**Question Creator Form Documentation**

This document is a documentation of the Question Creator windows form application, it will be sectioned off into three sections, **End user documentation**, **Developer documentation**.

**End user documentation**:

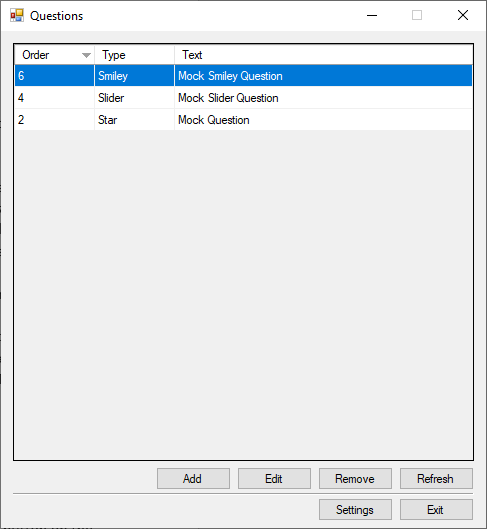
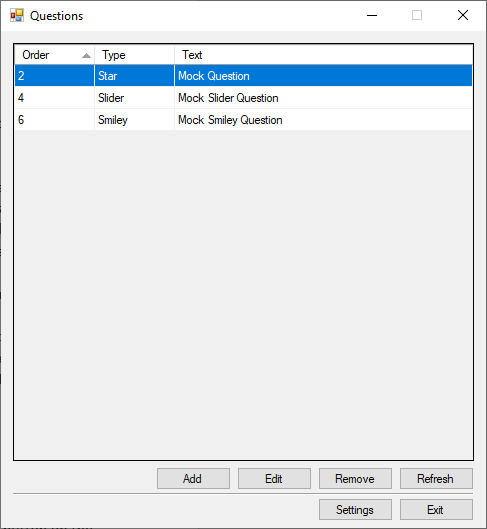
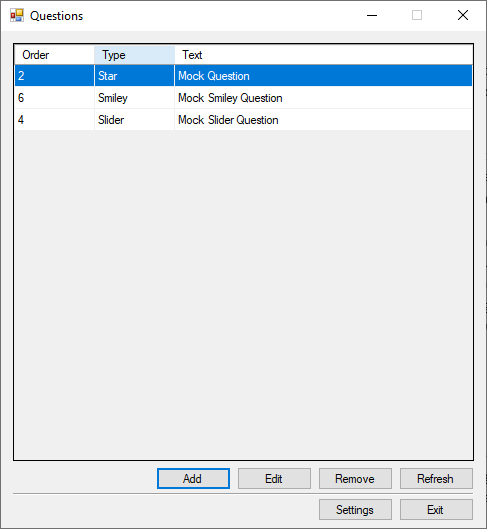
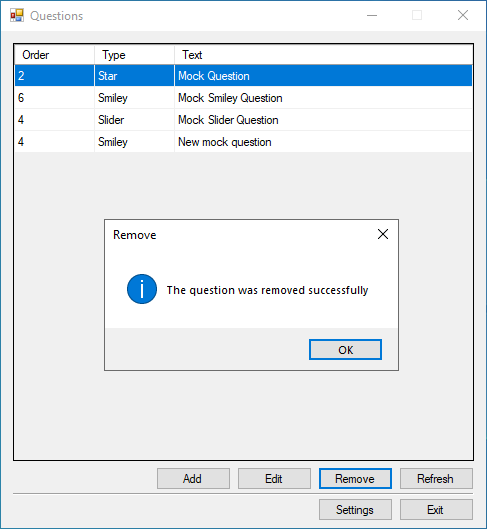
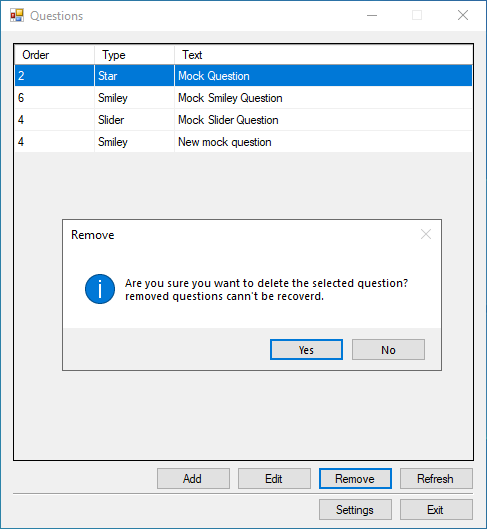
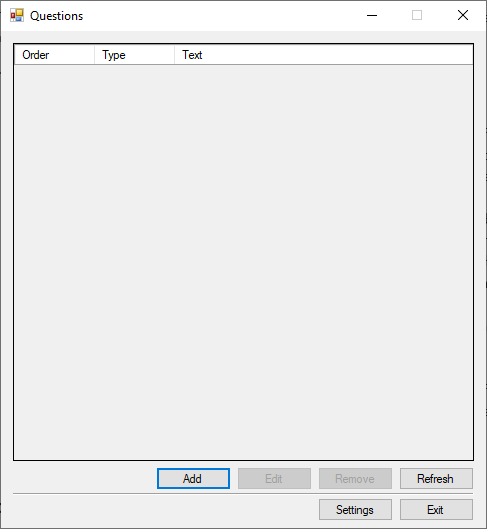
This section will walk you through the different forms, and what each form aims to accomplish.

