Order Management API Testing

Project Description for PB Software Development – Software Quality (Spring 2025)

Martin Grulyo

1 Project Description

The focus of this project is the development and testing of a small-scale Order Management API built using ASP.NET Core. The API supports a simplified e-commerce workflow and is intended as a platform for applying various software testing techniques taught in the Software Quality course.

The API supports the following core functionalities:

- User registration (basic for testing purposes)
- Product listing and stock management
- Placing orders, which are created through a 4-step workflow:
  1. **Order Created**
  2. **Order Processed**
  3. **Order Out for Delivery**
  4. **Order Delivered**
- Business Rule: Orders placed during the weekend (Saturday or Sunday) will not be processed until the following Monday.

The purpose of this API is to serve as a base for applying both theoretical and practical testing techniques, including path testing, black-box API testing, unit testing, and behavior-driven development (BDD) with SpecFlow/Reqnroll.

2 Testing Strategy and Techniques

This project applies a mix of white-box and black-box testing strategies to ensure both internal correctness and external functionality of the system. The following techniques are applied:

2.1 Path Testing (White-box)

Path testing is applied to the core order-processing logic. A program graph is created for the order flow, and test cases are designed to ensure complete node and edge coverage. Special attention is given to decision points, such as validating business rules like the weekend processing delay.

2.2 BDD with Cucumber Testing

Behavior-Driven Development (BDD) scenarios are written in Gherkin syntax and implemented using SpecFlow. These scenarios test business behavior rather than internal implementation.

2.3 Postman Testing (Black-box)

API endpoint functionality is tested using Postman. Test cases are created for:

- Happy paths (valid order workflow)
- Edge cases (e.g., invalid product IDs, out-of-stock scenarios)
- Business rules (e.g., weekend processing logic)

Tests are run in both isolated and sequential environments to ensure correct state transitions and data integrity.

2.4 Unit Testing (White-box)

Unit tests target the business logic in the core layer. These include:

- Stock checking
- Order status transitions
- Validation logic (preventing empty orders)
- Time-based logic (processing orders only on weekdays)

Mocking frameworks like Moq are used to isolate dependencies and simulate external behavior.

Conclusion

This project combines practical development and testing of an API system with the application of theoretical software quality concepts. Each technique contributes to verifying the API from different perspectives: internal logic, user-facing behavior, and edge-case handling. This exercise provides a comprehensive overview of modern testing practices in real-world systems.