Integration Testing

Course: Software Testing

Worked by: Evisa Nela

Accepted by: Ari Gjerazi, Jurgen Cama

Integration Test 1: Product Stock Update and Verification

This test is classified as Database Integration Testing and follows a Decomposition-Based Integration Testing approach, as it validates the interaction between the Product module and the MySQL database through JDBC by executing multiple dependent database operations in sequence.

Test Objective

The objective of this integration test is to verify that the Product module correctly interacts with the database when performing a full product lifecycle operation. This includes inserting a product, retrieving it, updating its stock quantity, verifying the update, and finally deleting the product. The test ensures that multiple database operations work together correctly.

Integration Strategy

Strategy used: Big Bang Integration (within Product module)

All related product operations were integrated and tested together in a single test case. This approach was chosen because the product-related functionalities are closely connected and depend on the same database table.

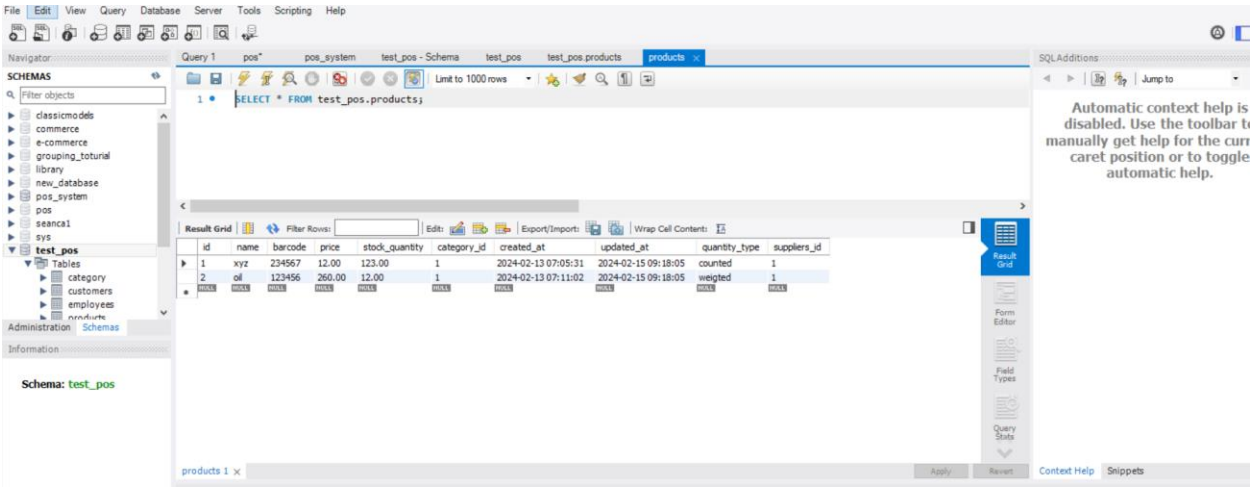
This integration test validates the correct interaction between the Product module and the database layer using JDBC. Each operation is executed against a real MySQL test database, ensuring that data persistence, retrieval, update, and deletion behave as expected. The test confirms that database state changes are accurately reflected after each operation, providing confidence in the reliability of the product stock management functionality.

Integrated Components

- addProduct (INSERT into products table)
- getProducts / SELECT product by ID
- updateProductStock (UPDATE stock quantity)
- deleteProduct (DELETE product)
- MySQL database (products table)

Database Details

- **DBMS:** MySQL
- **Database name:** test_pos
- **Table:** products
- **Connection:** JDBC (DriverManager.getConnection)



Test Environment

- IDE: IntelliJ IDEA
- Language: Java
- Testing Framework: JUnit 5
- Database Tool: MySQL Workbench

Preconditions

- MySQL server is running.
- Database test_pos exists.
- Table products exists with required columns.
- At least one valid supplier exists in the suppliers table.
- JDBC connection credentials are valid.

Test Objective

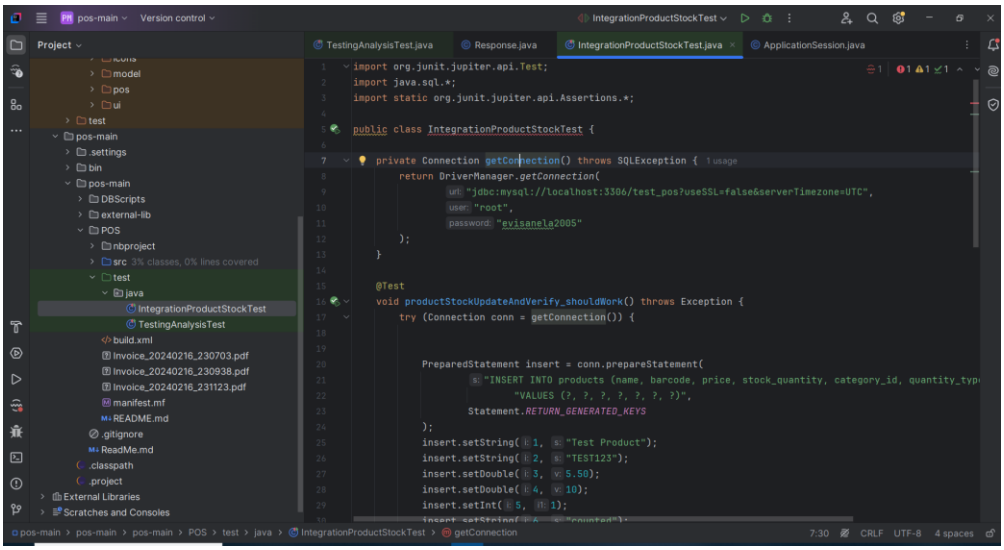
The objective of this test is to verify that product stock quantity can be:

