Student Attendance Tracker

Technical Reference Manual





By team TRC: Harold Smith, Matt Rawson, Zach Minnix, Yingjun Liao, Austin Bower

**Table of Contents**

[Introduction 4](#_Toc450073556)

[Purpose 4](#_Toc450073557)

[Description 4](#_Toc450073558)

[File Information 5](#_Toc450073559)

[Welcome Screen 5](#_Toc450073560)

[Description 5](#_Toc450073561)

[Files 5](#_Toc450073562)

[Functions 5](#_Toc450073563)

[UI Elements 6](#_Toc450073564)

[Possible Errors 6](#_Toc450073565)

[Login Screen 6](#_Toc450073566)

[Description 6](#_Toc450073567)

[Files 6](#_Toc450073568)

[Functions 7](#_Toc450073569)

[UI Elements 7](#_Toc450073570)

[Possible Errors 7](#_Toc450073571)

[Register Screen 8](#_Toc450073572)

[Description 8](#_Toc450073573)

[Files 8](#_Toc450073574)

[Functions 8](#_Toc450073575)

[UI Elements 8](#_Toc450073576)

[UI Elements (cont.) 9](#_Toc450073577)

[Possible Errors 9](#_Toc450073578)

[Home Screen 9](#_Toc450073579)

[Description 9](#_Toc450073580)

[Files 10](#_Toc450073581)

[Functions 10](#_Toc450073582)

[UI Elements 10](#_Toc450073583)

[Possible Errors 11](#_Toc450073584)

[Add Courses Screen 11](#_Toc450073585)

[Description 11](#_Toc450073586)

[Files 11](#_Toc450073587)

[Functions 11](#_Toc450073588)

[UI Elements 11](#_Toc450073589)

[Possible Errors 12](#_Toc450073590)

[Add Student Screen 12](#_Toc450073591)

[Description 12](#_Toc450073592)

[Files 12](#_Toc450073593)

[Functions 12](#_Toc450073594)

[UI Elements 13](#_Toc450073595)

[Possible Errors 13](#_Toc450073596)

[Student Swipe Screen 13](#_Toc450073597)

[Description 13](#_Toc450073598)

[Files 13](#_Toc450073599)

[Functions 13](#_Toc450073600)

[UI Elements 14](#_Toc450073601)

[Possible Errors 15](#_Toc450073602)

[Edit Attendance by Date Screen 15](#_Toc450073603)

[Description 15](#_Toc450073604)

[Files 15](#_Toc450073605)

[Functions 16](#_Toc450073606)

[UI Elements 16](#_Toc450073607)

[Possible Errors 16](#_Toc450073608)

[Index 17](#_Toc450073609)

# Introduction

## Purpose

At the time of this software’s inception our school (Concord University) handled taking roll either by passing around a roll sheet or by doing roll call. However, this method of taking attendance is time consuming and distracting. We have created this software in an attempt to mitigate the potentially adverse impacts of the taking attendance.

## Description

The Student Attendance Tracker software is an attendance management system which was written in the C# programming language via the Microsoft Visual Studio 2015 development tool under the Microsoft .NET framework and environment. Additionally, we utilized the SQL Server 2014 for the database development.

# File Information

Note that .xaml files contain the UI aspects of each screen and the .xaml.cs files contain the actual functional code. For the sake of simplicity some UI elements (such as simple display labels or images) will not be included here unless they actually do something other than display information.

## Welcome Screen

### Description

The Welcome screen is displayed when the software is first started and when a user either logs out, registers a new account, or cancels their registration or login.

### Files

MainWindow.xaml

MainWindow.xaml.cs

### Functions

|  |  |
| --- | --- |
| **Name** | **Purpose** |
| private void welcomeRegisterButton\_Click(object sender, RoutedEventArgs e) | This function is called when the user clicks on the register button. The function creates a register screen and closes the welcome screen. |
| private void welcomeLoginButton\_Click(object sender, RoutedEventArgs e) | This function is called when the user clicks on the login button. The function creates a login screen and closes the welcome screen. |

### UI Elements

|  |  |  |
| --- | --- | --- |
| **Type** | **Name** | **Purpose** |
| Button | welcomeLoginButton | The purpose of this button is to give the user a simple means of navigating to the login screen. |
| Button | welcomeRegisterButton | The purpose of this button is to give the user a simple means of navigating to the register screen. |

### Possible Errors

As long as the software builds and runs this screen should produce no errors.

## Login Screen

### Description

The login screen is used to retrieve login credentials from the user in order to check to see if the user is registered already and to display their information on the home screen.

