In this task, we need to refactor the code of an existing application. Upon opening the solution, we will see two projects:

- LegacyApp - this is the application that we will want to refactor.
- LegacyAppConsumer - this is an example of an application that utilizes LegacyApp.

Remember that refactoring means that we do not change the operation of the existing application. We assume that the application works correctly. We need to refactor the UserService class along with the AddUser method. Note: in LegacyApp, you will find classes that simulate querying external data sources through the use of Thread.Wait.

- Remember, the application should function the same as it currently does after the refactoring process.
- During refactoring, you may modify any files in LegacyApp except for the UserDataAccess class. This class represents an example of a legacy library, which for various reasons, we cannot edit.
- Keep in mind that the code in the LegacyAppConsumer application must still compile and work - also after the refactoring process. We do not want the code of other applications to suddenly stop working after our refactoring. We also cannot modify the code from this project in any way.
- Follow the SOLID principles, code testability, and its readability.
- Try to control the structure of the program, remembering about cohesion and coupling metrics.
- Try to use unit tests in the solution.