Testing of demos:

Use case: Register a Scout

1. Date of Birth: Should be validated to check that the user enters the date in YYYY-MM-dd form, and *that is the form into which it is entered into the database.* (Demo of value in the database is necessary). Suggestion: use the DatePicker class, and if necessary, the DateFormat and SimpleDateFormat classes to store date in String form.
2. Phone Number: Validate that it is entered in *one* of the following formats:
3. (xxx)xxx-xxxx
4. xxx-xxx-xxxx
5. xxxxxxxxxx (exactly 10 characters)
6. Each ‘x’ must be a value from 0 – 9 (nothing else).

Demo of value stored in the database is necessary

1. Email: Validate that the string entered contains the ‘@’ symbol. Demo of value stored in database is necessary
2. TroopID: Validate that it is a sequence of 5 digits (it is actually 9 for the local council, but let us go with 5). *IMPORTANT:* Must enter a Scout with a certain value for a TroopID and then try to enter another one (different names) with the *same* TroopID and that should *fail*. Demo that the new scout was *not* stored in the database.

Of course, must demo that the new scout exists in database.

Use Case: Update a Scout

1. The GUI screen for update should not allow changing for the TroopID (suggestion, rechecking for uniqueness and handling errors would be too much in the current situation)
2. One or more names will be changed: Testing of new name will check for *length*. If the length exceeds the max allowed length in the databases, the name should be rejected. See this image:



Of course, must demo that the database contains ALL the updated data (including new phone number, email, date of birth, etc. – if changed)

Use case: Remove Scout

1. Select any scout to remove. Demonstrate that in the database, the selected Scout’s status is shown as “Inactive”
2. *IMPORTANT:* Based on the decisions made during today’s (3/27/2023) meeting of leaders present: Once a scout is removed, go back to the Update Scout use case and only demonstrate *search for scout* using name strings (first name and last name) in which the just-removed Scout will be included as *one* of the scouts that could show up. BUT..the just-removed Scout should *NOT* show up in the list of scouts, although the other matching names should.

Use Case: Add a Tree

1. The GUI screen should only allow for entry of barcode and Notes. All other values in the Tree record should be computed or set as default (e.g., Status is by default “Available”)
2. Barcode: Validate that it is exactly 5 digits long. Also, the barcode should have one of the prefixes shown in the table below.

+----+------------------------+------+---------------+

| ID | TypeDescription | Cost | BarcodePrefix |

+----+------------------------+------+---------------+

| 1 | Frasier Fir - Regular | 45 | 20 |

| 2 | Frasier Fir - Premium | 45 | 21 |

| 3 | Douglass Fir - Regular | 45 | 30 |

| 4 | Douglass Fir - Premium | 50 | 31 |

| 5 | Blue Spruce - Regular | 45 | 40 |

| 6 | Blue Spruce - Premium | 50 | 41 |

| 7 | Concolor - Regular | 45 | 50 |

| 8 | Concolor - Premium | 50 | 51 |

| 9 | Balsam Fir - Regular | 45 | 60 |

| 10 | Balsam Fir - Premium | 50 | 61 |

+----+------------------------+------+---------------+

1. Test with a barcode that does NOT begin with one of the above prefixes and return “ERROR: Invalid barcode” on the screen
2. Test with Notes field NOT empty – have something like “Charlie Brown tree – sell for $ 30 if possible” in this field

Of course, the added Tree should be shown in the database. If “Notes” field not empty for the tested tree, show in the database that the Notes value exactly as entered is in the database.

Use Case: Update a Tree

1. The GUI screen for this update should NOT allow the barcode of the tree to be changed.
2. Notes: Provide new notes. *IMPORTANT*: If the “Notes” field is changed to be longer than what the database can accommodate, the new “Notes” value should be rejected.
3. Status: Change status from “Available” to “Damaged” for this test

Of course, show the new tree data (new Notes value, new Status value) in the database

Use Case: Remove a Tree

1. Database must have a few Tree records in it already to test this use case. Edit one database record to change the Status of the tree from “Available” or “Damaged” to “Sold”. Note barcode of this tree.
2. Sold Tree: Provide barcode of already sold tree, and try to remove it. GUI should be updated with message: “ERROR: Tree already sold”
3. Not sold Tree: Provide barcode. Demonstrate that the tree with this barcode is *actually not present* in the database table

Use Case: Add Tree Type

1. Barcode prefix: Must be *exactly 2 digits*. Validate that and provide error message on GUI screen if not.
2. Cost: Must be a double value. Provide an error message if not.

Of course, show that the new tree type is added to the database

Use Case: Update Tree Type

1. Cost: Change cost of tree type, and validate that it is a double value. Provide an error message if not.

Of course, show that the new cost is in the database for the selected tree type