# **Team Activity – Improving your skills on Dependency injection and testing**

### **Goals**

* Explore an existing ASP.NET Core MVC project.
* Practice teamwork: analyze, discuss, and assign tasks.
* Identify design problems using SOLID principles.
* Refactor to improve the code.
* Write unit tests to verify functionality.

## **Step 1: Setup & Run the Project**

1. Download the StudyRoomMvc starter project
2. Create one share github repository for your team (Add me as a contributor: [audrey.nangue@googlemail.com](mailto:audrey.nangue@googlemail.com)) and push the starter project in that repository
3. Each team member should clone that repository
4. Update the connection string in appsettings.json with your own Azure SQL Database (refer to connection string format used in Web VI).
5. Create the required database and ensure migrations can run.
6. Run the project and confirm that:
   1. You can list bookings.
   2. You can create, edit, delete, and view details of a booking.

## **Step 2: Analyze the Code**

1. Explore the Controllers, Services, Repositories, Models, and DbContext.
2. Individually first, then as a team, review the code and identify violations of SOLID principles.
3. Document the issues you find on a shared doc (one doc for the group). Add screenshots. Explain how to fix them.

(share the link with me)

## **Step 3: Team Discussion & Task Breakdown**

1. Hold a short team meeting to discuss:
   1. What problems did we identify?
   2. How can we fix them?
   3. Which fixes are priorities?
2. Break the work into clear tasks.
3. Assign tasks to each team member.

## **Step 4: Refactor the Code**

1. Apply the fixes you agreed on as a team.
2. Use SOLID principles as your guide.
3. Make sure the project still builds and runs correctly after each change.

## **Step 5: Write Unit Tests**

Write unit tests for different layers:

* Model Layer:
  + Validate that model properties respect annotations (e.g., required fields, valid dates).
* Service/Repository Layer:
  + Test adding, updating, retrieving, and deleting a booking.
  + Use mocking frameworks (e.g., Moq) where needed.
* Controller Layer:
  + Test that each action returns the correct view or redirect.
  + Verify model validation is respected.

Each test must assert meaningful conditions that confirm the feature works as expected.

## **Step 6: Verify as a Team**

1. Run all tests and ensure they pass.
2. Demo the refactored project to your teammates.
3. Review together:
   1. Did we remove the violations?
   2. Do our tests prove that the system works?

## **Deliverables**

* Refactored project code (committed to your repo).
* A suite of unit tests covering Models, Services/Repositories, and Controllers.
* A short summary document:
  + Problems found.
  + Fixes applied.
  + Testing strategy.