Login Program User Guide

By: Austin Scholp

Table of Contents

[Front Panel 2](#_Toc80091257)

[Main Menu 2](#_Toc80091258)

[Reports 2](#_Toc80091259)

[Full Report 3](#_Toc80091260)

[Admin 3](#_Toc80091261)

[Block Diagram 4](#_Toc80091262)

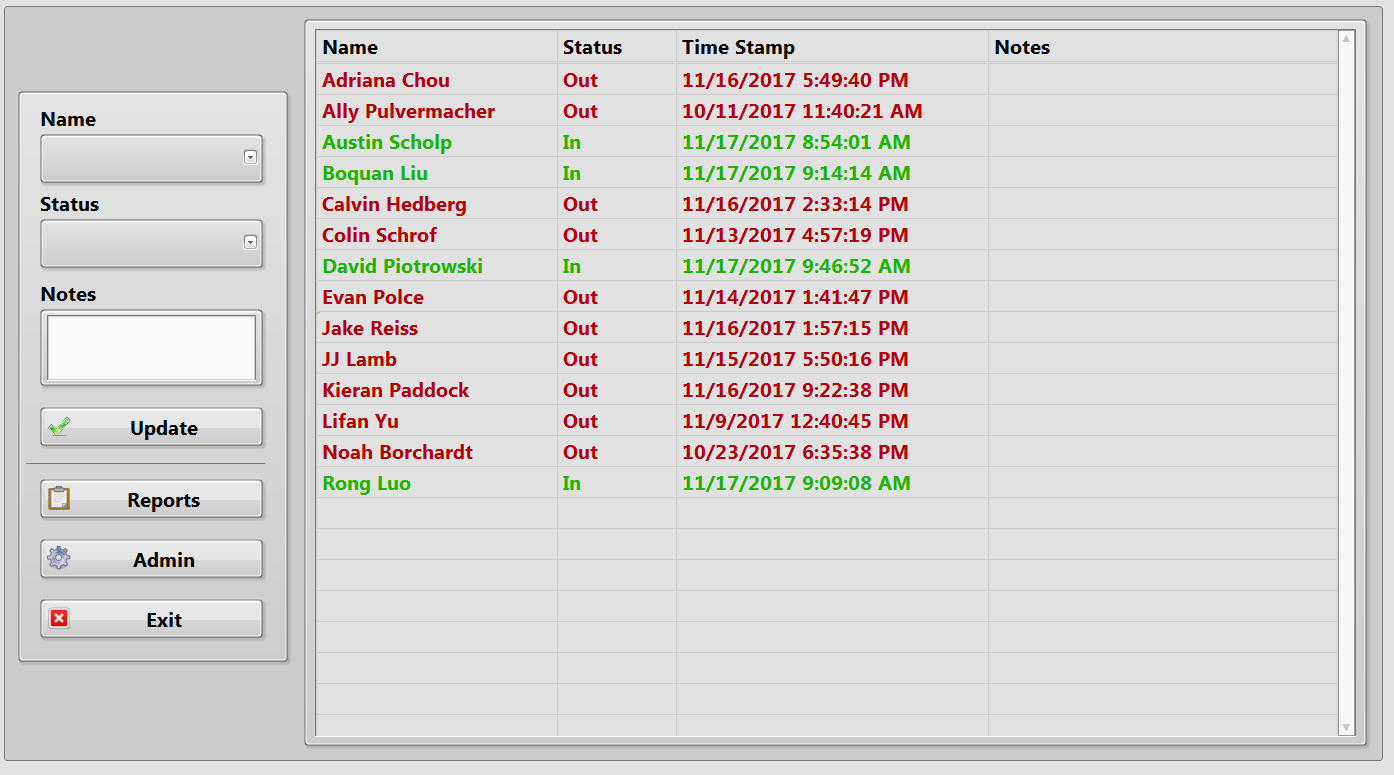
[Main Menu 4](#_Toc80091263)

[Event Handling Loop 5](#_Toc80091264)

[Message Handling Loop 5](#_Toc80091265)

