Student Attendance Tracker

Technical Reference Manual





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

**Table of Contents**

[Introduction 3](#_Toc449476826)

[Purpose 3](#_Toc449476827)

[Description 3](#_Toc449476828)

[File Information 4](#_Toc449476829)

[Welcome Screen 4](#_Toc449476830)

[Description 4](#_Toc449476831)

[Files](#_Toc449476832) 5

[Functions](#_Toc449476833) 5

[UI Elements](#_Toc449476834) 6

[Possible Errors 5](#_Toc449476835)

[Index 6](#_Toc449476836)

# 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

This is the section of the manual involving the Student Swipe user interface and how each section of it works. The section is set up to discuss the code for each of the listed sections under the Navigating the Software Tab. This is done by breaking down the code thoroughly and thoroughly explaining it. The software was developed using Microsoft Visual Studios. It uses the programming language C# to build the code outside the design of the interface. As for the interface itself it was designed using Microsoft Visual Studios XAML language in the Windows Presentation Foundation (WPF).

# 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.

## Student Login Screen

### Description

The Student Login screen is displayed when the take attendance button is pressed.

### Files

StudentSwipe.xaml

StudentSwipe.xaml.cs

### Functions

|  |  |
| --- | --- |
| **Name** | **Purpose** |
| 1. static StudentSwipe() | Creates a new reference to the dispatcher timer to be used in the timeDelay method. |
| 2. public StudentSwipe() | Initializes all events, sets the focus of the cursor to the text block, readies the timer for when it is called. |

### Functions (cont.)

|  |  |
| --- | --- |
| 3. public string GetDate(DateTime Date) | This method returns the time the user swiped in using the 24 hour clock. First it accesses the date using built in classes then the date is taken through a while loop where the unwanted 12:00:00 AM is removed by going through the string, putting each character in a temporary string, and stops once it reaches a space character in the string. The try and catch are their for extra protection just in case there a chance that the while loop exceeds the length of the string some how. |
| 4. 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. |
| 5. 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. |
| 6. 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. |
| 7. 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. |

### Functions (cont.)

|  |  |
| --- | --- |
| 8. public void retrieveCourseNumber(string userInput) | This method gets the chosen class's course number stored in the database. First the method checks the if statement to see if the combo box is not empty. If it is not it creates a connection to the database, but if the combo box is empty a message box will open to inform the person swiping. Next it goes through the database looking for the course number that has the name selected in the combo box. Last it closes the connection to the database. |

### 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 (for more information refer to 3.) 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 (for more information refer to 5.).  Next the method databaseLoginCheck() is called to determine if the person who swiped in is in the selected class. Next the text box the student's swipe into is disabled to avoid any accidental swipe ins. Last after the databaseLoginCheck() method is finished the software reaches the if statement that uses a boolean check. If the boolean is true the software runs the insertTime method() (for more information refer to 5.), the studentAttendanceTracker(string userId) method (for more information refer to 7.), changes the text of the instruction block for students states that the swipe has been successful, hides the go back button on the user interface, activates the timeDelay(object sender, EventArgs e) (for more information refer to 6.), and clears the text box student swipe into so that their no chance of residual data. The else of the if statement changes the text of the instruction block for students states that the swipe has been unsuccessful, hides the go back button on the user interface, activates the timeDelay(object sender, EventArgs e) (for more information refer to 6.), and clears the text box student swipe into so that their no chance of residual data. |

### UI Elements(cont.)

|  |  |  |
| --- | --- | --- |
| 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. |

# Index

**No index entries found**