### Files

loginScreen.xaml

loginScreen.xaml.cs

### Functions

|  |  |
| --- | --- |
| **Name** | **Purpose** |
| private void loginSubmitButton\_Click(object sender, RoutedEventArgs e) | This function is called when the user clicks the login button. The function checks the database to see if the credentials entered are valid. It can either create a home screen and close this screen or display an error message. |
| private void loginGoBackButton\_Click(object sender, RoutedEventArgs e) | This function is called when the user clicks on the go back button. The function creates a new welcome screen and closes this screen. |

### UI Elements

|  |  |  |
| --- | --- | --- |
| **Type** | **Name** | **Purpose** |
| Button | loginSubmitButton | The purpose of this button is to submit the login credentials entered by the user in order to check against the database to see if they are valid. |
| Button | loginGoBackButton | The purpose of this button is to give the user a simple means of navigating back to the welcome screen. |
| Text Box | usernameTextBox | This textbox simply holds the username value for the user. |
| Password Box | loginPasswordBox | This password box holds the password value for the user. As a built in function of this object it displays the entered in text as dots. |

### Possible Errors

|  |  |
| --- | --- |
| **Error Message/Type** | **Cause** |
| "INVALID CRENDENTIALS! The username / password combination entered is invalid, please try again." | The user entered in login credentials that do not exist within the database. It could be an invalid password, username, or both. |
| Complete software crash | The database that values are checked against is not up and running. A simple connection test will need to be implemented to prevent this from happening. |

## Register Screen

### Description

This page allows a professor to register to the Student Attendance Tracker System.

### Files

registerScreen.xaml

registerScreen.xaml.cs

### Functions

|  |  |
| --- | --- |
| **Name** | **Purpose** |
| registerSubmitButton\_Click | This function is called when the user clicks the submit button. The function checks the database to see if the credentials entered are valid. It can display both error and success messages based on the validity of the credentials. |
| registerGoBackButton\_Click | This function is called when the user clicks on the go back button. The function creates a new welcome screen and closes this screen. |
| passwordCheck | This function is called whenever the user clicks the submit button. The functions checks to make sure that the password is at least 6 characters, contains an uppercase letter, a lowercase letter, a number, and a punctuation mark. It also makes sure that the password and confirm password field match. |

### UI Elements

|  |  |  |
| --- | --- | --- |
| **Type** | **Name** | **Purpose** |
| Text Box | usernameTextBox | Field that the user will input their desired username to be registered with the system. |
| Text Box | professornameTextBox | Field that the user will put their full name to be stored in the database. |
| Text Box | departmentTextBox | Field where the professor will indicate the department that they work in. |
| Text Box | emailTextBox | Will contain the user’s email for password retrieval and as a means for contact. |

### UI Elements (cont.)

|  |  |  |
| --- | --- | --- |
| **Type** | **Name** | **Purpose** |
| Text Box | passwordTextBox | Will contain the desired password for the user’s account. |
| Text Box | confirmpasswordTextBox | Used to ensure that the user entered in the password accurately to prevent typos. |

### Possible Errors

|  |  |
| --- | --- |
| **Error Message/Type** | **Cause** |
| "Query execution error" | Insert Data into Database is not successful. |
| "Your password must be at least 6 characters." | Password Length checking |
| "Your confirm password is not match to password." | Confirmation password is not match to original password |
| "Your password needs to have an uppercase letter." | Password checking |
| "Your password needs to have a lowercase letter." | Password checking |
| "Your password must have a number." | Password checking |
| "Your password must have a punctuation mark." | Password checking |
| "Valid password!" | Password checking |

## Home Screen

### Description

The Home screen guide user to use add courses, add students, and take attendances functions. There is a status bar to give user an instruction. In addition, user can check attendances via select CRN in comboBox, and it will in “dataGrid”. Besides, user can update attendance numbers via input number in “Total Attendance” textbox, and click “Update Total Attendances” button.

### Files

homeScreen.xaml

homeScreen.xaml.cs

### Functions

|  |  |
| --- | --- |
| **Name** | **Purpose** |
| addCourseButton\_Click | Jump to addCourse screen after click addCourse button |
| addStudentButton\_Click | Jump to addStudent screen after click addCourse button |
| logOutButton\_Click | Close the home window and go back to the welcome window |
| LoadTableButton\_Click | Jump to editAttendance screen after click addCourse button |
| TakeAttendanceButton\_Click | Jump to takeAttendance screen after click addCourse button |
| Course\_comboBox\_Initialized | This function creates a dropdown list that will allow the professor to choose which courses to check attendance. |
| Course\_comboBox\_DropDownClosed | After close the dropdown list in comboBox, it will fill data from database into dataGrid. |
| UpdateTotalAttendancesButton\_Click | Update attendances into database after input number in Total Attendance textbox and click button |