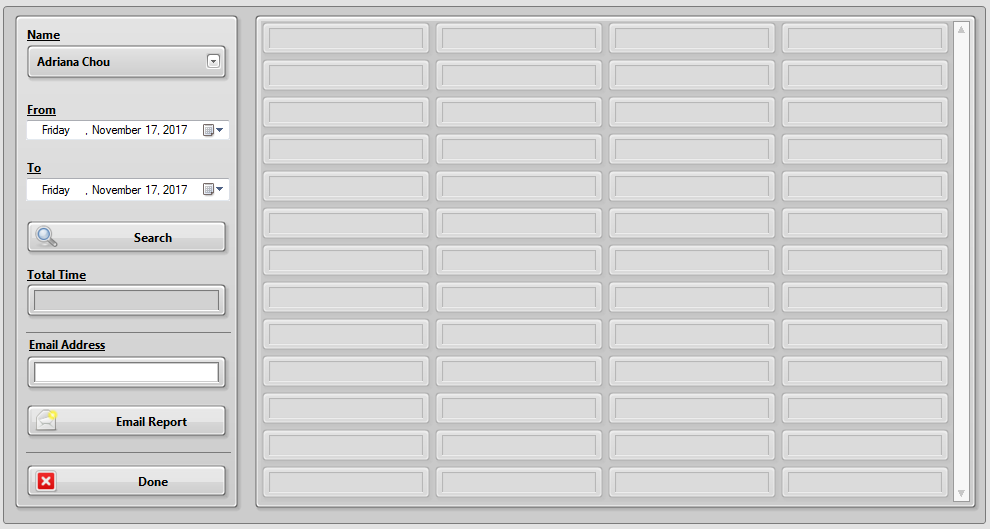
# Front Panel

## Main Menu

This program keeps track of when employees check in and out of the lab. It also can generate hour reports for each individual as well as a full report that includes hours for everyone. The user interface is pretty simple. All one has to do it select their name, select their status (In, Out, Break, or Lunch), enter any notes if desired, then click Update.

## Reports

To check on your (or someone else’s) hours, click reports, select your name and the date range you are interested in, then click search. This report can be emailed as well.



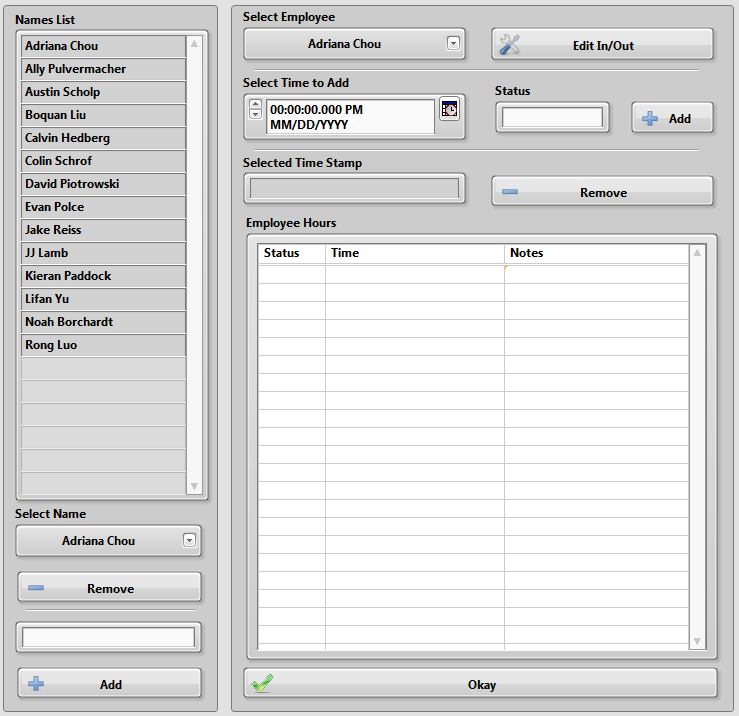
### Full Report

Graphical user interface, application

Description automatically generatedTo open the full report window (below), you need to hit F9 while you have the regular reports window open. The program will then iterate through all the employee names and compile a report.

The total hours for each employee are displayed on the left You can scroll through individual reports on the right.

## Admin

The admin window allows you to edit the hours that were input and add or remove employees.

