Decision record — Image storage gateway and externalized configuration (supports the scenario above)

Title
Adopt an object-storage gateway with partner adapters and externalized configuration for remote database access

Status
Accepted

Context
We must integrate product image uploads into the catalog flow while keeping partner changes low-effort and enabling developers to work from any device without provisioning local data services. The platform provides a managed relational database and managed object storage in Aiven Cloud. We want configuration to be consistent and simple across developer machines and environments.

Decision

- Introduce a storage gateway that exposes a single interface for uploading images and returning a public link. Partner-specific differences are handled by small adapters behind the gateway.
- Store all connection details in environment variables and keep configuration outside of the application code so any developer can sign in to the remote database and storage from any machine.

Consequences
Positive

- Faster integrations: new partner or storage provider differences are absorbed by an adapter, not by the rest of the system.
- Lower coupling: product logic depends only on the storage gateway contract, not on provider details.
- Developer mobility: a developer can clone the project, provide environment variables, and immediately run image upload and product linking without local database or local storage setup.
- Safer operations: secrets and connection strings are not hard-coded and can be rotated without code changes.
  Negative
- Additional layer: the gateway and adapters add a small maintenance surface.
- Contract stewardship: the gateway contract must remain stable as the system evolves.

Alternatives Considered

- Call the storage service directly from product logic.
  Rejected: couples business logic to a specific provider and spreads storage rules across the codebase.
- Maintain local databases and local object storage on every developer machine.
  Rejected: heavy setup, inconsistent states, poor mobility, and slow onboarding.

Mapped Pattern and how it is applied

- **Gateway pattern (object-storage gateway):** A single entry point handles image uploads, content-type checks, and size limits, and returns a public link. This hides provider differences and centralizes validation and retry behavior.
- **Adapter pattern (partner adapters):** When a partner sends images or metadata differently, an adapter maps those inputs to the gateway's expected format, confining change to a narrow place.
- **Externalized configuration pattern (environment-based settings):** Database address, credentials, and storage endpoints are read from environment variables, enabling secure remote sign-in to the managed database from any machine and keeping builds identical across environments.
- **Repository pattern (for product persistence):** Product create and update operations write the image link and fields in one place, keeping business logic clean and testable.

Decision record — Hosted OpenID Connect with silent authentication

Title
Use a hosted identity provider with OpenID Connect and silent token refresh for administration sign-in

Status
Accepted

Context
Administrators may attempt to open the administration dashboard while not signed in. We must authenticate quickly and redirect back to the original route while administration statistics load. End-to-end sign-in time should be eight hundred milliseconds or less.

Decision
Adopt a hosted identity provider that supports OpenID Connect with the Authorization Code flow using Proof Key for Code Exchange and silent token refresh. Offload e-mail delivery and token creation to the provider.

Consequences
Positive

- Lower end-to-end delay for sign-in; fewer authentication defects; no local e-mail or token creation to maintain.
- Focused application tier: less work on the critical path to load the administration page.
  Negative
- External dependency on the identity provider; provider incidents affect sign-in.

Alternatives Considered

- Local magic-link e-mail sign-in and locally minted tokens.
  Rejected: slower, more operational overhead, and more code to secure.

Mapped Pattern and how it is applied

- **External authentication pattern:** Delegate identity and token lifecycle to a hosted provider.
- **Session continuation pattern:** After sign-in, return the user to the originally requested route.
- **Deferred input and output pattern:** Load administration statistics after the authentication callback completes to reduce perceived delay.

Decision record — High-volume product upload performance controls

Title
Control upload demand and processing cost for one hundred concurrent product image uploads

Status
Accepted

Context
One hundred concurrent users may upload a product with an image of roughly three megabytes. We must complete at least ninety-five percent of uploads within thirty-five seconds and keep central processing unit utilization below eighty-five percent. Provided results show average total duration twenty-seven point zero one seconds, success rate ninety-nine point two percent, average upload time thirteen point four six seconds, and central processing unit usage values per run with an average of eighty point four percent.

Decision

- Limit how quickly new uploads enter the system using admission control: cap in-flight uploads per browser and apply a server-side rate limit for presigned upload addresses and listing creation. When limits are exceeded, queue client requests or return Hypertext Transfer Protocol status code four hundred twenty-nine with a Retry-After header.
- Reduce computational overhead by offloading image storage and identity to managed services and keeping the application tier focused on orchestration and metadata.

