====== Team's members ======
Aravind Kumar Machiry

====== What the team completed ======
We developed a solution with C# in .NET 4, that is in compliance with all the requirements and provides additional features such as:
- UI interface for the client
- Multiple clients and possibility to specify a limit of clients currently connected
- Multiple connections for each user
- Exceptions handling
- Possibility to add users into the database

====== Cryptographic algorithm used ======
The algorithm used have been taken from the System.Security.Cryptography namespace. They are:
- SHA-256, as hash function
- AES-128, for the symmetric encryption

====== Usernames' list and password in the database ======
The application SecureChat_InsertUser.exe allows to add users to the database. Invoke the application without parameters to get instructions.
The database is pre-filled with ready-to-use users. They are specified below as (username, password):
Sample Dummy Users:
- (user2, password2)
- (user3, password3)
- (user4, password4)
- (user5, password5)

Please make sure that you create a new file with the user name and password.

====== Step-by-step instructions to build the project ======
The project has been done in Visual Studio Ultimate with C#. It is composed of 3 solutions:
- ChatterServer.sln
- ChatterClient.sln
- ChatterInsertUser.sln

That can be opened with Visual Studio and built. The build configuration is contained in the solution.

====== How to run the project ======
The files provided are ready-to-run supposing that .NET 4 is installed in the machine. For each application:
- SecureChatServer.exe

- SecureChatClient.exe
- SecureChat_InsertUser.exe

Is sufficient to run the software without arguments to obtain contextual help and instructions.

A scenario to get started could be:
- Use SecureChat_InsertUser to insert one own's user;
- Start SecureChatServer listening to a port;
- Start as many SecureChatClient as needed to that port: note SecureChatClient accepts only IPs and not aliases (such localhost).

A concrete example supposing to be (user1, password1) and that the port 8020 is available, is issuing the following commands in two different prompt:
- SecureChatServer 8020
- SecureChatClient 127.0.0.1 8020