To remove an employee, select their name from the drop-down menu on the left, then click remove. To add an employee, type in their name in the space below the remove button and click add.

To edit hours, first select an employee, then click edit in/out. Once the hours are displayed, you can select an input by clicking on it in the generated list and click remove to delete the selected time input. To add an input, select a date and time, pick a status, then click add.

# Block Diagram

## Main Menu

First, the Initialize Data SubVI is called. This SubVI creates a message queue and reads the data from the List and Display text files. LIST.txt contains a list of all the lab members that use the check-in program and Display.txt has the most recent data that is displayed on the main front panel. It contains names, status, update time, and notes. The list of names and display data are output to arrays so they can be used by the program later. More details on this SubVI can be found in its own subsection.

Diagram

Description automatically generatedDiagram, qr code

Description automatically generatedSecond, a data cluster is created so information can be passed between cases of the message handling loop. This is made up of an enumerated list of names, an enumerated list of statuses, a string for notes, and a 2D array for display data.

Third, a for loop checks if a text file exists for each name in the list.

Each employee sheet text file contains a record of all updates for that employee.

Graphical user interface, application

Description automatically generatedThe main part of this program is set up in a producer consumer structure with an event handling/message sending loop and a message handling loop. More details for each loop can be found in later sections of the guide.

### Event Handling Loop

Diagram

Description automatically generatedThis loop contains an event structure. Here, specific responses to user inputs are handled.

1. **Update** – When the **Update** button is pressed, the ‘Update’ message is added to the queue.
2. **Write Note** – When the **Write Note** button is pressed, the ‘Note’ message is added to the queue.
3. **Name Selector**  - When the value changes in the name selector drop-down menu (a user selects their name), this case checks if the name selected is actually not empty and enables the **Update** button if it is not empty. The ‘Nothing’ message is added to the queue because nothing needs to happen after this.
4. **Reports** – The ‘Report’ message is added to the queue.
5. **Admin Editing**  - The ‘Edit’ message is added to the queue when the **Admin** button is pressed.
6. **Exit**  - When the user presses the **Exit** button, the ‘Exit’ message is added to the queue and the event handling loop is stopped.

#### Disable Controls SubVI

Diagram

Description automatically generatedThis SubVI uses the ‘Disabled’ property node to enable or disable front panel objects. An array of object references needs to be wired in. All the objects will be enabled or disabled.

### Message Handling Loop

Here, messages are dequeued and different cases are called based on each message.

#### Initialize

Diagram

Description automatically generatedThis is the first case to be called when the program starts.

The nested for loops search through the statuses of each employee on the list and changes all the text in their row to the appropriate color.

Diagram, text

Description automatically generatedThe other for loop assigns numerical values to each name in the list.

The name selector drop-down menu is set to the last value, which should be blank. Finally, the update button is disabled via property node.

#### Update

Diagram, schematic

Description automatically generatedThe first thing to occur in this case is a check to see if the Name or Status drop-down menus are blank. If either one of them are, then nothing will happen. The program will wait for the next user input.

If a name and status is selected, it will move on to the inner case structure. Wired to this structure (from top to bottom) are the Data cluster, 2D Display array, 1D Names List array, the selected Name string, the Base File Path, and the Status string.

A picture containing diagram

Description automatically generatedInside the case structure, the Display and employee sheet text files are updated. Additionally, the color of the text for the selected employee changes based on the status. The SubVIs used here are “Build Display Array”, “Log Hours to Array”, “2D Array Sorter”, and “Save Display Files.”

##### Build Display Array

Timeline

Description automatically generated with medium confidenceThis SubVI finds the current time then formats the time into a string. It then builds an array of the Status, Timestamp, and Notes. This is then converted into a spreadsheet string (to be added to the employee sheet). It also builds an array of Name, Status, Timestamp, and Notes (for the display).

##### Log Hours to Array

Diagram

Description automatically generatedIf in the main update case, the name selected does not match any in the list (search array index < 0), a new employee sheet is created for them. If there is a match, the updated 1D display array, created in the Build Display Array SubVI, is inserted into the 2D Display Array at the index of the matching name.

