

Homework #5A

LabVIEW

Dr. Pogo

Assignment is due on Thursday, October 12, 2023

Assigned October 3, 2023

Assignment #5: File Read/Write, Dialogs, Cameras. Also, be sure to see part B on the other side of this sheet!

- Inputs:** A “Log in” button, a “Log out” button, a “Quit” button, a “radio button” with 7 choices, an “Add User” button, and a “Delete User” button
- Hardware:** Webcam (See “extra” vi called “how-to-use-initcam” for video testing & structures)
- Outputs:** A “user ID” text indicator
“Logged In” and “Admin” indicator lights
An “Image” indicator
A current date and time indicator (including seconds)
- Required Dialogs:** Login & Password, ‘Really Quit?’, ‘Login Incorrect!’, ‘Really Logoff?’, ‘Nobody is logged in!’, ‘you are not an administrator!’, ‘__ is already logged in!’, ‘__ is already a user!’, ‘no such user!’, ‘Can’t delete yourself!’, ‘New Password may not be blank!’, ‘New User ID may not be blank’, ‘Delete successful!’.
- Submission:** You can name your subvi’s anything you want, but you should submit them in an entire folder using the “usual” filename rules (e.g., name the folder 05-Pogo).

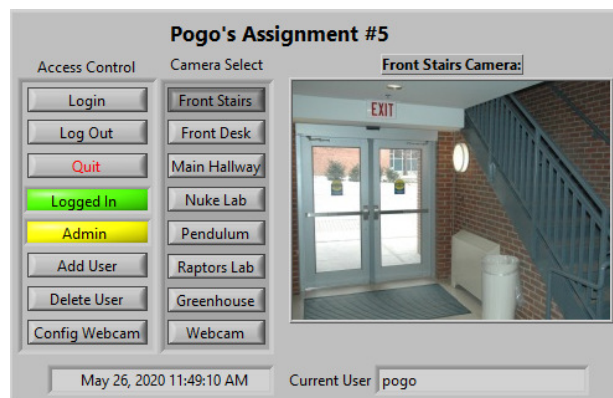
Potential users login by pressing a “login” button, which opens a user ID/password dialog box. The login dialog should initialize blank every time it is used. Typed password characters should appear as asterisks ‘*’ (see text control properties...). The dialog box should have an option to cancel. The user may not attempt to enter multi-line ID’s or passwords. An incorrect login must close the dialog, rather than making it convenient for hackers to quickly retry. Only validated users can log out or set a radio button. The program should not allow a new user to log in until the prior user logs out (a dialog box should appear indicated that *username* is already logged in). The “log out” button should activate a dialog which also allows the user to cancel. The “Quit” button should activate an “Are you sure?” dialog. When the program starts, or when a user logs off, all buttons should initialize off, all text displays should be blank, and the image should be blank. Be sure you understand and use each required dialog perfectly.

When a user is logged in, their user ID should appear in the text box, and the “logged in” light should be on. A recognized user should be able to change the camera choice (and corresponding image) using the radio buttons. One option shows actual live images from a webcam, but a set of seven .jpg pictures are provided to represent 7 other “virtual” cameras. Your program should convert them to images. The default camera selection should be “front stairs”. The image title should change appropriately. Validated users can choose from available cameras and settings. You may use my “initcam” subvi instead of creating your own. Once configured, webcam images should be rescaled to the same width as the jpg pictures. This will change depending on the camera used... and the one I test it with isn’t the same as yours!

The program must use the provided tab-delimited spreadsheet file (passlist.txt) to check whether a user ID/password pair is correct. This file must be opened and then closed each time it needed... you may not load all of the information into memory once at the beginning of the program. So, if I make manual changes (using “Notepad.exe”, for example) to the passlist file *while your program is already running*, then these changes should be available to your program. Be sure that every line ends in a CR/LF. Your program may not add extra blank lines into this file.

Additionally, a user may be flagged as an administrator (and the corresponding light will come on when they are logged in). Administrators may choose to add or delete users from the list. New users must be given a password and are also permitted to be administrators. Administrators may not remove themselves from the list. You may freely use my “delete user” subVI. Make sure that the user may not ever accidentally attempt to enter a two-line ID or password.

Please note that there is an extra section on the grading sheet for this assignment. You may not submit a .zip, but you must submit all required files (images, passlist.txt, your subvi’s, and my subvi’s that you choose to use).



Homework #5B

LabVIEW

Dr. Pogo

Assignment is due on Thursday, October 12, 2023

Assigned October 3, 2023

This is a writing assignment, not a programming assignment.

I will review the solution in class on the due date, so late submissions will *not* be accepted.

If you expect feedback on your work, submit a paper copy. Otherwise, you may submit a pdf file.

A friend of yours made a LabVIEW system to control the HVAC for a 2 story home. On the front panel, the user enters a desired temperature (floating point control) for the home. You want to reach the desired temperature as fast as is reasonable. The system operates the following hardware to achieve this result (your solution may not require the homeowner to buy *any* new hardware):

- Using a DAQ analog input, a thermostat on the 1st floor creates a floating point value for the current temperature.
- Using a DAQ, a Boolean output can be sent to the furnace to turn it on or off.
- Using a DAQ, a Boolean output can be sent to the AC unit to turn it on or off.
- The furnace and AC units are in the same box, and use the same ducts to deliver heated or cooled air to the house.
- There are two fans, each of which can be set to [off, slow, or fast]. Air going to the 1st floor ducts passes through fan #1, and similarly, all air going to the 2nd story passes through fan #2.

Here is the **specification** (or, “spec”) that your friend used to create the system:

While the actual temperature matches the desired temperature, then the furnace, AC, and both fans are turned off. While the actual temperature is too hot, then the AC unit and both fans are turned on “fast”. While the actual temperature is too cool, then the furnace is turned on with both fans on “fast”.

Sadly, the system has several flaws despite *correctly* implementing these steps in hardware. These problems exist because the spec is too vague, incomplete, or naïve. Specifically:

1. All day every day, the fans go on and off with a period of about 3 minutes. While this happens, either the furnace or AC is also being turned off at the same frequency. This is bad because frequent stops and starts severely reduce the life span of the machinery.
2. At thanksgiving, the furnace ran all morning because it was cold outside. But, near noon, while the turkey was cooking, the system suddenly started running the AC unit. Then, when the turkey was done, the system changed back to running the furnace. This seems wasteful of energy to the homeowner.
3. During both summer and winter, the 1st floor is typically 10 degrees colder than the 2nd story. Even worse, this problem exists to a lesser degree throughout the entire year!

You need to write a complete one-page (**or preferably much less**) specification for a new LabVIEW system that solves all of these issues using the same hardware and front panel. Your solution must work whether the home is in Alaska, Hawaii, or Iowa. You do not need to write the LabVIEW program, you just have to specify what it must accomplish. It must be ultra-explicit about the outcomes: if *this* then *that*, unless something *else*. You may not re-describe the problems listed above, you must merely state what you want the program to *do* under all possible circumstances. That is, the spec describes the *what*, but not the *why*. The spec will then be given to a programmer who will implement your spec without consulting you. This spec will be graded on:

- whether the programmer might misunderstand what is expected,
- whether it solves all the problems listed,
- whether it is concise.

Additionally, on the same page but after and separate from the spec, write three one-sentence explanations describing how the original spec led to the three problems described above. Hint: ask your parents about the real magnitudes of differences in temperature settings that cause them to argue with each other!