Online Bookstore Microservices Project

Checkpoint 4: Testing, CI/CD, and Logging

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: July 31st, 2025

“We declare that the attached assessment is wholly my own work in accordance with Seneca Academic Policy. No part of this assignment has been copied manually or electronically from any other source (including web sites) or distributed to other students.”

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

Introduction

This report presents the deliverables for Checkpoint 4 for our Service-Oriented Architecture (SOA) course project, we implemented unit, integration, and end-to-end (E2E) tests for the User and Product Services, set up a GitHub Actions CI/CD pipeline, and configured centralized logging using Fluentd. Building on the Kubernetes deployment from Checkpoint 3 within the bookstore namespace, we addressed testing and workflow issues to ensure a reliable pipeline and logging system.

Testing

We developed unit, integration, and E2E tests to ensure the reliability of the User and Product Services.

Unit Tests

Unit tests validate individual functions in user-service and product-service. For example, userController.test.js tests user registration and retrieval logic.

Example (user-service/src/__tests__/userController.test.js):

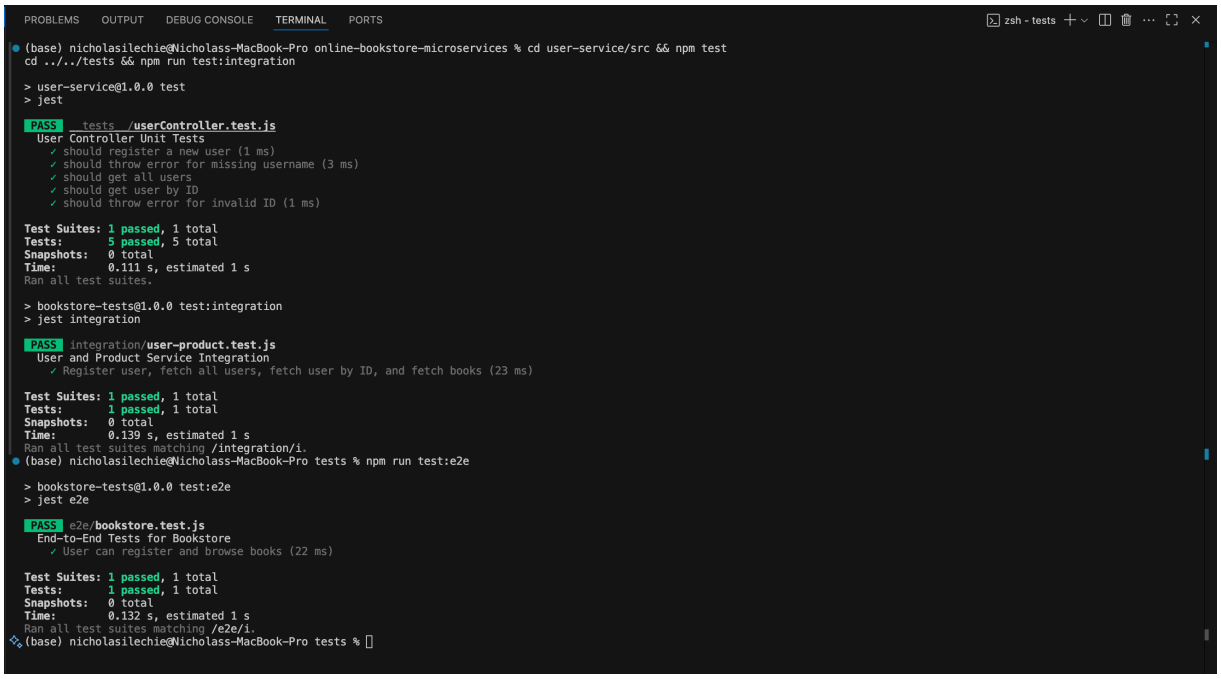
Output: All unit tests passed after fixing the user array reset logic see screenshot below

Integration Tests

Integration tests verify interactions between services, such as registering a user and fetching books (tests/integration/user-product.test.js). Tests use Axios to call service endpoints.

End-to-End Tests

E2E tests simulate user workflows, such as registering a user and browsing books (tests/e2e/bookstore.test.js). All tests passed after ensuring services were running via Docker Compose.