##### 2D Array Sorter

Table

Description automatically generatedI think the best way to explain how this works is with an example.

* Suppose you have a 2D array of names and statuses (right).
* To sort them by name, we need to look at just the first column. Thus, the sorting index we should use is 0.
* Table

  Description automatically generatedTo sort the entire 2D array, we need to assign values to each name. Instead of a 2D array, these are assigned in a cluster.
* The sort 1D array function then sorts the clusters alphabetically, giving us the 1D of clusters as Austin 2, Gabe 4, Jack 3, Larissa 1.
* We then rebuild the 2D array based on the numbers assigned to each name.
* Diagram

  Description automatically generatedThe assigned numbers are used to take the entire row of the unsorted 2D array and add each row to a blank array in the order of the 1D array of clusters. In our example, the 2nd row would be added first, then the 4th, 3rd, and 1st.

##### Save Display Files

Diagram

Description automatically generatedHere, the Display.txt file is updated with the new 2D Display Array (after it is converted to a spreadsheet string). The Status, Time, Notes array from “Build Display Array” is added to the end of the employee sheet (text file) of the selected name.

#### Note

This case is almost exactly the same as the Update case. The main difference here is that the updated display and status information is not saved. This is so these inputs are not read in when running reports.

#### Report

Text

Description automatically generatedThe report case simply opens up the “File Hour Report” SubVI. The first part of this SubVI opens the Emails.txt file and separates out the names and addresses into two 1D arrays.

The rest of the program is set up in a state machine layout. The cases that it works through are “Wait”, “Search”, “Email”, “Exit”, and “Full Report.” Details for each case can be found in the following subsections.

##### Wait

Text

Description automatically generatedThis default case contains an event structure that handles button presses for each of the buttons on the front panel (Search, Email Report, Done) as well as a case for if F9 is pressed.

F9 calls the Full Report case. This is so it is not readily accessible to anyone who does not know of it. All the other buttons are self-explanatory.

There is also a timeout case that triggers if no user input is detected for 6 minutes (360,000 msec). A small window will pop-up asking if the user is still there. If no one presses the button on the pop-up, the reports window will be closed.

##### Search

Diagram

Description automatically generatedThe first part of this case, the employee sheet (text file) containing all of that employee’s check in/out timestamps is loaded and converted to a 2D array.

Diagram

Description automatically generatedNext, the for loop iterates through the times in the selected employee sheet text file and checks if the dates are in between the Start/End dates selected by the user. If they are, it adds the Time, Status, and Notes to the report.

After the report is made, it runs through the “Hour Differences” SubVI. This will be described in more detail on the next page.

###### Hour Differences

Diagram

Description automatically generatedHere we have a for loop that iterates through each status on the report finds all those that are “Out”, “Break”, or “Lunch” statuses. It will then find the time difference from the previous index in the report (this assumes that the previous status will be an “In” status) . Once it works through the report, it sums to total differences and adds each difference to a new column in the report array. This is the final report array that is then displayed.

##### Email

Diagram

Description automatically generatedThis case takes the hour report, converts it into a spreadsheet string, and uses the Send Email Express VI to send the report via email. The subject of the email is “Hours Report – Total Time -<HH:MM:SS>”

If the selected name is not on the email address list (search index is -1), it is added through the use of the case structure shown.

If there are any errors detected here, the user will be warned via the simple error handler.

##### Full Report

This case uses a for loop to iterate through all of the employee names. It will open each employee’s text file containing their log in/out data then, similar to the regular report, iterate through all the times searching for those between the selected time points. The Hour Differences SubVI is used to calculate the total hours for each employee, and everything is formatted into the Full Report Dialog SubVI.

###### Full Report SubVI

Diagram

Description automatically generatedHere, a while loop waits for the user to click **Okay**. Once that happens, the user is prompted to select a save location for the report. If this step is not cancelled, the report is also sent to the [jiang\_voicelab@yahoo.com](mailto:jiang_voicelab@yahoo.com) email.

#### Edit

Graphical user interface, text, application

Description automatically generatedIn this case, there is a SubVI that simply waits for the user to enter a password. Right now, the password is “password” because everyone else will guess “voicelab1.”