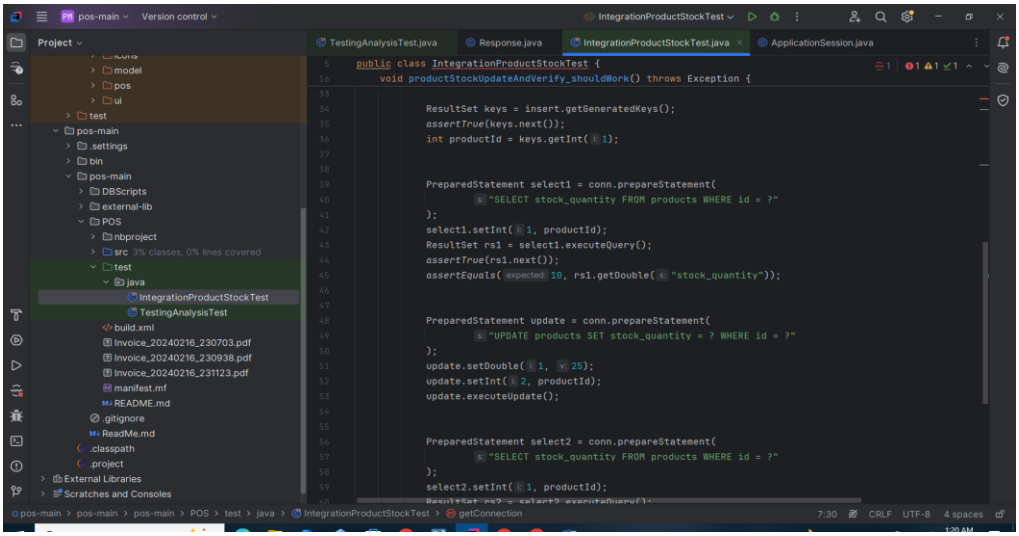
1. Inserted correctly into the database

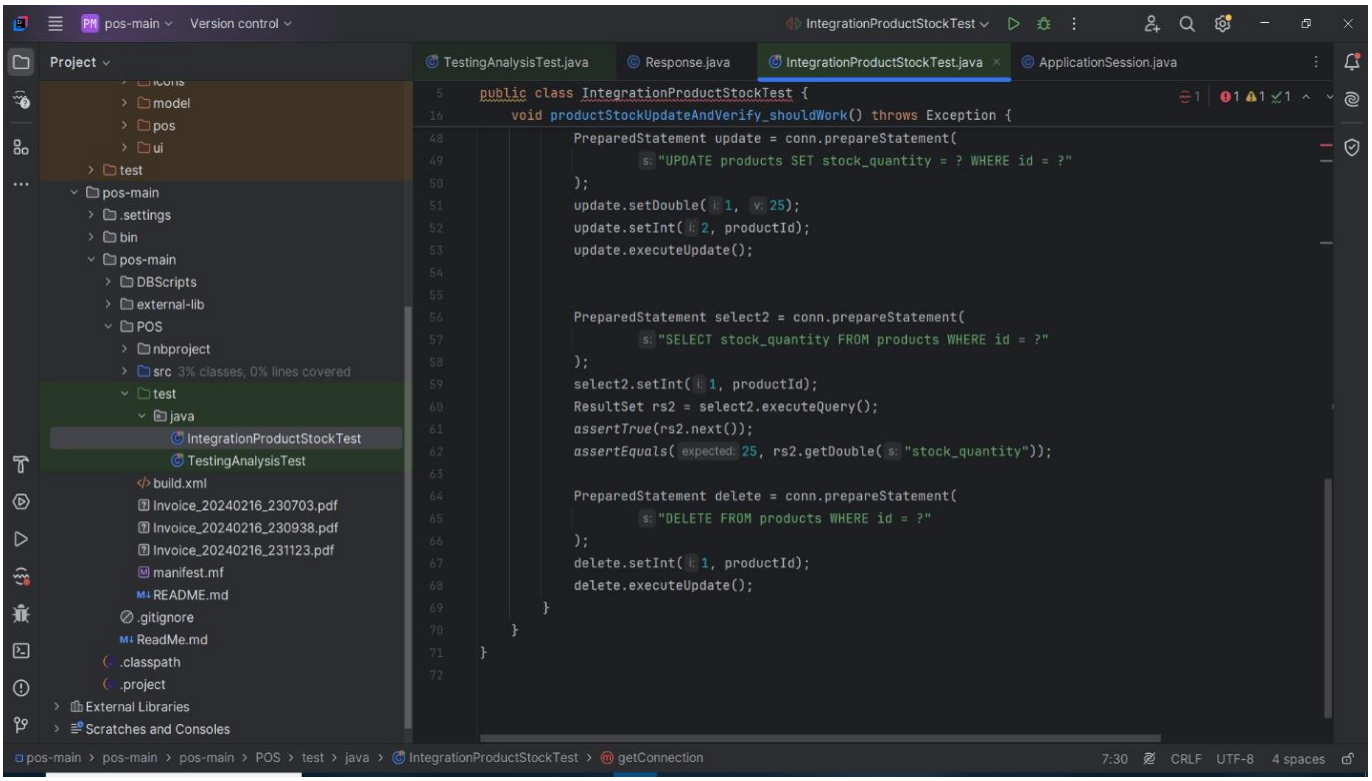
2. Read correctly from the database
3. Updated correctly
4. Verified after update
5. Removed after the test

