# Quality Audit Page

An important factor in the trainee’s development skills is the feedback from the QC (Quality Control) Team. This feedback is used to monitor the trainee’s individual progress over the course of the training program, as well as the entire group’s progress.

Quality Audits are performed once a week for every group, usually in the form of a verbal interview. Trainees are evaluated using verbal interviews and judged in qualitative way. This one-on-one interaction between the trainees and the QC team gives them an idea on how the batch is performing.

With the Caliber application, the QC teams can enter their assessments via the Quality Audit page, where they can give each trainee a grade based on how well they performed, as well as leave additional notes, such as what was asked how confident the trainee was in their answer.

Performing this quality audit is done in three main steps: Selecting the specific batch to perform the audit on, selecting the desired week, finally evaluating the batch.

All of this information can be viewed in real time by other departments as it becomes available, and if necessary, informed decisions can be made quickly in order to make corrective measures for unexpected developments.

The Quality Audit page was designed and developed using lighting components, providing a modular approach with increased code reusability that meet perform the necessary functions.

Additionally, the audits are also a way for the trainees to realize what topics are they fluent in, and which ones they need to brush up on.

## QA Batch Selector

The QC teams can search through all of the current and past training batches thorough this component. This is done by querying and narrowing down options through the 4 dropdowns at the top of the page. Each drop down menu acts as a filter, and narrows down the available batches by Year, Quarter, Location, and individual batch. An individual batch is identified by the trainer’s name and the groups starting date.

All of the dropdown menus are dynamically binded to each other, so that if the user changes the year, the available options in the rest of the menus will be updated to show only valid options. This is done by having the necessary menus query and update themselves whenever one of the menus to the left hand side changes values.

Once a batch is selected, the rest of the page is updated to show the data related to that specific batch.

#### Methods:

* Init: Runs when the component is created. It calculates the current date and sets the first dropdown menu’s selected item to the current year. After this the other drop down menus can be populated.
* updateQuarterLabel: updates the available options on the Quarter’s menu based on which quarters have batches in them. Quarters that do not have batches are excluded from the available options on the menu.
* updateLocationLabel: updates the options on the Location drop down menu.
* updateTrainingLable: updates the final dropdown menu to show the batches that match the conditions established by the other 3 drop down menus.
* updateTraining: fires an event passing the currently selected batch id to other components.
* changeTraining: queries and populates the available batches options for the last menu.
* buildQuarterForYear: calculates and populates what year quarters to include in the second menu.
* buildLocations: queries and creates the available locations for the third menu.
* buildTrainingStrings: creates the strings that show as the option for each batch for the last menu. The strings are made by joining the batch’s trainer name and the starting date.
* updateTheBatchId: fires an event and passes the newly selected batch id to other components.

## QA Weekly Tabs Component

After the user has selected a specific training batch, they can click on any of the week tabs in order to shift between the available weeks and see how that group did for that particular week.

The tabs used belong to a lighting tabset, where each tab is a separate container. Clicking on any one of the tabs will hide the content of the previously visible tab and lazy-load content into it by using the onactive attribute, meaning that once the tab becomes active, it injects the content into its body programmatically.

By default, there will always be at least 1 week tab. The user can however create new tabs by clicking on the add tab button marked with the + symbol. Doing so adds a new week tab and creates blank quality audit notes for all of the trainees in the group.

This component uses the apex controller batchTableController, do perform queries into the database, to increment the batch’s current week and to create blank notes after a new week has been created.

Methods:

* doInit: Runs on component creation. Is in charge to creating the necessary amount of tabs to meet the selected training batch current week number, as well as set the currentBatchID attribute.
* handleActive: Sets the currently active content when a user clicks on one of the week tabs. Afterwards, it inserts a batchTable component into the tabs body.
* handleAddTab: Calls the incrementWeek method from the apex controller, which will increment the Caliber\_Number\_of\_Weeks\_\_c field by 1, and creates a new tab with blank grade scores and notes for all of the trainees.
* handleChange: Responds to the updateBatchIDEvent which is fired when the user changes the selected batch. Since the event passes the newly selected ID, the method saves it into the currentBatchID attribute and reloads the entire component.
* injectComponent: Programmatically creates a new batch table component and inserts it into the Week tab’s body. This uses the $A.createComponent function to pass attribute values into the newly created component.
* addTab: Programmatically creates a new tab and adds it into the tabset. Since each tab is numbered and has a different id, it first has to determine how many tabs already exist and then it creates a new one, passing the necessary attributes to give it the desired values and functionality. Afterwards the new tab is added into the tabset.
* reloadOnEventFire: Performs the same actions as the doInit function, thereby reloading the entire component in order to show the proper elements to match the newly selected batch.

## QA Batch Table

When a given tab becomes active, an instance of this component is inserted into the tab’s body. The Batch Table component is the main container for all of the information of a batch’s QC audits for a specific week.

Each Batch Table will show the categories covered in assessments for that week, a row for each enrolled trainee, a group overall feedback section, and a save button at the end. Each of these elements is housed by a separate component, where the rows for each trainee are iterated through, so that the number of rows matches the number of trainees.

This component is also in charge of passing attributes into its subcomponents.

The Batch table components use the apex controller batchTableController, in order to save and retrieve data from the org.

Methods:

* doInit: Calls the apex controller in order to retrieve all of the trainees that are assigned to the currently selected batch.
* saveNotes: Saves the overall feedback note after it has been blurred off.
* updateTheBatchID: Used to reset the currentBatchID attribute when a new training group is selected from the QA Select.

## Categories Covered Component

This component queries the caliber assessments for the week for a specific batch, and retrieves the categories of those assessments. This gives the QC team information on what topics or content the group went over in the previous week.

The component takes in two attributes: the Id of the selected Training and the number of the current week. Internally, it has one additional String attribute that contains the retrieved categories, which it displays inside of the components body.

Methods:

* doInit: Runs as soon as the component is created, and passes both the Training Id and the current Week Number to an action that calls the apex controller. Once it receives the action’s answer, it saves that answer into the categories attribute.
* getCategories: Takes in the Training’s ID and the week’s number, and uses both to query for all of the Caliber Assessments that match both values. Once the Assessments have been retrieved, the category for each assessment is extracted, and then a single string is built containing those categories. If there were multiple assessments with the same category, then the category will only appear once.

## QA Trainee Rows:

This component displays the name of each trainee, their QA assessment value for the week, and a note from the QC team for that trainee using information taken from the Salesforce database.

#### Methods:

* getEverything: fetches information about a trainee from the database using a passed training assignment Id. Using this Id, this function then fetches information about the caliber note for the selected week.
* handleClick: sets QA assessment face and tooltip text and saves value to database. Also fires an event to update the overall feedback.
* handleBlurOnNote: saves note value to the database when the user focuses away from the text area.

## Save Button:

The save button at the bottom of the page calls the doBurrito function and mainly exists so that the user focuses off of the final component on the page when they click on the button so that the onBlur action for the text area can fire and save trainee note information to the database.

#### Methods:

* doBurrito: hides the save button and replaces it with a spinning icon to indicate that data is being saved. After two seconds, the spinning icon is replaced with a checkmark to indicate that the saving process has completed. Two seconds later, the icon disappears and the save button returns to take its place.