Software Testing and Analysis Final Report

Testing Vehicle Inspection APIs in AutoRegistry: A TDD Approach

Faculty: Contemporary Sciences and Technologies, Tetovo
Students: Endrit Mamuti – [em30173@seeu.edu.mk]
Mentor: Prof. Dr. Nuhi Besimi
Date: June 2025

Abstract

The AutoRegistry project simulates a national vehicle inspection system via a .NET-based RESTful API. It allows clients to check the inspection status of a vehicle using a dedicated endpoint. This paper presents a test-driven development (TDD) approach to building and testing the service logic behind this API. Unit tests written in xUnit, paired with Moq for mocking data sources, validate various scenarios such as inspection passed, failed, expired, or not found. Results show that TDD helped shape the system's behavior early, ensuring accuracy and reliability.

1 Introduction

1.1 Project Description

AutoRegistry is a .NET Core API application designed to support technical vehicle inspection operations. The core feature allows querying a vehicle's inspection status based on its ID. The API responds with whether the vehicle passed its inspection, failed (and why), or whether its certificate has expired.

1.2 Importance of Testing

TDD was used to develop the core logic of the system. In vehicle inspection systems, incorrect data can lead to real-world safety issues or service disruptions. TDD helped enforce correctness from the beginning and allowed isolating logic from database dependencies using mocks.

2 Related Work

2.1 Testing Technologies

- **xUnit** was chosen for unit testing due to its simplicity and .NET integration.
- **Moq** was used to mock repository interfaces, enabling isolation of business logic.

- **Swagger (Swashbuckle)** was used to document and test the API interactively.

2.2 Research Context

TDD has been extensively studied:

- Beck, K. (2002). *Test-Driven Development by Example* – foundational text establishing TDD as a design discipline.
- Erdogmus, H., Morisio, M., & Torchiano, M. (2005). *On the effectiveness of the test-first approach to programming.* IEEE Transactions on Software Engineering.

3 Solution

3.1 Testing Strategy

We applied the Red-Green-Refactor TDD cycle:

1. Red: Write failing test
2. Green: Implement just enough code
3. Refactor: Clean up and generalize

3.2 Test Case Design

Test cases covered:

- Valid inspections (passed)
- Failed inspections with reason
- Expired certificates
- Vehicle ID not found

4 Implementation

4.1 Implementation Process

1. Created solution structure with API, Core, and Test projects.
2. Defined model (`VehicleInspection`) and interface (`IVehicleRepository`).
3. Implemented `VehicleInspectionService` with dependency injection.
4. Wrote xUnit tests with Moq for 4 logical paths.
5. Added controller (`VehicleController`) and exposed endpoint: `GET /api/vehicle/{id}/inspec`
6. Used in-memory repository for demonstration.

4.2 Running Tests

Tests are run with:

```
dotnet test
```

The API runs on:

```
dotnet run --project AutoRegistry.API
```

Swagger UI is available at `http://localhost:[PORT]/swagger`

5 Discussion

5.1 Solution Evaluation

Using TDD improved reliability and clarified logic early. All branches of logic were tested before writing the API itself. Mocking made the tests fast and repeatable. The in-memory repo allowed simulating production behavior for testing.

5.2 Project Findings

- Discovered edge cases during test writing
- Improved modularity due to service/test separation
- Learned value of Swagger and Moq in real projects

6 Conclusion

TDD proved highly beneficial for this backend API development. Logic was shaped through tests, and bugs were minimized due to early validation. AutoRegistry demonstrates how government or enterprise systems can benefit from a disciplined test-first approach.

7 References

1. Beck, K. (2002). *Test-Driven Development by Example.*
2. Erdogmus, H., Morisio, M., & Torchiano, M. (2005). *On the effectiveness of the test-first approach to programming.* IEEE Transactions on Software Engineering.
3. Osherove, R. (2013). *The Art of Unit Testing: with examples in C.*