Assignment 3: Integration Testing

Overview: While organizations often invest in unit testing, integration testing is similarly critical, though often overlooked. In this assignment, you will discuss the differences between unit and integration testing and apply integration testing to the test system.
General notes: This assignment counts towards Lab 1. Deliver solutions to all exercises as defined in the *deliverables* to pass this assignment. When using material outside of the official course material (lecture, slides, tutorials), state the source and critically reflect on its usage. In case of questions, please contact Zeth Danielsson (zeda21@student.bth.se).
Work distribution: At the beginning of each assignment submission, state how the work was distributed among the two team members.

1. Test Levels

Integration tests are one level above unit tests in the testing pyramid: though still relatively small in scale, they already differ significantly from unit tests in purpose and approach. *Explain the difference between unit and integration tests.*
Deliverables: The submission to this exercise must contain all of the following:

1. An explanation of the difference in scope between unit and integration tests.
2. An explanation of the different purposes that mocking has in unit and integration tests.

2. Integration Testing

Integration testing becomes especially relevant when working with external systems: we can assume that the MongoDB—a widely used database—has undergone sufficient unit testing. However, that does not assure that the communication between our server and the Mongo database works correctly. The data access object (DAO) encapsulates the communication between any component of the EduTask system and the Mongo database. We want to assure that this communication works as specified, such that we can rely on our database operations. For this assignment, we will focus on the `create` method in the DAO object (`src/util/dao.py`).
Creating a new object in a collection of a Mongo database is regulated by validators, which constrain the otherwise very unrestricted process of inserting items into a Mongo database[1]. We want to make sure that—given a configuration specified in a validator—the object creation process will succeed only if the input data is compliant to the validators.
Design and implement integration tests for the communication between the data access object DAO (`backend/src/util/dao.py`) and the Mongo database focusing on the create method. Follow best practices for integration test design and utilize mocking where applicable.
Deliverables: The submission to this exercise must contain all of the following:

1. A list of test cases which have been derived using the test design technique.
2. A pytest fixture allowing interaction with the database without disturbing production code or data.
3. An implementation of the test cases in pytest. Provide a link to the file(s) in your forked repository, which contains your test code.
4. A brief report about the results of test execution, consisting of `pytest`'s console output and a brief statement evaluating it.

[1] More information on PyMongo's validators can be found in the documentation