User Interface/Front End Design and Related SQL Queries

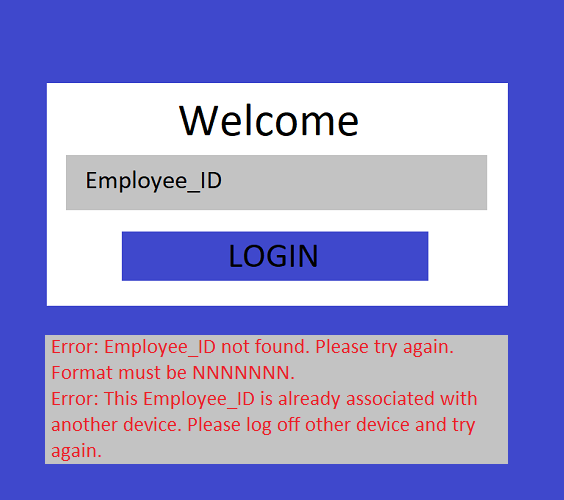
The user interface or front end design for the order picker is a web application (different interfaces would be required for other roles accessing the database, such as managers or inventory specialists).

We did not create the application itself—this is an example of what it would look like, as well as the SQL queries that would be utilized by the API. We have the beginnings of a Java-based interface on the AWS cloud, as well as the beginnings of a Python script to access the database, so those would be options for possible starting points in future development. The Java and Python files have been included in the submission.

Before the employee logs in for the first time, the employee table will have been updated with their employee id and name, by whomever has access to that table, likely their manager or perhaps HR:

INSERT INTO employee (employee\_id, first\_name, last\_name) VALUES

When the employee boots up the application, they will see the Welcome/Login screen:



The employee will enter (by keypad or scanning a barcode) their employee id. For security, before accessing the database itself, the application will always validate the input to make sure it is what is expected. For the employee id, the application will first check if it is a valid 7-digit integer (and return the first error shown if not). Then it will check if the employee id is found in the database:

SELECT COUNT(1) FROM employee WHERE employee\_id = <input value>

If not found, it will return the first error and get new input from the employee. If found, the database will then check if the id is already assigned to a device:

SELECT COUNT(1) FROM employee\_assigned\_device WHERE employee\_id = <input value>

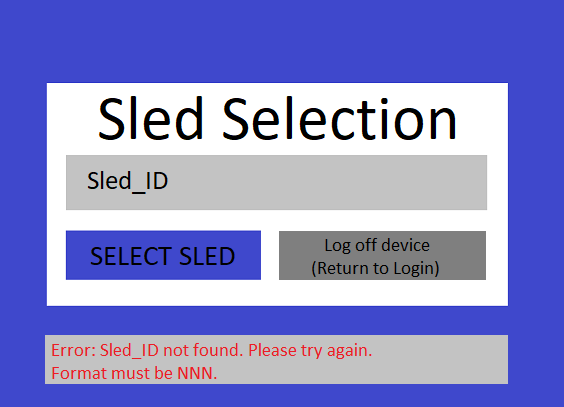
If found, the application will return the second error shown. If not found, the employee\_id and device\_id will be inserted into the employee\_assigned\_device table (device\_id is determined by the tablet on which the application is running and not input by the user):

INSERT INTO employee\_assigned\_device (device\_id, employee\_id) VALUES (<device value>, <input value>)

The device will be updated to checked out:

UPDATE device SET checked\_out = 1 WHERE device\_id = <device value>

The web application will move to the next page, sled selection:



The sled\_ids will have already been added to the sled table by those with the appropriate access:

INSERT INTO sled (sled\_id) VALUES

The employee will enter (by keypad or scanning) the sled id. The application will first check if it is a valid integer (range 1-999) and return the error shown if invalid. Then it will check if the id is in the database:

SELECT COUNT(1) FROM sled WHERE sled\_id = <input value>

If not found, it will return the error and get new input. If found, it will check if the sled has already been assigned to another employee:

SELECT COUNT(1) FROM employee\_select\_sled WHERE sled\_id = <input value>

If the sled in question is already assigned to an employee, an error will display. If not found, the sled\_id and employee\_id will be added to the employee\_select\_sled table:

INSERT INTO employee\_select\_sled (sled\_id, employee\_id) VALUES (<input value>, <previously input value>)

The sled will be updated to checked out:

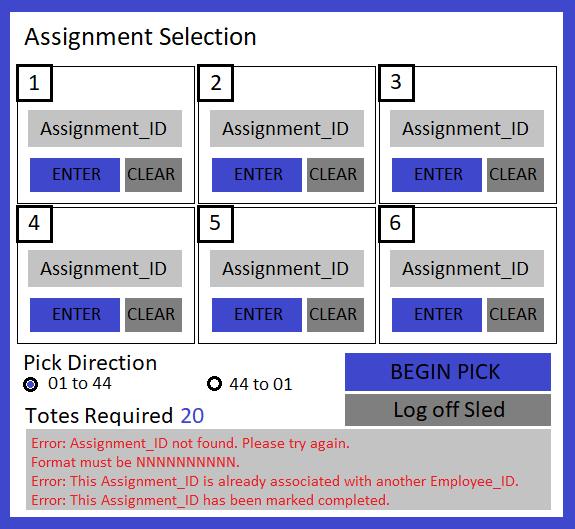
UPDATE sled SET checked\_out = 1 WHERE sled\_id = <input value>

If Log off device (Return to Login) is selected instead, the device will be unassigned from the employee and set to checked in:

DELETE FROM employee\_assigned\_device WHERE device\_id = <device value>

UPDATE device SET checked\_out = 0 WHERE device\_id = <device value>

Once the sled selection is confirmed and assigned to the employee, the web application continues to the assignment selection page:



The assignment\_ids will have been previously added to the assignment table, likely on a daily basis, and possibly automatically through a separate interface when the stores place orders:

INSERT INTO assignment (assignment\_id, batch\_num, store\_num) VALUES

The employee will enter (by keypad or scanning) the assignment id(s). They can enter between 1 and 6 assignments (they must enter at least 1 to begin picking). The application will validate these similarly to the employee\_id and sled\_id (confirming the integer input and checking the database):

SELECT COUNT(1) FROM assignment WHERE assignment\_id = <input value>

SELECT COUNT(1) FROM sled\_holds\_assignment WHERE assignment\_id = <input value>

To display the number of totes currently required, the application sums the number of totes in each assignment entered:

SELECT COUNT(1) FROM assignment\_contains\_tote WHERE assignment\_id = <input value>

After BEGIN PICK is selected, the database updates the assignments to assigned and connects them to the sled:

INSERT INTO sled\_holds\_assignment (sled\_id, assignment\_id) VALUES (<previously input value>, <input value>)

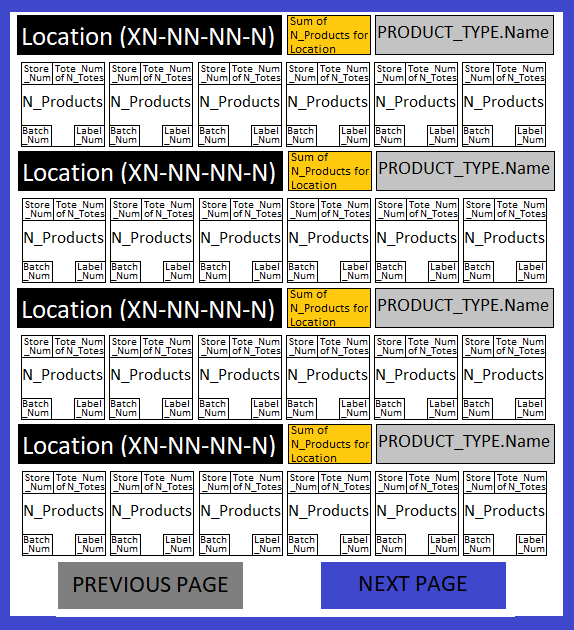
UPDATE assignment SET assigned = 1 WHERE assignment\_id = <input value>