If the password is correct, the Admin SubVI will open. This SubVI is set up in a state machine design pattern. The following subsections contain details for each state.

##### Set Display

A picture containing diagram

Description automatically generatedThe first state called sets the display seen earlier in the front panel section of this guide. It is also called after some other cases when updates occur. While I did name it “Set Display” it really just opens the Display.txt file and takes that data and converts it to a 2D Array. It also disables the **Remove**, **Edit In/Out**, **Status**, **Add Time**, **Remove Time**, and the **Select Time to Add** controls.

##### Wait

Graphical user interface, text, application

Description automatically generatedHere we find an event structure. It responds to what the user presses in the front panel. There is a response for each button and other inputs. It also has the Mouse Up case that updates the ActiveCell property. This is important for removing inputs and the Time Select Loop.

Here are the different button presses and events handled by this structure.

* **Edit In/Out Button** – When this button is pressed, the **Remove Time** button, the **Select Time to Add** input, and the **Status** drop-down menu are enabled. The ‘Edit In/Out’ state is called from here as well.
* **Okay Button** – This just calls the ‘Exit’ case.
* **Employee Hours Display** – When the user clicks on a cell in the table of employee hours, this case uses the mouse coordinates to mark the selected cell as active. The ‘Wait’ state is called after this.
* **Add Button** – This just calls the ‘Add Name’ state.
* **Remove Button** – This just calls the ‘Remove Name’ state.
* **Remove Time** – This just calls the ‘Remove Input’ state.
* **Status Drop-Down**  - When the value of the **Status** drop-down menu changes, this case checks what the new value is and enables/disables the **Add Time** button depending on that value. All values except zero will enable the button. The ‘Wait’ case is then called.
* **Select Emloyee Drop-Down** – When the value of the **Select Employee** drop-down menu changes, this case checks the new value and enables the **Edit In/Out** button if that value is not zero.
* **Add Time** – This just calls the ‘Add Input’ state.
* **Select Name Drop-Down**  - When the value of the **Select Name** drop-down menu changes, this case checks the new value and enables the **Remove** button if that value is not zero.

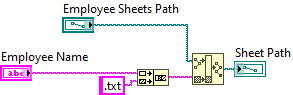
The wait case also opens up the LIST.txt file, removes the empty name (Remove Empty VI), sorts them ([Sort 2D Array VI](#_2D_Array_Sorter)), then adds them to the **Select Name** and **Select Employee** drop-down menus.

###### Remove Empty SubVI

Qr code

Description automatically generatedThis SubVI finds the row index where a space or empty string are found and then deletes them from the array.

##### Edit In/Out

In this state, the Name to Path SubVI is used to build a path to the text file containing the in/out data for the selected employee.

Diagram

Description automatically generatedThis array is then flipped so it shows the most recent times first. Each column (transposed row) is flipped then re-indexed before being transposed again.

##### Add Name

This case adds whatever is in the string input (where you are supposed to enter a name) to the LIST.txt and Display.txt files. This is done by adding a new array for that employee to the Display and List arrays, then re-sorting them to be in alphabetical order. A new employee sheet is also created. Diagram, schematic

Description automatically generated

##### Remove Name

Diagram, schematic

Description automatically generatedThis case removes names from the LIST.txt and Display.txt files. It also deletes the corresponding employee sheet.

First, the column of names (index zero) is extracted from the display and names arrays (red circles). Using Search 1D Array functions, we can find the indices of the selected name (red squares). The entire row is removed from each 2D array (blue rectangles).

##### Add Input

If an employee forgets to check in or out, the admin can add a time and status to their employee sheet. The 2D Hours Array is flipped so new additions are added to the end of the sheet. I currently do not have a sorter for the hours sheet but it is not needed since all of the hours are searched through when making reports.

##### Remove Input

Using the ActiveCell row and value, the row of the selected timestamp is removed from the array. The employee sheet is updated with this new data.

##### Time Select Loop

When the user selects a cell on the spreasheet found on the front panel, the “Employee Hours” case is triggered and the ActiveCell proptery is updated. This loop is constantly updating Selected Time Stamp string. This is just the second column (LabVIEW is zero-indexed) of the Active Cell’s row.