**Landing form**:

The landing form is where the user can see all the questions and interact with them, it shows the user a list of questions that are present in the database as well as them being updated with the database every ten seconds.

Landing form controls:

* List: lists all the questions from the database, showing off their text, type and their order, and the list column headers can be clicked and that will allow the user to sort the questions based on the column header that was clicked.
* Refresh button: refreshes the data directly from the database, if successful a message will show at the bottom of the list indicating that a refresh happened and the data is up to date with the database.
* Add button: Launches the questions form which allows the user to add a new question.
* Edit button: Launches the questions form with the currently selected question in the list and allows the user to edit that specific question. (This button is only activated when a question is selected in the list)
* Remove button: launches a message to prompt the user for consent to remove the currently selected question in the list. (This button is only activated when a question is selected in the list
* Settings button: Launches the settings form
* Exit button: Closes the application

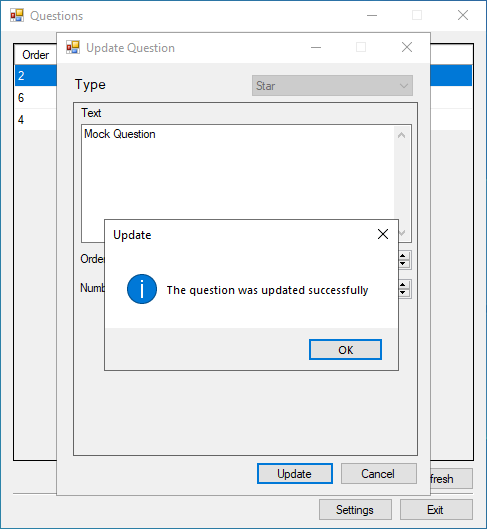
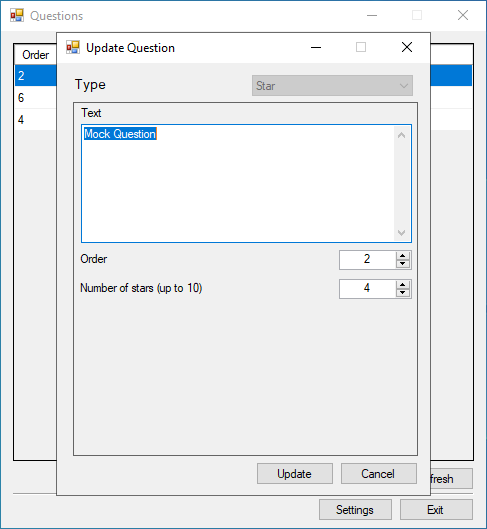
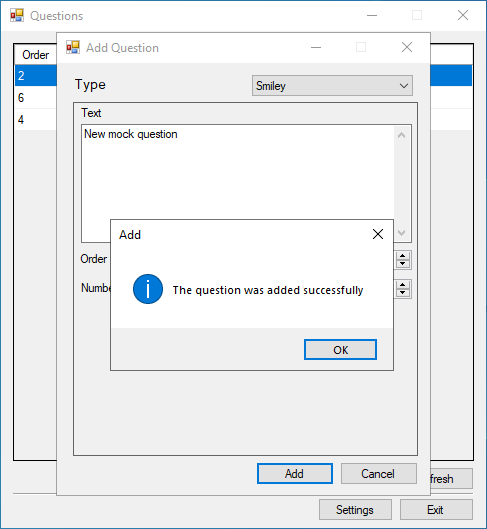
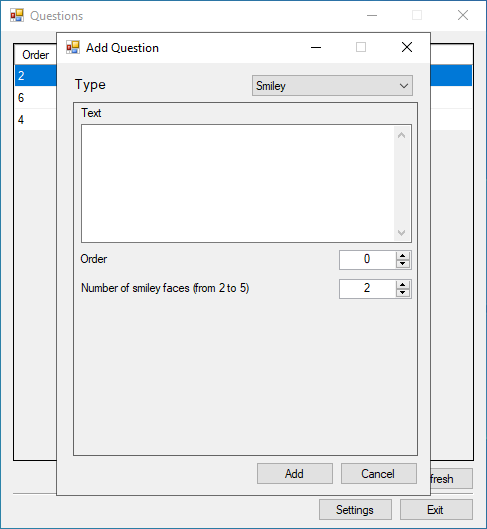


