|  |  |
| --- | --- |
|  | **2021** |
|  |  |

|  |
| --- |
| **[Code Complexities]** |
|  |

Table of Contents

[Admin Details 2](#_Toc70075229)

[Admins Cannot Delete Themselves 2](#_Toc70075230)

[Admins Can Revoke Their Own Admin Privileges 2](#_Toc70075231)

[Original Admin Cannot Be Deleted 3](#_Toc70075232)

[Combo Boxes Automatically Updated To Include New Text 4](#_Toc70075233)

[Admin Returns to Admin Menu If They Accessed Form from Admin Menu 5](#_Toc70075234)

[Randomisation 6](#_Toc70075235)

# Admin Details

As a lot of the complexity of the system lies within the admin areas it only makes sense to supply the admin login details:

Username: ethanAdmin

Password: qwert1

# Admins Cannot Delete Themselves

This code shown below stops a user from deleting themselves. It also shows the usage of customized exceptions. It is simply achieved by disabling the delete button when a user chooses to edit their own data.

if (user.Username == thisUser.Username)

{

try

{

btnDelete.Enabled = false;

sameUser = true;

throw new CustomException("You will not be able to delete a user \n that is currently in use");

}

catch (CustomException ex)

{

lblIssue.Text = (ex.Message);

lblIssue.Visible = true;

}

}

# Admins Can Revoke Their Own Admin Privileges

This code shows what will happen if a user tries to save their own user data and disables their admin privileges. This could obviously cause errors/bugs in the code if a non-admin user was able to edit data and as such I made it so that the user will be kicked out of the admin facilities however will be given a warning message before so.

if (thisUser.Admin != true)

{

this.Close();

new frmMain(thisUser, easyDummyArray, hardDummyArray).Show();

}

else

{

this.Close();

new frmUserEdit(thisUser, easyDummyArray, hardDummyArray).Show();

}

if(sameUser == true && cbAdmin.Text == "False")

{

string message = "Saving now will revoke your admin privileges \n you will be unable to access admin facilities with this account unless changed by another admin \n Are you sure you want to continue?";

string title = "Save data";

MessageBoxButtons buttons = MessageBoxButtons.YesNo;

DialogResult result = MessageBox.Show(message, title, buttons);

# Original Admin Cannot Be Deleted

This code shows how if the original admin is selected then the delete button, changing of admin privileges and changing of username are all automatically disabled.

if (user.Username == thisUser.Username)

{

try

{

btnDelete.Enabled = false;

sameUser = true;

throw new CustomException("You will not be able to delete a user \n that is currently in use");

}

catch (CustomException ex)

{

lblIssue.Text = (ex.Message);

lblIssue.Visible = true;

}

}

if (user.Username == "ethanAdmin")

{

btnDelete.Enabled = false;

cbAdmin.Enabled = false;

txtUsername.Enabled = false;

}

This code shows that the original admin won’t even be an option to select unless you are the original admin. Meaning only the original admin can view the original admin’s data

foreach (User user in users)

{

if (thisUser.Username != lblMasterAdmin.Text)

{

if(user.Username != lblMasterAdmin.Text)

{

cbUser.Items.Add(user.Username);

}

}

else

{

cbUser.Items.Add(user.Username);

}

}

# Combo Boxes Automatically Updated To Include New Text

This code shows that any time a user types in the respective field it will automatically populate the combo box with whatever has been typed. This happens on every single key press to assure it is accurate. It is a small detail however makes it appear more professional. This occurs on the “Correct Answer” combo box for the add and edit question.

private void txtAnswer1\_KeyUp(object sender, KeyEventArgs e)

{

cbCorrectAnswer.Items.RemoveAt(0);

cbCorrectAnswer.Items.Insert(0, txtAnswer1.Text);

}

private void txtAnswer2\_KeyUp(object sender, KeyEventArgs e)

{

cbCorrectAnswer.Items.RemoveAt(1);

cbCorrectAnswer.Items.Insert(1, txtAnswer2.Text);

}

private void txtAnswer3\_KeyUp(object sender, KeyEventArgs e)

