Software Architecture and Design Specification

Project: Online Marketplace for Handcrafted Goods

Version: 1.0

Authors: Akshay, Akash, Anirudha, Anish

Date: 01-10-2025

Status: Draft

Revision History

Version | Date | Author | Change Summary

Approvals

Role | Name | Signature/Date

1. Introduction

1.1 Purpose
This document specifies the architecture and design of the Online Marketplace for Handcrafted Goods system.

1.2 Scope
Covers core marketplace services: product listing, user registration, shopping cart, payments, and order management.

1.3 Audience
Developers, QA engineers, security auditors, designers, instructors, and maintenance teams.

1.4 Definitions
SKU, JWT, PCI-DSS, TLS, ADR.

2. Document Overview

2.1 How to use this document
Provides architectural deliverables including UML diagrams, ADRs, threat models, and API design.

2.2 Related Documents
SRS, STP, RTM.

3. Architecture

3.1 Goals & Constraints

Goals: Secure, scalable, user-friendly, 99.9% availability

Constraints: PCI-DSS compliance, third-party payment/shipping integration, mobile responsiveness

3.2 Stakeholders & Concerns

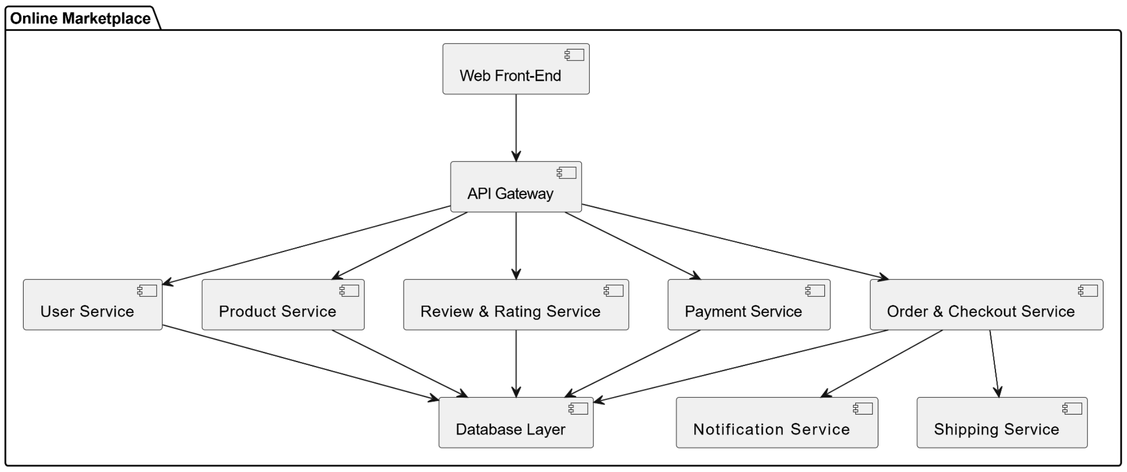
Buyers/Sellers: security, ease of use, availability

Marketplace Admins: maintainability, analytics

Regulators: data protection, compliance

Developers: modularity, scalability

3.3 Component (UML) Diagram



3.4 Component Descriptions

Web Front-End: Handles buyer and seller interactions through a responsive UI.

API Gateway: Routes client requests to appropriate backend services securely.

User Service: Manages user registration, login, and authentication.

Product Service: Handles product creation, editing, and catalog browsing.

Order & Checkout Service: Processes carts, orders, and payment initiation.

Review & Rating Service: Manages user reviews and product ratings.

Payment Service: Handles secure payment capture and records.

Database Layer: Stores user, product, order, and transaction data.

Notification Service: Sends emails or messages for order updates.

Shipping Service: Integrates with third-party APIs to calculate and track shipping

3.5 Chosen Architecture Pattern and Rationale

Layered architecture chosen for clear separation of concerns and maintainability. Microservices used only where necessary to support scalability and modular development.

3.6 Technology Stack & Data Stores

Next.js (frontend), Node.js + Express (API), MYSQL (DB), AWS S3 (media), TLS encryption, JWT for authentication.

3.7 Risks & Mitigations

Risk: Payment failure due to third-party API outage → Mitigation: Fallback to alternative provider or queue retry.

Risk: Image upload abuse → Mitigation: File size/type restrictions, virus scanning.

3.8 Traceability to Requirements

R1 (User authentication) → User Service

R2 (Product listing) → Product Service

R3 (Order placement) → Order & Checkout Service

R4 (Payment capture) → Payment Service

R5 (Review submission) → Review & Rating Service

3.9 Security Architecture

Threat Modeling (STRIDE):

- Spoofing → JWT-based auth with role checks
- Tampering → Signed requests, input validation
- Repudiation → Audit logs for orders/payments