**Question form**:

The question form lets the user either add a new question or edit a new question, editing a question is accessed through pressing the Edit button on the landing form, while adding a new question is accessed through pressing the Add button in the landing form. (Adding or updating questions will reflect automatically to the database and to the landing form list)

Question form controls:

* Question type dropdown: this is a dropdown list containing the possible types of the questions that can be chosen from, choosing different types will change the extra data that the new question has. (disabled when editing a question)
* Question text input field: an input field that allows the user to edit/add the question text, has a max character count of 250.
* Question order numeric box: a numeric box that lets the user change the value of the new/old question order value.
* Update/Add button: this lets the user finish the action of creating/editing a question, then launches a message box that contains the outcome of the Adding/Updating operation, whether It’s a success or not.
* Cancel button: lets the user exit the question form and return to the landing form.

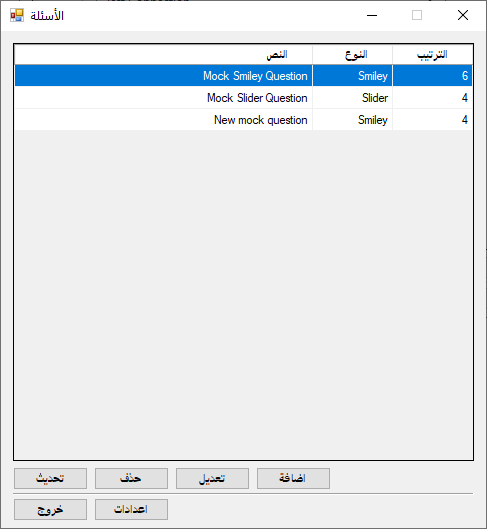
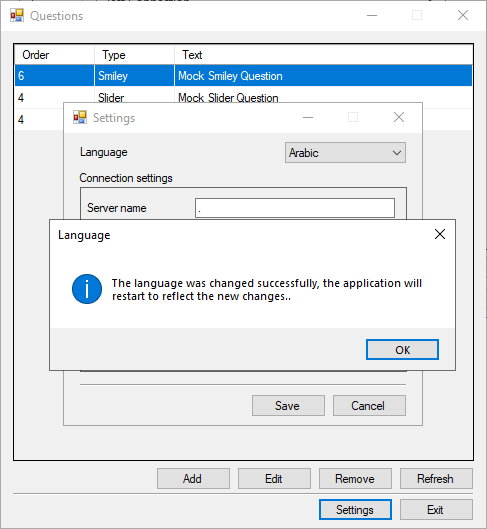
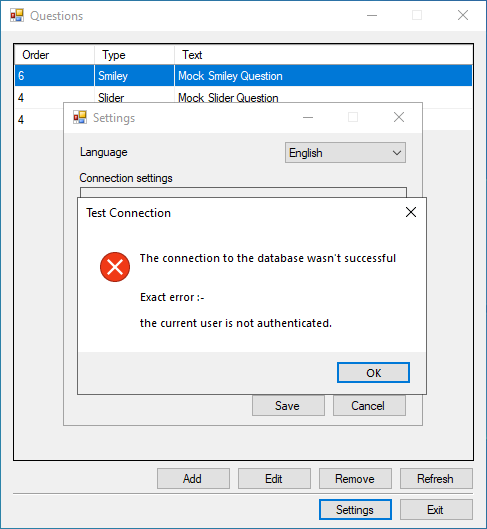
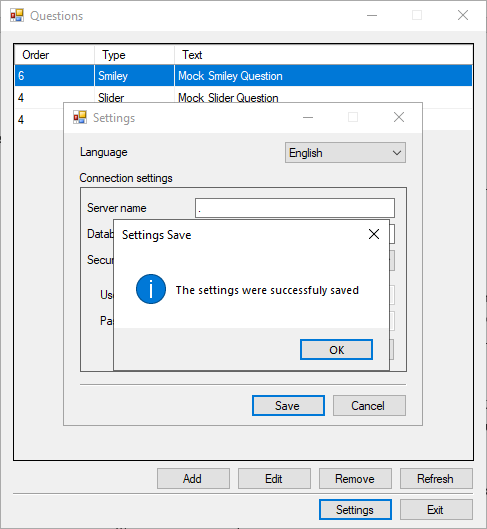
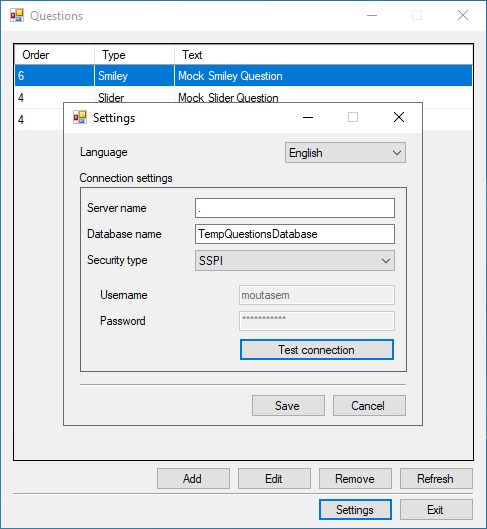