Test Flow / Steps

1. Insert a new product with:
    - name = *Test Product*
    - price = 5.50
    - stock_quantity = **10**
2. Retrieve the inserted product and obtain its generated ID.
3. Read the stock quantity and verify it is **10**.
4. Update the stock quantity to **25**.
5. Read the product again and verify the stock quantity is **25**.
6. Delete the test product from the database.





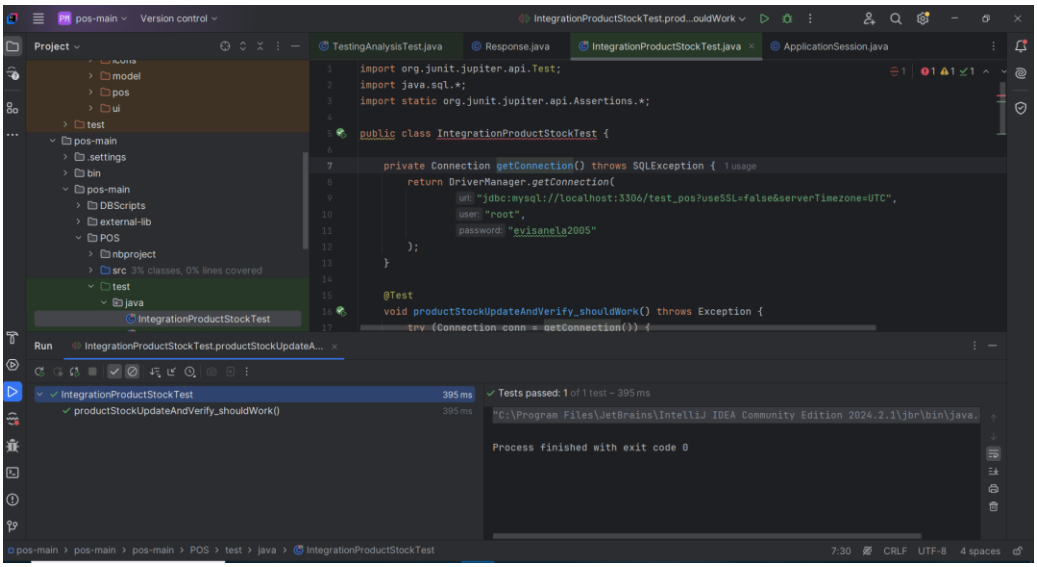


Expected Result

- Product is inserted successfully.
- Product ID is generated and retrieved.
- Initial stock quantity equals **10**.
- Stock quantity updates successfully to **25**.
- Updated value is correctly stored in the database.
- Product is deleted successfully after the test.

Actual Result

- All database operations executed successfully.
- Stock quantity was correctly updated from **10 → 25**.
- Assertions passed without errors.
- Test finished with exit code **0**.



Error Encountered & Resolution

Unknown column 'stockQuantity'

During testing, an SQLSyntaxErrorException occurred due to a mismatch between Java column naming (stockQuantity) and the actual database column name (stock_quantity).
The issue was resolved by aligning SQL queries with the database schema.

Conclusion

This integration test confirms that the product stock update functionality works correctly across the application and database layers. The test verifies correct data insertion, update, retrieval, and cleanup, ensuring reliable interaction between Java application logic and the MySQL database.