Consequences
Positive

- Predictable averages and tails; central processing unit headroom under load; demand kept within safe limits for storage and the application tier.
- Clear back-pressure signals to clients rather than silent slowdown.
  Negative
- Users may experience short waits during bursts; requires transparent user interface messaging.

Alternatives Considered

- Unlimited concurrency.
  Rejected: causes queue explosions and long tail delays.
- Single global queue.
  Rejected: becomes a bottleneck and a single point of contention.

Mapped Pattern and how it is applied

- **Admission control pattern (throttling):** Bound the arrival rate and number of concurrent uploads to protect downstream services.
- **Work shedding pattern:** When limits are reached, respond with a clear retry signal instead of accepting unbounded work.
- **Offloading pattern:** Move byte handling (images) and identity to managed services to reduce computation per request.

Decision record — Aiven Cloud database continuity

Title
Use a managed relational database cluster with automatic failover and point-in-time restore

Status
Accepted

Context
The primary database node may become unavailable due to a regional incident or scheduled maintenance. The service must continue with minimal interruption.

Decision
Adopt a managed relational database cluster on Aiven Cloud with automatic failover, continuous replication, frequent backups, and point-in-time restore.

Consequences
Positive

- Client traffic is redirected to a healthy node; write operations resume without manual intervention; recent changes are preserved by replication and backups.
- Monthly availability target of at least ninety-nine point ninety-five percent and restore to any second within the most recent seven days.
  Negative
- Higher service cost than a single self-managed instance.

Alternatives Considered

- Single primary database without managed failover.
  Rejected: longer outages and higher recovery time.

Mapped Pattern and how it is applied

- **Multiple copies of data pattern:** Synchronous or near-synchronous replicas survive a single-node loss.
- **Fault detection and automatic repair pattern:** Health checks, leader election, and automatic failover.
- **Backup and restore pattern:** Continuous archive with tested restores to bound possible data loss.
- **Resource isolation pattern:** Separate storage and compute to limit blast radius.

Decision record — Business catalog upload with feedback

Title
Provide per-item feedback and targeted recovery during catalog uploads

Status
Accepted

Context
A verified business user uploads a catalog file that often contains both valid and invalid rows. Users need actionable, item-level feedback.

Decision
Return immediate per-item results that separate successes from failures, include specific error reasons, and enable targeted recovery actions such as retrying only failed rows. Support optional pause, resume, and cancel for long uploads.

Consequences
Positive

- Faster recovery, higher task completion, and reduced confusion during multi-step tasks.
  Negative
- Additional user interface states and server response shapes to maintain.

Alternatives Considered

- Provide only a single global success or failure result.
  Rejected: difficult for users to correct mistakes quickly.

Mapped Pattern and how it is applied

- **Support user initiative pattern:** Keep users in control with precise feedback and minimal rework.
- **Partial retry pattern:** Enable reprocessing only of the failing subset.

Decision record — Administration updates to products on sale

Title
Allow administrators to start, end, or overwrite a sale window with immediate effect

Status
Accepted

Context
An administrator selects one or more products and configures a sale window and discount. Edits may shorten or extend existing windows.

Decision
Provide clear controls to start or end a sale and overwrite a sale window by changing the interval. Apply the change deterministically so the effective sale for a product is unambiguous.

Consequences
Positive

- Immediate, predictable changes on linked products; simple mental model for operators.
  Negative
- Requires explicit conflict handling and clear confirmations.

Alternatives Considered

- Implicit last-writer-wins without visibility.
  Rejected: increases confusion and mispricing risk.

Mapped Pattern and how it is applied

- **Deterministic precedence pattern:** Choose a single effective sale based on priority and, on ties, the most recent update time.
- **Support user initiative pattern:** Show previews, confirmations, and results for overrides.

Decision record — Business panel module addition

Title
Separate business features to reduce coupling and increase cohesion

Status
Accepted

Context
We must add business verification and catalog upload features without changing existing user routes.

Decision
Create a business-focused module and endpoints that group verification and catalog upload together, leaving existing user routes unchanged.

Consequences
Positive

- Faster addition of features and clearer ownership boundaries.
  Negative
- More modules to coordinate and document.

Alternatives Considered

- Place all routes in a single monolithic module.
  Rejected: harder to maintain and evolve.

Mapped Pattern and how it is applied

- **Separation of concerns pattern:** Group related features to increase cohesion and reduce coupling.
- **Modular boundary pattern:** Stable interfaces between modules limit ripple effects.