### UI Elements

|  |  |  |
| --- | --- | --- |
| **Type** | **Name** | **Purpose** |
| Button | addCoursesButton | Jump to add course screen |
| Button | addStudentButton | Jump to add student screen |
| Button | TakeAttendanceButton | Jump to take attendance screen |
| Combo Box | Course\_comboBox | Initialize CRN from course list. After select CRN number, it will fill data from database to dataGrid |
| Data Grid | dataGrid | Read data from database to dataGrid. |
| Text Box | txtStudentID  {Binding SelectedItem.StudentNumber, ElementName=dataGrid}" | This field will had binding method in xaml to read data from dataGrid to textbox after click each row in dataGrid |
| Text Box | txtTotalAttendance  {Binding SelectedItem.Attendances, ElementName=dataGrid} | This field will had binding method in xaml to read data from dataGrid to textbox after click each row in dataGrid |
| Button | UpdateTotalAttendancesButton | Update attendances after input new number in txtTotalAttendance textbox and click this button |
| Button | LoadTableButton | Jump to edit attendance by date screen. |
| Button | logOutButton | Close the home window and go back to the welcome window |

### Possible Errors

|  |  |
| --- | --- |
| **Error Message/Type** | **Cause** |
| Catch Exception | Cannot connect to database |

## Add Courses Screen

### Description

The Add Course screen inserts “CRN” and “Course Name” into database when the “Submit” button is pressed.

### Files

addCourse.xaml

addCourse.xaml.cs

### Functions

|  |  |
| --- | --- |
| **Name** | **Purpose** |
| public addCourse() | Initializes all events, sets the focus of the cursor to the text block, readies the timer for when it is called. |
| private addStudentSubmit\_Click\_1 () | Insert Course information into database via click submit button |

### UI Elements

|  |  |  |
| --- | --- | --- |
| **Type** | **Name** | **Purpose** |
| Button | addCourseSubmit | This button will insert course information into database after click it. |
| Text Box | addCourseCRN | This field will hold the CRN for the course that the professor is adding. |
| Text Box | courseName | This field will hold the name for the course that the professor is adding. |

### Possible Errors

|  |  |
| --- | --- |
| **Error Message/Type** | **Cause** |
| "Query execution error" | Insert Data into Database is not successful. |
| CRN already exists! | Check whether CRN is already exist in database |

## Add Student Screen

### Description

The add student screen allows a professor to add students to one or more of the professor’s classes.

### Files

addStudent.xaml

addStudent.xaml.cs

### Functions

|  |  |
| --- | --- |
| **Name** | **Purpose** |
| addStudentSubmit\_Click\_1 | This function is called when the user clicks the submit button. The function checks the database to see if the student ID entered is valid or not and displays a success/failure message accordingly. |
| CRNcomboBox\_initialized | This function creates a dropdown list that will allow the professor to choose which courses to register the new student in. |

### UI Elements

|  |  |  |
| --- | --- | --- |
| **Type** | **Name** | **Purpose** |
| Text Box | studentID | This field will hold the student ID for the student that the professor is adding. |
| Text Box | studentName | This field will hold the full name for the student that the professor is adding. |
| Combo Box | CRNcomboBox | This field will let user select CRN to add student into this course. |

### Possible Errors

|  |  |
| --- | --- |
| **Error Message/Type** | **Cause** |
| "Query execution error" | Insert Data into Database is not successful. |
| "Student ID already exists!" | Check whether Student ID is already exist in this course. Duplicate checking. |

## Student Swipe Screen

### Description

The student swipe screen is shown when the professor chooses to take attendance and selects a class to take attendance for. While this screen is up students can swipe their student ID card to record their attendance.

### Files

StudentSwipe.xaml

StudentSwipe.xaml.cs

### Functions

|  |  |
| --- | --- |
| **Name** | **Purpose** |
| static StudentSwipe() | Creates a new reference to the dispatcher timer to be used in the timeDelay method. |
| public StudentSwipe() | Initializes all events, sets the focus of the cursor to the text block, and readies the timer for when it is called. |
| public string GetDate(DateTime Date) | This method returns the day, month, and year the user swiped in using built in classes and then converting the date to the form used in the database using the built in replace class. |
| public void databaseLoginCheck() | This method accesses the database that holds all the students the professor is currently teaching. The method first sets the boolean variable studentFound to false then it creates a connection to the database. Then the method looks for the student with their student ID and the course number of the class. If the student is found for the specific class it will set the studentFound to true before closing the connection. |
| public void insertTime() | This method accesses the database where the time and date of the student is stored and inserts the information into the database. First the method creates a connection to the database and looks for the specific student that swiped in. Next it updates the chosen student's information before closing the connection. |
| private void timeDelay(object sender, EventArgs e) | This method stops all changes to the user interface for 3 seconds using classes built into Visual Studios. First it checks the ‘if’ statement to see if justSwiped is true. If the variable is true then the variable is set to false the text box for instructing students is changed back, the go back button is reset to visible and the timer is stopped. After which the text box for students to swipe in is enabled again and the focus is set back to it. |
| private void studentAttendanceTracker(string userId) | This method accesses the database that keeps tracks of the number of swipes the student has succeeded at. First it creates a connection to the database, updates the number of swipes by one, then closes the connection. |

