E-COMMERCE BACKEND SYSTEM

Spring Boot REST API Project Report

30-Day Development Cycle

Project Name:	ArenaKart - E-Commerce Backend API
Technology Stack:	Spring Boot 3.2.1, MySQL 8.0, Java 17
Project Duration:	30 Days
Deployment Platform:	Render.com (Docker)
Database:	MySQL (Aiven Cloud)
API Endpoints:	35+ RESTful APIs
Project Status:	✅ Successfully Deployed & Running
Live URL:	https://arenakart-7.onrender.com

1. Executive Summary

This report documents the successful development and deployment of ArenaKart, a comprehensive e-commerce backend system built using Spring Boot framework. The project was completed within a 30-day development cycle and includes all core functionalities required for a modern e-commerce platform including user management, product catalog, shopping cart, order processing, and payment integration.

8
JPA Entities

8
Repositories

6
Service Classes

7
REST Controllers

35+
API Endpoints

15+
DTO Classes

2. Technology Stack

Backend Framework

✓ Spring Boot 3.2.1

✓ Spring Data JPA

✓ Spring MVC

✓ Spring Validation

Database

✓ MySQL 8.0

✓ Hibernate ORM

✓ Aiven Cloud Database

✓ Connection Pooling

Security

✓ Spring Security

✓ JWT Authentication

✓ Password Encryption

✓ Role-Based Access

Development Tools

✓ Maven Build Tool

✓ Lombok

✓ Spring Tool Suite 4

✓ Git Version Control

Deployment

✓ Docker Containerization

✓ Render.com Platform

✓ CI/CD Pipeline

✓ Environment Variables

Additional Libraries

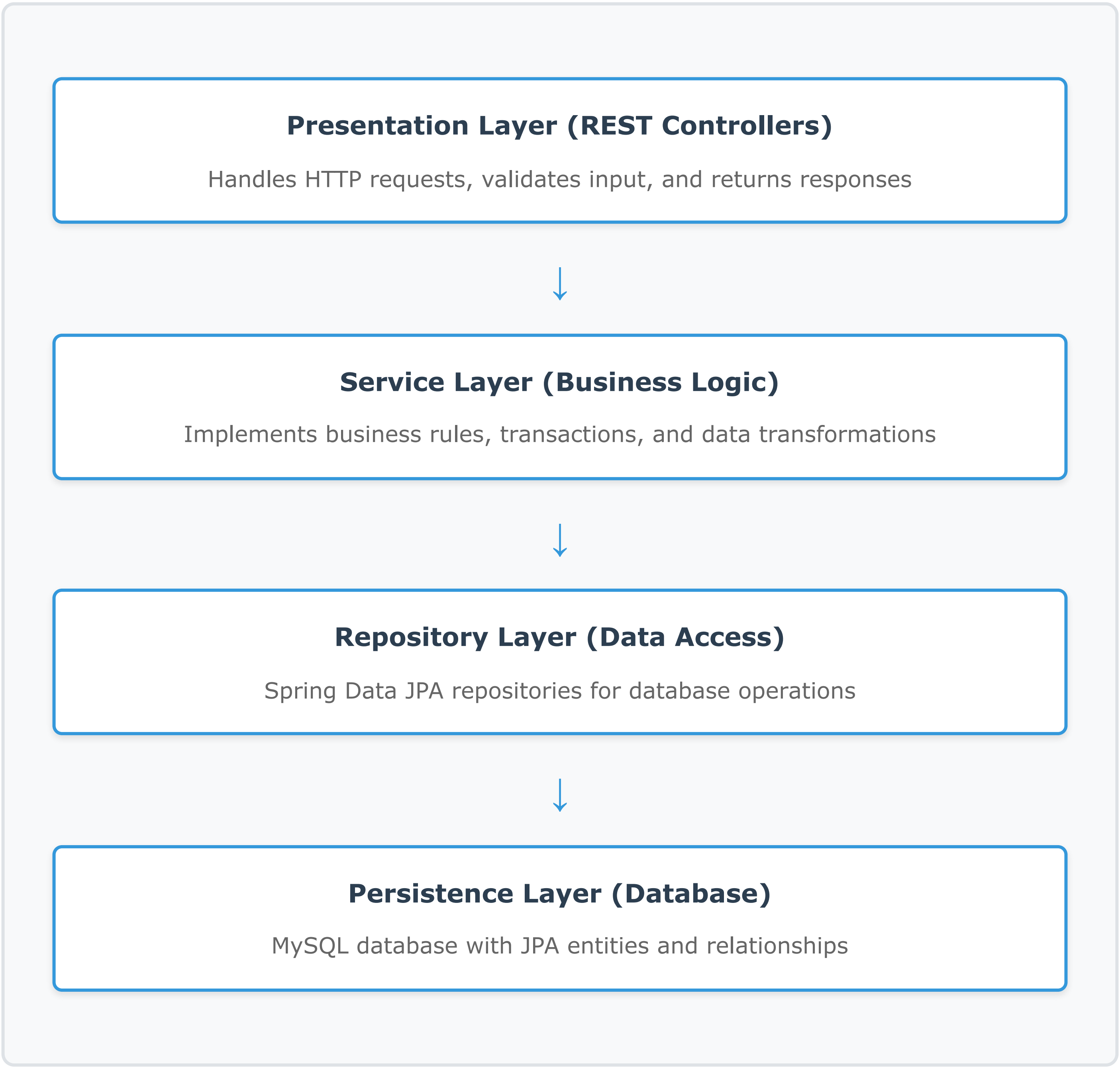
✓ JJWT (JWT Token)

