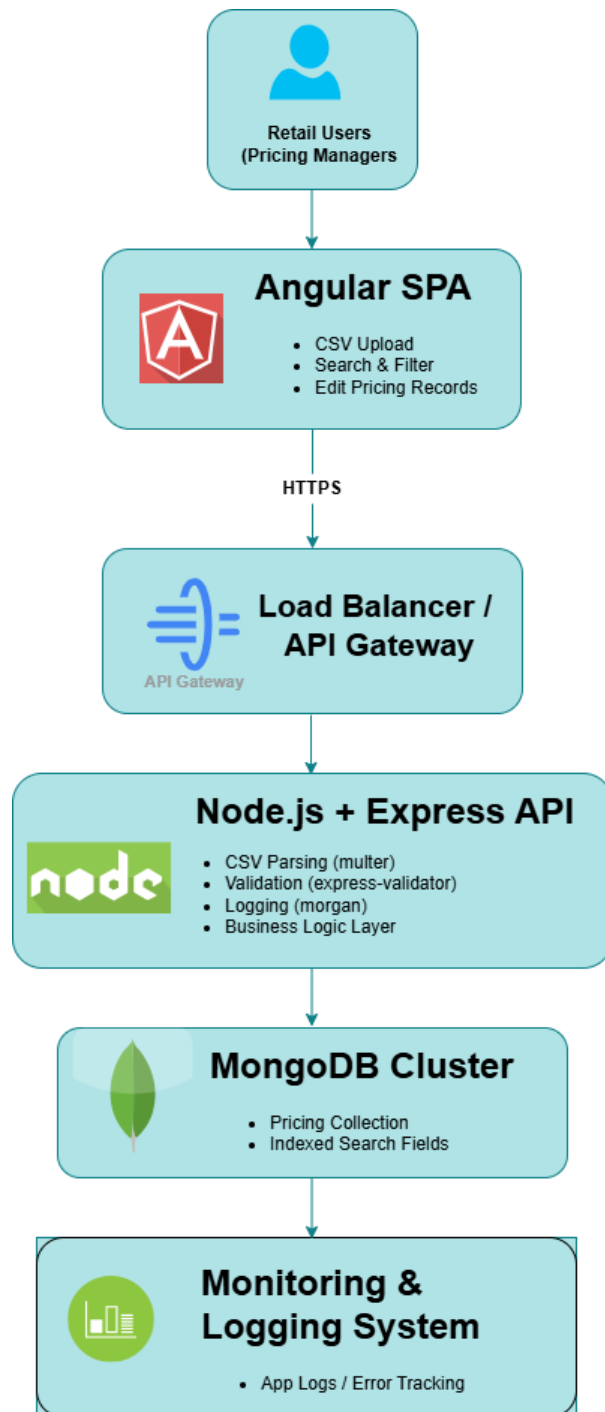


Retail Pricing Management System (RPMS)

High-Level Architecture Solution:

The system follows a 3-tier architecture:

1. Presentation Layer (Angular SPA)
2. Application Layer (Node.js REST API)
3. Data Layer (MongoDB Cluster)



Layered Architecture Breakdown

1. Presentation Layer

Technology:

- Angular (Standalone SPA)

Responsibilities:

- CSV file upload
- Search/filter pricing records
- Editable pricing grid
- Form validation
- API communication via HttpClient
- Error handling UI

Key Features:

- Reactive forms
- Pagination
- Client-side validation
- Responsive UI

2. Application Layer (Backend API)

Technology:

- Node.js + Express

Responsibilities:

- RESTful API endpoints
- CSV parsing
- Data validation
- Business rules enforcement
- Logging
- Error handling

Packages Used:

- express
- mongoose
- multer (CSV upload)
- csv-parser
- express-validator
- morgan
- dotenv
- cors

Architecture Pattern:

Controller → Service → Repository Pattern

3. Data Layer

Technology:

- MongoDB Atlas Cluster

Collection:

- pricing
- Example Document:

```
{
  "storeId": "S1001",
  "sku": "SKU123",
  "productName": "Laptop",
  "price": 999.99,
  "date": "2026-02-23T00:00:00Z",
  "createdAt": "...",
  "updatedAt": "..."}

```

Indexes:

- storeId
- sku
- date
- compound index (storeId + sku + date)

Key Architectural Decisions

Decision	Reason
Angular SPA	Fast UI, reactive updates
Node.js	High concurrency, non-blocking I/O
MongoDB	High write volume support
Docker	Environment consistency
REST API	Standard integration model
Indexing strategy	Fast search across 3000 stores