**Settings form**:

The settings form is where the user can change different settings options, such as the language and the connection type and info, as well as being able to test out new connections.

**Settings form** controls:

* Language dropdown menu: The language dropdown menu lets the user choose the language they want to use, the two options are Arabic and English, whenever a user chooses a language to use a message box letting the user know the result of selecting that language and then automatically restarting the application with the newly selected language.
* Connection settings input fields: These are a group of input fields that let the user control the connection settings to the database, on top of that a test connection button that allows the user to test out connection settings before actually saving them and using them.



**Developer documentation**:

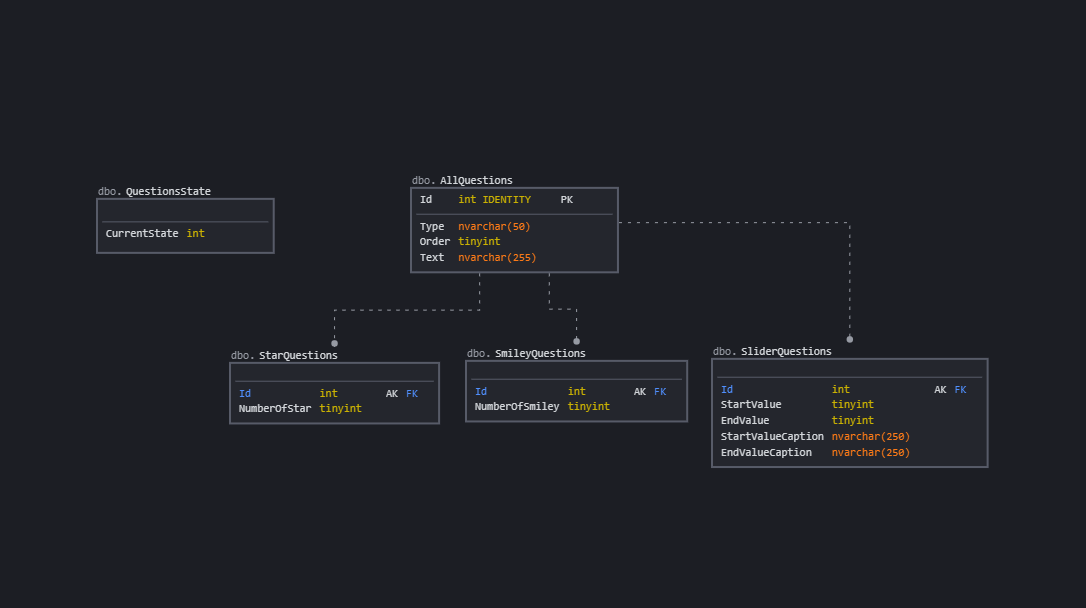
The application consists of three forms, a main landing form for the users to interact with and launch other forms using it.