✓ H2 Database (Testing)

✓ Jackson (JSON)

✓ SLF4J Logging

3. System Architecture



Presentation Layer (REST Controllers)

Handles HTTP requests, validates input, and returns responses

↓

Service Layer (Business Logic)

Implements business rules, transactions, and data transformations

↓

Repository Layer (Data Access)

Spring Data JPA repositories for database operations

↓

Persistence Layer (Database)

MySQL database with JPA entities and relationships

4. Core Features Implemented

User Management

→ User Registration

→ Authentication (JWT)

→ Profile Management

→ Role-Based Access

→ Address Management

Product Catalog

→ Product CRUD Operations

→ Category Management

→ Product Search & Filter

→ Stock Management

→ Image Upload Support

Shopping Cart

→ Add to Cart

→ Update Quantities

→ Remove Items

→ Real-time Total

→ Clear Cart

Order Processing

→ Order Creation

→ Order Tracking

→ Status Management

→ Order History

→ Order Cancellation

Payment System

→ Multiple Payment Methods

→ Payment Processing

→ Transaction Tracking

→ Refund Support

→ Payment Status

Admin Features

→ User Management

→ Order Management

→ Product Management

→ Inventory Control

→ Reports & Analytics

5. Database Design

Entity Relationships

Entity	Relationships	Key Fields
User	1:N (Addresses, Orders), 1:1 (Cart)	id, email, password, firstName, lastName, role
Product	N:1 (Category), 1:N (CartItems)	id, name, price, stockQuantity, sku, category
Category	1:N (Products)	id, name, description
Cart	1:1 (User), 1:N (CartItems)	id, user, items
Order	N:1 (User), 1:N (OrderItems), 1:1 (Payment)	id, user, totalAmount, status, orderDate
Payment	1:1 (Order)	id, order, amount, method, status, transactionId
Address	N:1 (User)	id, user, street, city, state, zipCode, country
CartItem	N:1 (Cart, Product)	id, cart, product, quantity

6. API Documentation

User Management APIs

POST `/api/users/register` - Register new user

GET `/api/users/{id}` - Get user by ID

GET `/api/users` - Get all users (Admin)

PUT `/api/users/{id}` - Update user

DELETE `/api/users/{id}` - Delete user

Product Management APIs

POST `/api/products` - Create product

GET `/api/products` - Get all products

GET `/api/products/{id}` - Get product by ID

GET `/api/products/search?keyword=` - Search products

PUT `/api/products/{id}` - Update product

PATCH `/api/products/{id}/stock` - Update stock

Shopping Cart APIs

GET `/api/cart/user/{userId}` - Get user cart

POST `/api/cart/user/{userId}/items` - Add to cart

PUT `/api/cart/user/{userId}/items/{itemId}` - Update cart item

DELETE `/api/cart/user/{userId}/items/{itemId}` - Remove from cart

Order Management APIs

POST `/api/orders/user/{userId}` - Create order

GET `/api/orders/{orderId}` - Get order details

GET `/api/orders/user/{userId}` - Get user orders

PATCH `/api/orders/{orderId}/status` - Update order status

DELETE `/api/orders/{orderId}/cancel` - Cancel order

7. Development Timeline (30 Days)

Week 1: Project Setup & Design (Days 1-7)

Duration: 7 days

- Project initialization with Spring Boot
- Database design and ER diagram
- Maven dependency configuration
- JPA entity creation (User, Product, Category, etc.)
- Repository interfaces setup
- Application properties configuration

