feedback Reactions
6. Testing Strategy
7. Bonus Features
8. Suggested Enhancements

Project Overview

Objective: Develop a Spring Boot REST API to manage business listings and feedback interactions between two types of users:

- **Uploader**: Can create, edit, and delete business listings.
- **Reviewer**: Can search businesses, submit feedback, and react to others feedback.

Entity Structure

https://github.com/fsdtrinings/Final_POC_FinosBatch_Sep2025.git

Business Management (Uploader)

- POST /businesses – Create business
- PUT /businesses/{id} – Update business
- DELETE /businesses/{id} – Delete business

Product Management

- POST /products – Add product to business
- GET /products/{id} – View product

Feedback Submission (Reviewer)

- POST /feedbacks/business/{businessId} – Submit feedback for business
- POST /feedbacks/product/{productId} – Submit feedback for product

Note: [must add few more endpoints, based on your understanding]

Implement

1) Spring Cache
2) Pagination
3) Swagger
4) Role-based Access control
5) DTO
6) Custom Exception Handling
7) Custom Query
8) HATEOAS (optional)