Test doubles

I use Fakes by using an in-memory database to simulate the real database when testing. This means that I can run the tests in a controlled and isolated database which gets reset after each test by the dispose function.

Mutation testing

I have done manual mutation testing by adding small changes to functions and seeing if the unit tests still worked. This helped identify weak spots in tests and ensured more robust tests.

Verification vs. Validation

Verification: Ensures that the product is built according to the requirements. This is usually done with reviews, inspections and unit tests. These are considered static testing methods.

The unit tests verify that the TaskModelController works as expected. They ensure that the function works in isolation.

Validation: Ensures that the product fulfills the user's needs and requirements. This is usually done with system tests, integration tests and acceptance tests. These are considered dynamic testing methods.

The integration tests ensure that the TaskModelController works correctly when including the API. The tests ensure that the controller interacts correctly with the database and the API.

Software Quality Reflection

Som of the thing I could improve about my software quality would be expanding tests to include acceptance tests that simulate user scenarios. I could also add performance tests and usability tests to ensure the application meets quality standards as well as regularly reviewing and changing the tests based on feedback and requirements changing.

Discussion on Test Categories

Martin Fowler’s categories of testing are:

- Unit Tests: Test individual functions in isolation.
- Integration Tests: Test the integration of multiple components.
- System Tests: Test end-to-end functionality.
- Acceptance Tests: Validate the system against user requirements and acceptance criteria.
- Regression Tests: Ensure that changes do not break existing functionality.
- Performance Tests: Test the performance of the system.

The test I have implemented:

Five unit tests which test the different CRUD operations created for tasks in the TaskModelController.

Two integration tests that ensure that the API works with the database and that you can create a task and get a task.

One specification-based test that checks that you cannot create a new TaskModel object with a null title.