Online Bookstore Microservices Project

Checkpoint 2: Microservices Implementation and Dockerization

Prepared For: Prof. Pedram Habibi
Prepared By: SOA915-NAA-Group9
Members:

Helly Rajeshbhai Patel
Jiyad Mohammed
Arif Shaikh
Nicholas Nwanua Ilechie
Nirajbhai Ranchhodbhai Limbasiya

Date Submitted: June 17th, 2025

https://github.com/SOA915-NAA-GROUP9/online-bookstore-microservices

Introduction

This report presents the deliverables for Checkpoint 2 of the Service-Oriented Architecture (SOA) course project, where we have successfully implemented and containerized two fundamental microservices: the User Service and Product Service. These services form the architectural foundation of an online bookstore application, enabling user authentication and book catalog management through RESTful APIs, Docker containerization with multi-stage builds, Docker Compose orchestration, integration testing, and comprehensive API documentation. This report systematically examines the technical deliverables, demonstrates inter-service communication patterns, and evaluates progress toward establishing a scalable microservices architecture that supports distributed e-commerce operations.

Microservices Implementation

A. User Service:

The User Service handles user registration and profile retrieval, implemented in Node.js with Express.js and in-memory storage (to be replaced with a database in Phase 2).

The service supports the following RESTful APIs:

- POST /users/register: Registers a new user with username and email.
- GET /users: Retrieves all users (added for consistency with Product Service).
- GET /users/{id}: Fetches details of a specific user by ID. Example code for GET /users (user-service/src/index.js)
    - app.get('/users', (req, res) => { res.json(users);});

B. Product Service

The Product Service manages the book catalog, also implemented in Node.js with Express.js.

It supports:

- GET /books: Lists all books in the catalog.
- GET /books/{id}: Retrieves details of a specific book by ID. Example code for GET /books (product-service/src/index.js):
    - app.get('/books', (req, res) => { res.json(books); });

Dockerization

Each service is containerized using a multi-stage Dockerfile to optimize image size. The Dockerfiles use node:18-alpine for both build and runtime stages, copying only production dependencies.

Example (user-service/Dockerfile):

- FROM node:18-alpine AS builder
- WORKDIR /app
- COPY src/package*.json ./

https://github.com/SOA915-NAA-GROUP9/online-bookstore-microservices

- RUN npm install --production
- COPY src/ .

- FROM node:18-alpine
- WORKDIR /app
- COPY --from=builder /app .
- EXPOSE 3000
- CMD ["npm", "start"]

The docker-compose.yml file orchestrates both services for local testing:

version: '3.8'

Services:

- user-service:
    - build: ./user-service
    - ports:
        - "3000:3000"
    - environment:
        - NODE_ENV=production
- product-service:
    - build: ./product-service
    - ports:
        - "3001:3001"
    - environment:
        - NODE_ENV=production

API Documentation

The API contracts have been updated to include the new GET /users endpoint, ensuring consistency with GET /books. The documentation is available in docs/api-contracts.md. Example:

• GET /users: Lists all users.

– Response: [”id”: ”string”, ”username”: ”string”, ”email”: ”string”

]

– Example: curl http://localhost:3000/users

GitHub Repository

The project is hosted at https://github.com/SOA915-NAA-GROUP9/online-bookstore-microservices

Demonstration

For the lab, we will:

• Run docker compose up –build to start services.

• Demonstrate API calls using Postman:

– Postman: POST http://localhost:3000/users/register …

– Postman: http://localhost:3000/users

– Postman: http://localhost:3001/books

• Run npm test in tests/ to show integration tests.

• Showcase the GitHub repository, highlighting branches and documentation.

https://github.com/SOA915-NAA-GROUP9/online-bookstore-microservices

Conclusion

Checkpoint 2 has successfully delivered two functional microservices—User and Product Services—with comprehensive RESTful APIs, containerized through Docker for streamlined local testing via Docker Compose orchestration. Integration testing validates inter-service communication patterns, while enhanced documentation ensures technical clarity and maintainability. The project remains on schedule for Phase 2 implementation, which will encompass the development of Order and Notification Services alongside Kubernetes deployment for production-ready orchestration. Constructive feedback is welcomed to further refine the architectural approach and implementation methodology.

https://github.com/SOA915-NAA-GROUP9/online-bookstore-microservices