{

cbCorrectAnswer.Items.RemoveAt(2);

cbCorrectAnswer.Items.Insert(2, txtAnswer3.Text);

}

private void txtAddAnswer1\_KeyUp(object sender, KeyEventArgs e)

{

cbAddCorrectAnswer.Items.RemoveAt(0);

cbAddCorrectAnswer.Items.Insert(0, txtAddAnswer1.Text);

}

private void txtAddAnswer2\_KeyUp(object sender, KeyEventArgs e)

{

cbAddCorrectAnswer.Items.RemoveAt(1);

cbAddCorrectAnswer.Items.Insert(1, txtAddAnswer2.Text);

}

private void txtAddAnswer3\_KeyUp(object sender, KeyEventArgs e)

{

cbAddCorrectAnswer.Items.RemoveAt(2);

cbAddCorrectAnswer.Items.Insert(2, txtAddAnswer3.Text);

}

# Admin Returns to Admin Menu If They Accessed Form from Admin Menu

This shows an example of the code in a button that will take an admin to a form from the admin menu. It will pass an admin bool variable. This happens for each form accessible from the admin facility.

private void btnDragDropTest\_Click(object sender, EventArgs e)

{

admin = true;

this.Close();

new frmQuestionDragDrop(thisUser, easyDummyArray, hardDummyArray, difficulty, admin).Show();

}

This shows how the button changes depending on if the admin variable is true.

if (Admin == true)

{

btnContinue.Text = ("Back to admin");

}

private void btnContinue\_Click(object sender, EventArgs e)

{

if (admin == true)

{

this.Close();

new frmAdmin(thisUser, easyDummyArray, hardDummyArray).Show();

}

else

{

this.Close();

new frmHardQuestionTextBox(thisUser, easyDummyArray, hardDummyArray, difficulty, admin).Show();

}

}

# Randomisation

This code will separate easy questions and hard questions in the bin file for easier randomization.

do

{

listLength = questions.Count;

if (j == firstLength)

{

break;

}

Question = questions[i];

if (Question.Hard == false)

{

questions.Remove(Question);

}

else

{

i += 1;

}

j += 1;

} while (i < listLength);

This is the code that will populate an original dummy array for randomisation. Each time a randomised question appears an element in the dummy array will change from a 0 to a 1 signifying the question has been used.

private void frmHardQuestionRadioButton\_Load\_1(object sender, EventArgs e)

{

if (hardDummyArray == null)

{

int listLength = questions.Count;

hardDummyArray = new int[listLength];

for (int i = 0; i < listLength; i++)

hardDummyArray[i] = 0;

}

Random random = new Random();

try

{

int randomElement = random.Next(listLength);

do

{

randomElement = random.Next(listLength);

Question = questions[randomElement];

if (Question.Hard == true)

{

if (hardDummyArray[randomElement] == 0)

{

lblShowQuestion.Text = Question.Question;

rdbAnswer1.Text = Question.Answer1;

rdbAnswer2.Text = Question.Answer2;

rdbAnswer3.Text = Question.Answer3;

hardDummyArray[randomElement] = 1;

newQuestion = false;

foreach (int number in hardDummyArray)

{

if (number == 1)

{

noQuestionsUsed++;

}

}

}

else

{

hardDummyArray[randomElement] = 1;

randomElement += 1;

newQuestion = true;

}

}

} while (newQuestion == true);

}

catch

{

}

}

This code appears on the first form of each difficulty (one for easy and hard dummy arrays). This code checks if the dummy array is full and will reset it if it is fully as well as display a message informing the user. It will also appear in the randomised question forms as if an admin were testing the randomisation they may access the randomised question form straight from the admin menu which would bypass the check to see if the dummy array has been filled causing the system to crash.

if (hardDummyArray != null)

{

hardListLength = hardDummyArray.Length;

foreach (int num in hardDummyArray)

{

if (num == 1)

{

hard++;

}

}

if (hard == hardListLength)

{

MessageBox.Show("Randomised questions have been exhausted. They will now repeat");

foreach (int num in hardDummyArray)

{

for (int a = 0; a < hardListLength; a++)

{

hardDummyArray[a] = 0;

}

}

}

}