Survey Configurator Requirements

The task is to create a Survey questions configurator application, as in the following requirements:

1) It should be developed using .NET framework as desktop application with SQL server DB.

2) All application configuration should be saved in the DB, so when the application is closed and reopened, it should reflect the latest status.

3) The main screen content should have the following:
   1. A list of already created questions (loaded from the DB). The question text for each question should appear in the list.
   2. The list should support selection and ordering.
   3. Three buttons:
      1. Add: it will open a dialog form to create a new questions
      2. Edit: it will open a dialog form to edit the properties of the selected question from the list.
      3. Delete: it will delete the selected question from the list (after asking the user for confirmation).

4) There should be three types of questions:
   1. Smiley faces question. it has the following properties:
      1. Question text
      2. Question order
      3. Number of smiley faces (from 2 to 5)
   2. Slider question with the following properties:
      1. Question text
      2. Question order
      3. Start value
      4. End value (up 100)
      5. Start value caption
      6. End value caption
   3. Stars question
      1. Question text
      2. Question order
      3. Number of stars (up to 10)

5) All modifications should be reflected directly on the DB. For example, when the user create a question and click save button, the question should be directly inserted to the DB.

6) User experience and design should be constructed to match standard windows applications, and office applications.

7) The application should present helpful messages to the user, and should inform the user for any error happened.

8) User input validations should be implemented when required.

9) The application should not crash due to any user action.

10) Accessing the DB should be done using ADO.NET command. You can NOT use LINQ or Entity framework.
11) The code should have proper amount of comments.
12) All relations should reflected on the DB with validation.
13) The application should support multi-instance running from same PC or different PC. The user should only add the DB connection parameters in a configuration file beside the application.
14) All classes, variables, DB tables should be named informatively and with a convention.
15) All errors should be written in a log file with a proper format.
16) Apply learned knowledge from latest reading in this task, so the code should reflect what you learned.
17) Code Reusability and abstractions are highly important.
18) Create a Github repository and upload the task on it. You should make at least one commits every three hours of work to reflect gradual changes. Once you create the repository, kindly send its link to me (it should be a public one)
19) Create a progress.txt file and put it on the same github repository. The file should include an entry per day, containing the following:
    1. Date
    2. What is accomplished today.
    3. What is remaining to be done (as list of items).

    This text file should be updated once at the end of the day.
20) At the end of the task, create a documentation of your work in word format. The documentation should have three sections and each one should target one of these types of audiences:
    1. End user who will use the task (it is like user manual)
    2. Administrator: the one who will install the task on the user PC.
    3. Developer: Someone who will work on your task, so you should document for him all the technical decisions you have made and to explain the design/architecture of your application & DB.
21) Test your application thoroughly and deeply for all strange scenarios until there are no bugs left to be found.

As you can see, the functionality of this task is easy. However, what is important is **the none-functional requirements** to evaluate this task. So, keep attention to them and the time is less important. The minimum time for the task is three days, and there is no maximum. **Don't hurry, keep enhancing until you reach the maximum level of perfection at all none-functional requirements/quality attributes**.