Decision record — Test mode for safe automation

Title
Enable a test-only authentication bypass and deterministic fixtures

Status
Accepted

Context
We need repeatable tests without changing production authentication behavior.

Decision
Add a test-mode environment switch that enables a limited authentication bypass for automation and uses seeded data fixtures; keep production behavior unchanged.

Consequences
Positive

- Fast, reliable automated tests; zero impact on production.
  Negative
- Additional configuration path to secure and document.

Alternatives Considered

- Run tests against production settings.
  Rejected: unsafe and slow.

Mapped Pattern and how it is applied

- **Defer binding pattern:** Choose authentication strategy at runtime based on environment.
- **Test isolation pattern:** Seed and clean state per run.

Decision record — Business partner catalog integration

Title
Integrate new partner catalogs quickly using an extract-transform-load contract

Status
Accepted

Context
External partners need to integrate their product catalogs with minimal disruption to existing flows.

Decision
Use a stable extract-transform-load contract and add partner-specific adapters that emit a canonical structure to the catalog system.

Consequences
Positive

- New partner integration delivered within one week and no more than two person-weeks; existing uploads keep working.
  Negative
- Adapter growth over time requires stewardship.

Alternatives Considered

- One-off direct mappings per partner.
  Rejected: duplicated logic and brittleness.

Mapped Pattern and how it is applied

- **Encapsulation pattern:** Hide source differences behind a single contract.
- **Standards adherence pattern:** Require comma-separated values or JavaScript Object Notation with required fields to reduce format mismatch.

Decision record — Student seller image upload with multi-format support

Title
Accept multiple image formats for student sellers and link them reliably to products

Status
Accepted

Context
Student sellers may upload images in different formats such as Joint Photographic Experts Group, Portable Network Graphics, or Web Picture format. The system must accept supported formats, reject unsupported ones clearly, and store a link to the image on the product.

Decision
Detect format and validate size and content type at upload; store the image in object storage;

return a public link and write it to the product record. Adding support for a new format requires only extending validation and storage mapping.

Consequences
Positive

- Compatibility across common formats; clean failures for unsupported files; no partial product writes on rejection.
  Negative
- Ongoing maintenance of the allowed formats list and validation rules.

Alternatives Considered

- Accept any format and attempt best-effort conversion.
  Rejected: unpredictable outcomes and higher failure rates.

Mapped Pattern and how it is applied

- **Capability negotiation pattern:** Check and accept only supported formats.
- **Validation and guard pattern:** Enforce size and content-type rules at the boundary.
- **Idempotent write pattern:** Ensure the product record always points at the intended last image.

Decision record — Flash sale buying load test

Title
Exercise flash sale purchasing with concurrent buyers and verify stock integrity

Status
Accepted

Context
We must validate flash sale behavior under high concurrent load, ensuring latency bounds and correct stock handling.

Decision
Run a surge test with fifty concurrent buyers, measure latency, and assert that stock counts never go below zero and do not oversell.

Consequences
Positive

- Early detection of race conditions and long-tail latency.
  Negative
- Test harness maintenance and data seeding required.

Alternatives Considered

- Manual spot checks.
  Rejected: not repeatable and low coverage.

Mapped Pattern and how it is applied

- **Test isolation pattern:** Reset data before each run.
- **Invariant enforcement pattern:** Guard stock mutations with transactional checks.

Decision record — Unauthorized business access prevention

Title
Enforce identity, role, and verified-business status before privileged actions

Status
Accepted

Context
A malicious user may attempt to upload a product catalog without proper business verification.

Decision
Validate signed tokens for identity, require both an appropriate role and verified-business status, and scope upload routes so they are reachable only through protected paths. Deny by default when checks fail.

Consequences
Positive

- Zero successful unauthorized catalog uploads; clear audit trail.
  Negative
- Policy and role lifecycle require governance.

Alternatives Considered

- Role checks without verified-business status.
  Rejected: does not enforce real-world eligibility.
- Manual reviews for every upload.
  Rejected: slow and error-prone.

Mapped Pattern and how it is applied

- **Authenticate actors pattern:** Verify the signer, audience, issuer, and expiration on the token before any action.
- **Authorize actors pattern:** Enforce role and verified-business status; deny by default.
- **Route scoping pattern:** Restrict endpoints to protected areas to reduce attack surface.
- **Document validation pattern (optional):** Where required, validate trade license details before enabling business features.