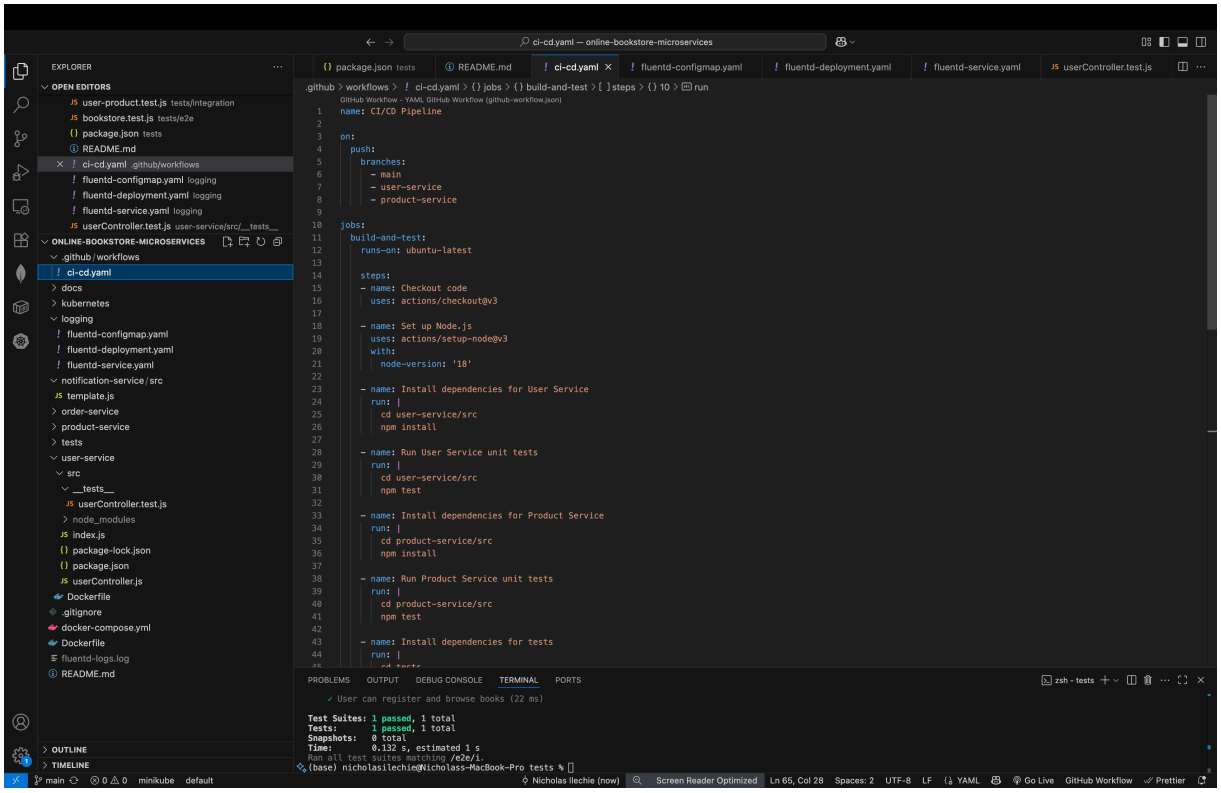
```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS                                   zsh - tests

(base) nicholasilechie@Nicholass-MacBook-Pro online-bookstore-microservices % cd user-service/src && npm test
cd ../../tests && npm run test:integration

> user-service@1.0.0 test
> jest

 PASS  __tests__/userController.test.js
  User Controller Unit Tests
    ✓ should register a new user (1 ms)
    ✓ should throw error for missing username (3 ms)
    ✓ should get all users
    ✓ should get user by ID
    ✓ should throw error for invalid ID (1 ms)

Test Suites: 1 passed, 1 total
Tests:       5 passed, 5 total
Snapshots:   0 total
Time:        0.111 s, estimated 1 s
Ran all test suites.

> bookstore-tests@1.0.0 test:integration
> jest integration

 PASS  integration/user-product.test.js
  User and Product Service Integration
    ✓ Register user, fetch all users, fetch user by ID, and fetch books (23 ms)

Test Suites: 1 passed, 1 total
Tests:       1 passed, 1 total
Snapshots:   0 total
Time:        0.139 s, estimated 1 s
Ran all test suites matching /integration/i.
(base) nicholasilechie@Nicholass-MacBook-Pro tests % npm run test:e2e

> bookstore-tests@1.0.0 test:e2e
> jest e2e

 PASS  e2e/bookstore.test.js
  End-to-End Tests for Bookstore
    ✓ User can register and browse books (22 ms)

Test Suites: 1 passed, 1 total
Tests:       1 passed, 1 total
Snapshots:   0 total
Time:        0.132 s, estimated 1 s
Ran all test suites matching /e2e/i.
(base) nicholasilechie@Nicholass-MacBook-Pro tests %
```