### UI Elements

|  |  |  |
| --- | --- | --- |
| **Type** | **Name** | **Purpose** |
| Button | go\_back\_button | Returns the user back to the home screen from StudentSwipe UI. |
| Text Box | student\_ID\_input\_box | This is the main event of the user interface. Here we collect the ID information from the ID card. The information is checked against an if statement that sees if the information from the card is at a string length of 18 characters and checks to see if you had just swiped before. If the information is 18 characters and the user had not just swiped it will enter the if statement where it will retrieve the 774 number of the student from the information swiped in. Next it retrieves the course number of the class chosen in the combo box. Next the date and time the student swiped in are added to the 774 number string, using already made classes in the visual studios library, to keep track of the student's attendance. |
| Combo Box | class\_select\_comboBox | This method fills the combo box from the database that stores the class names. First it creates a connection to the database. Next it fills the combo box inside a while loop with all of the classes selected from the database. Finally it closes the connection to the database and sets the focus of the cursor to the text box for students to swipe in. |

### Possible Errors

|  |  |
| --- | --- |
| **Error Message/Type** | **Cause** |
| “There is no selected class.” | The user has not selected a class in the combo box. |
| "Error date counter exceeded string length!" | The while loop in getDate reached the end of the DateTime variable without out reaching a space. |
| Complete software crash | The database that values are checked against is not up and running. A simple connection test will need to be implemented to prevent this from happening. |

## Edit Attendance by Date Screen

### Description

The Edit Attendance by Date screen allow user to choose CRN to check student’s attendances, attendance date, and attendance time via “dataGrid”. In addition, user can delete attendance via click “Delete” button.

### Files

editAttendances.xaml

editAttendances.xaml.cs

### Functions

|  |  |
| --- | --- |
| **Name** | **Purpose** |
| EditcomboBox\_Initialized | This function creates a dropdown list that will allow the professor to choose which courses to check attendance. |
| EditcomboBox\_DropDownClosed | After close the dropdown list in comboBox, it will fill data from database into dataGrid. |
| btnDelete\_Click | This function delete attendance from database after click delete button |
| btnView\_Click | After click View button to check whether delete attendance in database |
| btnGoBack\_Click | Jump back to home screen |

### UI Elements

|  |  |  |
| --- | --- | --- |
| **Type** | **Name** | **Purpose** |
| Combo Box | EditcomboBox | Initialize CRN from course list. After select CRN number, it will fill data from database to dataGrid |
| Data Grid | dataGrid | Read data from database to dataGrid. |
| Text Box | txtStudentID  {Binding SelectedItem.StudentNumber, ElementName=dataGrid}" | This field will had binding method in xaml to read data from dataGrid to textbox after click each row in dataGrid |
| Date Picker | dateAttendance  {Binding SelectedItem.AttendanceDate, ElementName=dataGrid} | This field will had binding method in xaml to read data from dataGrid to textbox after click each row in dataGrid |
| Text Box | txtTime  {Binding SelectedItem.AttendanceTime, ElementName=dataGrid} | This field will had binding method in xaml to read data from dataGrid to textbox after click each row in dataGrid |
| Button | btnDelete | Delete attendances click this button |
| Button | btnView | Check data from database via dataGrid |
| Button | btnGoBack | Close the home window and go back to the welcome window |

### Possible Errors

|  |  |
| --- | --- |
| **Error Message/Type** | **Cause** |
| Catch Exception | Cannot connect to database |

## Index

Button, 6, 7, 10, 11, 12, 14, 16

Combo Box, 10, 13, 15, 16

Data Grid, 10, 16

database, 4

Date Picker, 16

Description, 2, 3, 4, 5, 6, 8, 9, 11, 12, 13, 15

Files, 2, 3, 5, 6, 8, 10, 11, 12, 13, 15

framework and environment, 4

Functions, 2, 3, 5, 7, 8, 10, 11, 12, 13, 16

Possible Errors, 2, 3, 6, 7, 9, 11, 12, 13, 15, 16

programming language, 4

Text Box, 7, 8, 9, 10, 11, 12, 13, 14, 16

UI Elements, 2, 3, 6, 7, 8, 9, 10, 11, 13, 14, 16