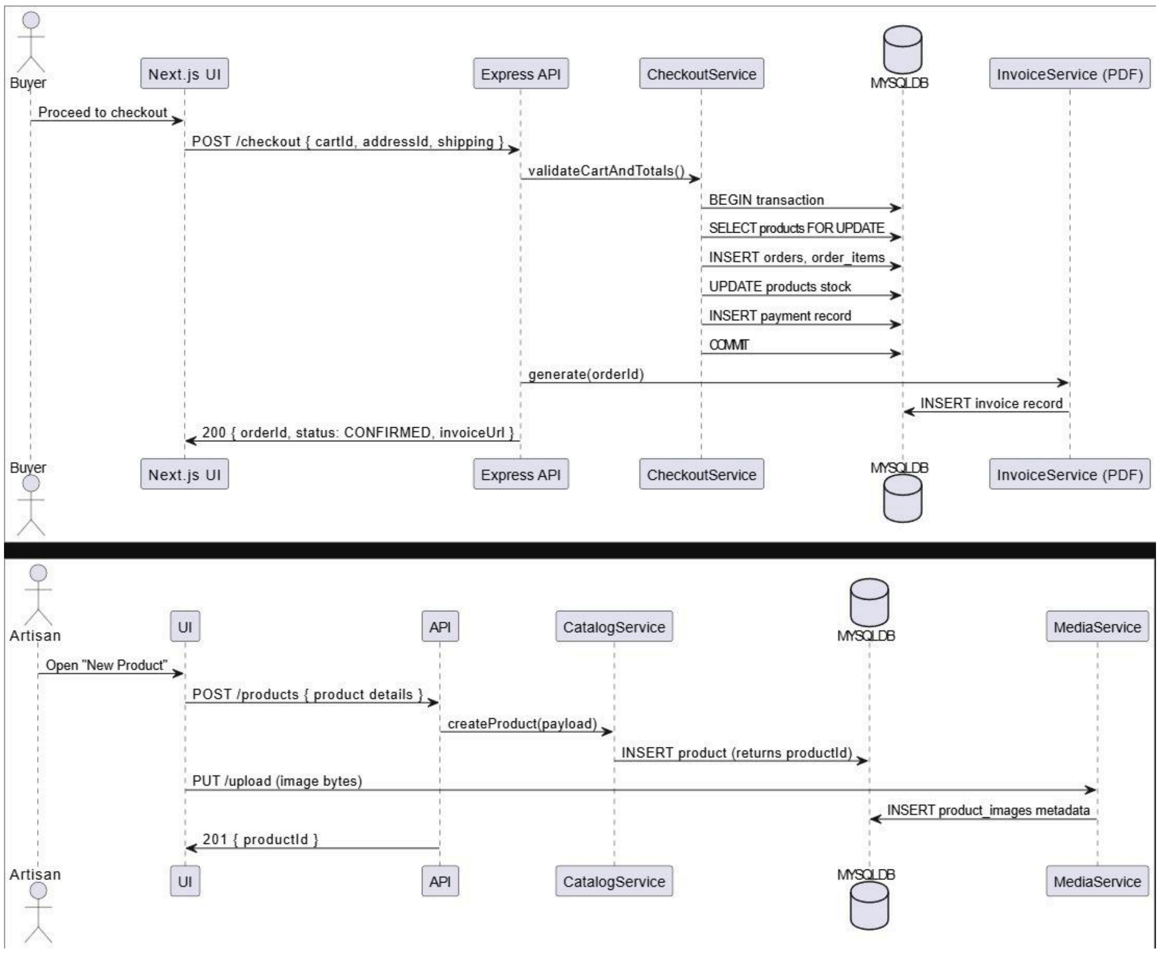
- Information Disclosure → TLS, hashed passwords
- Denial of Service → Rate limiting, input throttling
- Elevation of Privilege → RBAC for admin/seller roles

4. Design

4.1 Design Overview

The marketplace is designed using a layered architecture, separating the user interface, API, business logic, and data layers to ensure modularity, scalability, and easier maintenance.

4.2 UML Sequence Diagrams



4.3 API Design

Interface definitions for two components:

Endpoint: `/checkout`

Method: POST
Request:

```
{
  "cartId": "c123",
  "addressId": "a456",
  "shippingMethod": "standard"
}
```

Response:

```
{
  "orderId": "o789",
  "status": "CONFIRMED",
  "invoiceUrl": "https://example.com/invoices/o789.pdf"
}
```

Errors:

- 400 Invalid cart or missing address
- 402 Payment failed
- 409 Out-of-stock item

Endpoint: `/products`

Method: POST
Request:

```
{
  "title": "Handmade Wooden Bowl",
  "priceCents": 2500,
  "stock": 10,
  "categoryIds": [1, 4]
}
```

Response:

```
{
  "productId": "p321",
  "status": "draft"
}
```

Errors:

- 400 Invalid product data
- 401 Unauthorized
- 413 Image file too large

4.4 Error Handling, Logging & Monitoring

Consistent error responses with structured error codes and messages

No logging of sensitive information (e.g., passwords, tokens)

Monitoring includes API error rate, payment failure rate, and system uptime

Centralized logging for traceability and debugging

4.5 UX Design

Clean, responsive interface for both desktop and mobile users

Accessible color contrast, readable font sizes, and keyboard navigation support

Seller dashboard and product management UI optimized for usability

4.6 Open Issues & Next Steps

Future features:

- Seller analytics dashboard
- Advanced search filters for buyers
- In-app messaging between buyers and sellers
- Mobile app version (iOS/Android)

Pending decisions:

- Integration with a second payment provider
- Moderation flow for product approval

5. Appendices

5.1 Glossary:

SKU – Stock Keeping Unit

JWT – JSON Web Token

PCI-DSS – Payment Card Industry Data Security Standard

TLS – Transport Layer Security

ADR – Architecture Decision Record

5.2 References: IEEE 42010, OWASP, NIST SP 800-160.
5.3 Tools: PlantUML, draw.io