Figure 1: Test Results

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

CI/CD Pipeline

We configured a GitHub Actions workflow to build Docker images, run tests, and prepare for deployment. The pipeline triggers on pushes to main, user-service, and product-service branches. Example (.github/workflows/ci-cd.yaml)

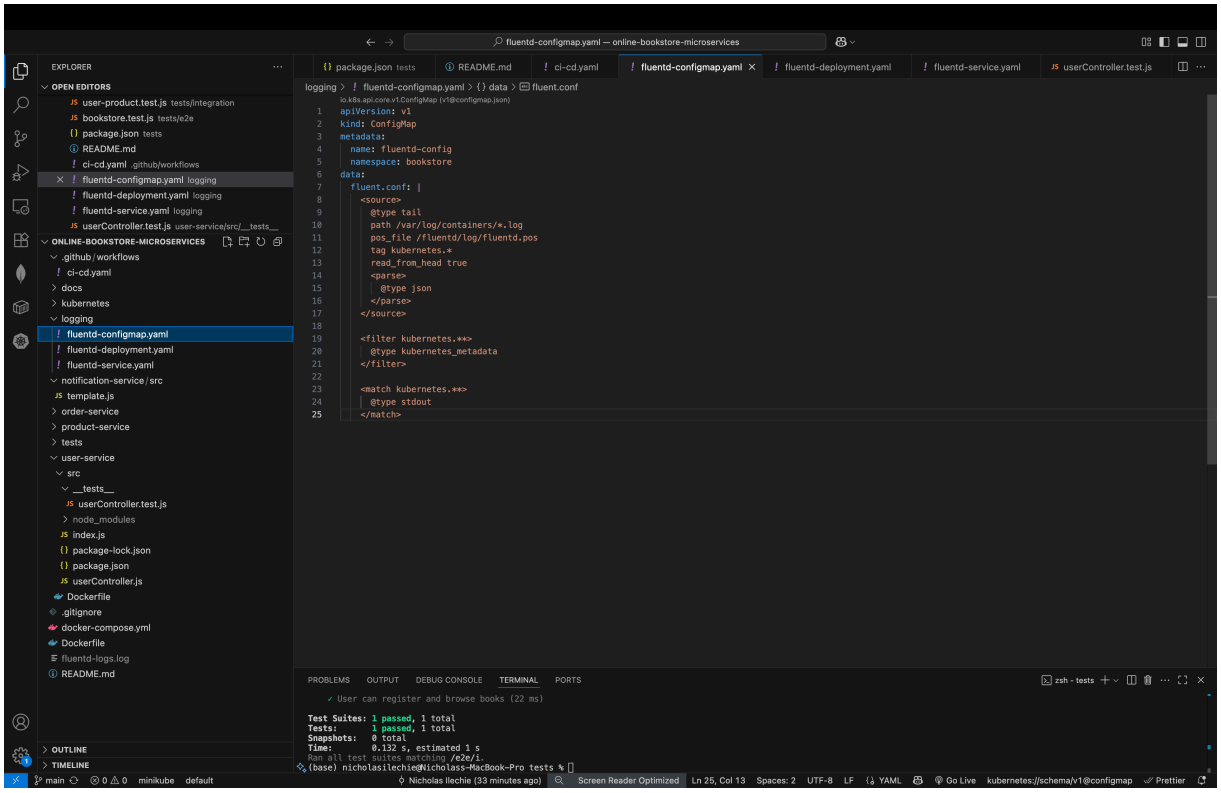


The workflow was fixed to remove Minikube deployment and ensure services run during tests, achieving a successful run

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

Logging

Fluentd aggregates logs from all pods in the bookstore namespace, outputting to stdout for simplicity. The setup includes a ConfigMap, Deployment, and Service. Example (logging/fluentd-configmap.yaml):



Logs from user-service and product-service were captured successfully after re⍰solving Minikube API server issues

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

Conclusion

Checkpoint 4 successfully implements testing, CI/CD, and centralized logging for the User and Product Services. Test failures were resolved, the GitHub Actions workflow was refined, and Fluentd was deployed to enable robust log management. With these enhancements, the system is stable and ready for Phase 2, which involves adding Order and Notification Services and deploying the application to a cloud environment. Feedback is welcomed to further improve the implementation.