The main architecture of the application is a three layered application, any other layers are support layers that help the application out:

* The UI Layer: this is built with windows forms using C#, consisting of one main form and two sub forms
* The Business Layer: this is built using C#, which consists of a QuestionsController, this is basically the in memory data, this gets it’s information from the Data Layer whenever we use it, this also automatically updates the in memory data every 10 seconds and notifies the Event handler it has to fire an event whenever it does actually update the data.
* The Data Layer: This is built using C# and SQL
* The Model Layer: This is built using C#, this consists of the different questions classes and their specification, this also has Utility classes that help the application out in retrieving ResultCodes, this also has a QuestionsFactory class that creates different instances of the Questions we have
* The Logging Layer: This is a utility layer that handles the logging of the exceptions that are thrown anywhere in the application to a log file.

The three layered architecture was used because it separates the different components of the application and creates a high level of abstraction, so whenever something needs to change in any of the layers, only that layer will be changed and no other layers will be affected.

Database Diagram:



The database consists of Five tables:

* QuestionsState: This table only consists of an Int and is used to determine the current state of the data in the other tables, so It’s basically a table that the developer can access and see if the current application data is up to data with the data in the database or not.
* AllQuestions: This is the main table for the data, it has an Id primary key, Type, Order, Text, so basically the main Question attributes.
* SmileyQuestions: This is a sub type of Questions, it only has an Id and the extra data this sub type has, the Id is a FK for the allQuestions table, since the main data of any SmileyQuestions existing currently in this table are in the AllQuestions table.
* SliderQuestions: This is also another sub type of Questions, and the same behavior that the SmileyQuestions has also applies here.
* StarQuestions: This is also another sub type of Questions, and the same behavior that the SmileyQuestions has also applies here.

The database also consists of different Procedures that do different kind of operations,

Examples:

CREATE OR ALTER PROCEDURE [dbo].Update\_CurrentState

AS

BEGIN

BEGIN TRY

BEGIN TRAN

UPDATE QuestionsState SET [CurrentState] = [CurrentState] + 1;

DECLARE @QuestionsStateCount INT;

SET @QuestionsStateCount = (SELECT MAX([CurrentState]) FROM QuestionsState);

IF (@QuestionsStateCount = 2147483647)

UPDATE QuestionsState SET [CurrentState] = 0;

COMMIT TRAN

END TRY

BEGIN CATCH

ROLLBACK TRAN

THROW;

END CATCH

END

GO

This procedure updates the current state of the database whenever we call any other procedures, and also handles overflowing of the int that currently lives in the QuestionsState table.

CREATE OR ALTER PROCEDURE [dbo].Add\_StarQuestions

(@Text nvarchar(250), @Order INT, @NumberOfStar INT, @Id INT = NULL OUTPUT)

AS

BEGIN

BEGIN TRY

BEGIN TRAN

SET XACT\_ABORT ON;

INSERT INTO AllQuestions VALUES('Star', @Order, @Text);

SET @Id = SCOPE\_IDENTITY();

INSERT INTO StarQuestions VALUES(@Id, @NumberOfStar);

EXEC [dbo].Update\_CurrentState;

COMMIT TRAN

END TRY

BEGIN CATCH

ROLLBACK TRAN

THROW;

END CATCH

END

GO

This procedure takes care of creating a new question of type star, which basically inserts into the AllQuestions table, then inserts into the StarQuestions table.

CREATE OR ALTER PROCEDURE [dbo].Update\_StarQuestions

( @Text nvarchar(255), @Order INT, @NumberOfStar INT, @Id INT )

AS

BEGIN

BEGIN TRY

BEGIN TRAN

SET XACT\_ABORT ON;

DECLARE @totalRows INT;

SET @totalRows = 0;

UPDATE AllQuestions SET Text = @Text, [Order] = @Order WHERE Id = @Id AND Type = 'Star';

SET @totalRows = @totalRows + @@ROWCOUNT

UPDATE StarQuestions SET NumberOfStar = @NumberOfStar WHERE Id = @Id;

SET @totalRows = @totalRows + @@ROWCOUNT

if (@totalRows = 2)

EXEC [dbo].Update\_CurrentState;

COMMIT TRAN

END TRY

BEGIN CATCH

ROLLBACK TRAN;

THROW;

END CATCH

END

GO

This procedure takes care of updating StarQuestions, which all it does is Update the AllQuestions table with new data, then updates the StarQuestions table with the new data, then just checks if 2 rows were affected, if so update the CurrentState column in the QuestionsState indicating that a Question was updated.

Other procedures exist as well for each Questions type etc and for each operation, with all of them being a transaction to make sure that everything correctly happens and with try catches to catch out any errors that might arise, and they can be all found in the SQL file that exists with this project.