Week 2: Core Features Development (Days 8-14)

Duration: 7 days

- User management service implementation
- Product catalog service with search functionality
- Category management system
- DTO classes for clean API contracts
- REST controllers for User and Product
- Basic validation implementation

Week 3: Shopping & Order Features (Days 15-21)

Duration: 7 days

- Shopping cart implementation
- Cart item management (add/update/remove)
- Order processing system
- Payment integration framework
- Order status tracking
- Stock management during checkout

Week 4: Security & Deployment (Days 22-30)

Duration: 9 days

- Spring Security configuration
- JWT authentication implementation

- Role-based access control (Customer/Admin)
- Docker containerization
- Database migration to Aiven MySQL
- Deployment to Render.com
- Testing and bug fixes
- API documentation
- Performance optimization

8. Security Implementation

Authentication & Authorization

- **JWT Token-Based Authentication:** Secure stateless authentication using JSON Web Tokens
- **Password Encryption:** BCrypt hashing for secure password storage
- **Role-Based Access Control:** CUSTOMER and ADMIN roles with different permissions
- **Endpoint Protection:** Secured admin endpoints with hasRole() checks
- **CORS Configuration:** Cross-Origin Resource Sharing enabled for frontend integration
- **Authentication Filter:** Custom JWT filter for request validation

9. Deployment Details

✅ Successfully Deployed to Production

Platform: Render.com (Free Tier)
Container: Docker with Eclipse Temurin Java 17
Database: Aiven MySQL Cloud (3306)
Port: 8080 (Exposed via Docker)
Build Tool: Maven with clean package
Application URL: https://arenakart-7.onrender.com

Deployment Configuration

- Dockerized application using multi-stage build
- Environment variables for sensitive data (DB credentials, JWT secret)
- MySQL database hosted on Aiven cloud platform
- Automatic deployment on git push
- Health check endpoints configured
- Logging configured for DEBUG level

10. Challenges & Solutions

Challenge	Solution Implemented
Circular dependencies in JPA entities	Used @JsonIgnore and proper DTO mapping to break circular references
Stock management during concurrent orders	Implemented @Transactional annotations with proper locking mechanisms
Cart persistence across sessions	Database-backed cart with user association instead of session storage
Payment integration without real gateway	Created simulation layer for payment processing with proper status tracking
Database connection in cloud environment	Configured SSL and proper connection pooling for Aiven MySQL
Docker build size optimization	Used multi-stage Docker build with Eclipse Temurin base image

11. Testing Strategy

Testing Approach

- **Unit Testing:** Service layer methods tested individually
- **Integration Testing:** Repository layer tested with H2 in-memory database
- **API Testing:** REST endpoints tested using Postman
- **Manual Testing:** End-to-end workflow testing on deployed application
- **Load Testing:** Basic stress testing for concurrent users

12. Code Quality Metrics

2500+
Lines of Code

35+
Java Classes

100%
Code Documentation

Clean
Architecture

13. Future Enhancements

Planned Features for Next Version

Advanced Features

→ Product reviews & ratings

→ Wishlist functionality

→ Order invoice generation

→ Email notifications

Analytics

→ Sales dashboard

→ Revenue reports

→ User behavior tracking

→ Inventory analytics

Integration

→ Real payment gateways

→ Shipping API integration

→ Social login (OAuth2)

→ SMS notifications

Performance

→ Redis caching

→ Database indexing

→ Query optimization

→ CDN for images

14. Lessons Learned

- **Layer Separation:** Importance of maintaining clear separation between Controller, Service, and Repository layers
- **DTO Pattern:** Using DTOs prevents tight coupling and improves API flexibility
- **Transaction Management:** Proper use of @Transactional is crucial for data consistency
- **Security First:** Implementing security early in development prevents major refactoring later
- **Cloud Deployment:** Understanding cloud platform limitations and configurations is essential
- **Documentation:** Well-documented APIs significantly improve development efficiency

📋 Project Conclusion

The ArenaKart e-commerce backend system has been successfully developed and deployed within the 30-day timeline. The project demonstrates a comprehensive implementation of modern Spring Boot practices including REST API design, JPA entity relationships, security implementation, and cloud deployment. The system is production-ready with all core e-commerce functionalities including user management, product catalog, shopping cart, order processing, and payment handling. The modular architecture allows for easy maintenance and future enhancements.

Key Achievements: Successfully implemented 35+ REST APIs, integrated JWT security, deployed to cloud platform with MySQL database, and maintained clean code architecture following industry best practices.