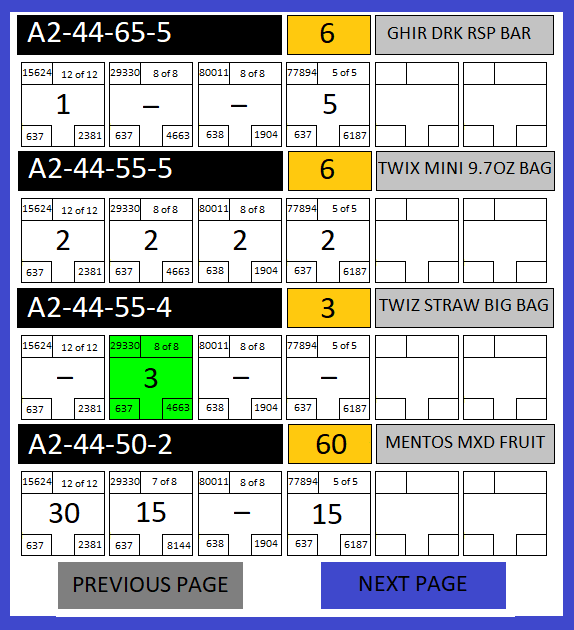
If Log off Sled is selected instead of Begin Pick, the sled is unassigned to the employee, marked to checked in, and the web application returns to the sled selection page:

DELETE FROM employee\_select\_sled WHERE sled\_id = <previously input value>

UPDATE sled SET checked\_out = 0 WHERE sled\_id = <previously input value>

Once assignments are selected, the pick pages are generated:





The application compiles all of the products for each tote in each assignment that is assigned to the sled. The products are put in a list with no duplicates to indicate that the employee will only visit each location once and pick all the products required for the whole sled. The implies that the assignments will have some of the same products as each other, which would be ensured by checking that the zones of the assignments match. We did not include this aspect in our database, for simplicity. The products will display in ascending or descending bay order as determined by the user selection on the assignment selection page.

The information on the pick pages is pulled from the database by using each assignment\_id on the sled to pull product and tote information for all of the totes in that assignment:

SELECT S\_h\_A.sled\_id, A.assignment\_id, T.tote\_id, T\_h\_P.n\_products,

P.name, P.floor\_letter, P.floor\_num, P.bay\_num, P.col\_num, P.row\_num

FROM sled\_holds\_assignment as S\_h\_A LEFT JOIN

(assignment as A LEFT JOIN

(assignment\_contains\_tote as A\_c\_T LEFT JOIN

(tote as T LEFT JOIN

(tote\_has\_product as T\_h\_P LEFT JOIN product\_type as P

ON T\_h\_P.product\_id = P.product\_id)

ON T.tote\_id = T\_h\_P.tote\_id)

ON A\_c\_T.tote\_id = T.tote\_id)

ON A.assignment\_id = A\_c\_T.assignment\_id)

ON S\_h\_A.assignment\_id = A.assignment\_id;

The database is only called once to create this table and the pick pages are generated at that time. Then the products are displayed four at a time. Once each product on the page has been picked and placed into the corresponding tote, the employee selects Next Page and the next products in order are shown. Upon reaching the last pick, instead of a Next Page option the option will be to Drop Assignments. Upon selecting Drop Assignments, the assignments will be marked completed and the employee will be returned to the assignment selection page, where they can either log off or select more assignments for picking:

DELETE FROM sled\_holds\_assignment WHERE assignment\_id = <previously input value>

UPDATE assignment SET assigned = 0 WHERE assignment\_id = <previously input value>