User Stories

# View and Manage Underwriting (UW) Comments for a Specific Policy

Type: QC

Detailed description: As a user, I want to view and manage underwriting (UW) comments for a specific policy, so that I can review and update the comments based on the user's profile and policy reference.  
  
Acceptance criteria:  
1. The system should retrieve the current date and set it in the control block.  
2. The system should maximize the main window and the specific window for the QC module.  
3. The system should search for the policy based on the provided proposal number.  
4. The system should create a timer named 'VIEW\_IMAGE' with a duration of 100 milliseconds.  
5. The system should retrieve the current user and their profile.  
6. If the user does not have a specific prefix (P00% or UU%), the user ID field in the UW block should be hidden.  
7. The system should clear the UW block and display the first record.  
8. If the user has a specific prefix or is a supervisor, the system should retrieve and display all comments for the policy.  
9. If the user does not have the specific prefix and is not a supervisor, the system should retrieve and display comments based on specific conditions.  
10. The system should also retrieve and display comments from another table based on the policy reference.  
11. If no comments are found, the system should display a warning alert indicating that there are no UW comments for the policy number.  
  
Definition of Done:  
- The user can view and manage UW comments for a specific policy.  
- The system retrieves and displays comments based on the user's profile and policy reference.  
- The system handles cases where no comments are found and displays appropriate alerts.  
- The system maximizes the relevant windows and sets the current date in the control block.  
- The system creates and manages a timer for the 'VIEW\_IMAGE' functionality.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Retrieve current user:  
 ```sql  
 SELECT USER INTO :control.users FROM DUAL;  
 ```  
  
- Retrieve user profile:  
 ```sql  
 SELECT user\_profile INTO v\_uw\_profile FROM azbj\_user\_prof\_ref WHERE user\_name = :control.users;  
 ```  
  
- Retrieve comments for the policy:  
 ```sql  
 SELECT FROM azbj\_uw\_comments WHERE policy\_no = :control.pol\_ref;  
 ```  
  
- Retrieve comments with specific conditions:  
 ```sql  
 SELECT FROM azbj\_uw\_comments WHERE policy\_no = :control.pol\_ref AND 0 = (CASE WHEN :control.users LIKE 'UU%' AND v\_uw\_profile <> 'SUPERVISOR' THEN (CASE WHEN (user\_id LIKE 'P00%' AND NVL(flag, 'N') = 'Y') THEN 1 ELSE 0 END) WHEN NVL(flag, 'N') = 'N' THEN 0 ELSE 1 END);  
 ```  
  
- Retrieve additional comments:  
 ```sql  
 SELECT user\_id, co\_date, uw\_comments FROM azbj\_ri\_co\_details WHERE policy\_ref = :control.pol\_ref ORDER BY co\_date;  
 ```

# Implement QC Section with Specified Fields and Properties

Type: QC

Title: Implement QC Section with Specified Fields and Properties  
  
Acceptance Criteria:  
1. The QC section should display up to 30 records at a time.  
2. The following fields should be available in the QC section:  
 - FLG: A numeric field.  
 - DESC: A display-only text field with a maximum length of 200 characters.  
 - OPUSVAL: A read-only text field with a maximum length of 5000 characters.  
 - CI: A radio group with an initial value of 'N'.  
 - SCAN\_DATE: A date field.  
 - INS\_DT: A date field.  
 - FLG1: A numeric field.  
 - MIS\_USER: A hidden text field with a maximum length of 50 characters.  
3. The fields should be displayed with appropriate formatting, colors, and positions as specified.  
4. The QC section should be part of a window titled "QC Correction" with a vertical scrollbar.  
  
Definition of Done:  
- The QC section is implemented with all specified fields and properties.  
- The section displays up to 30 records at a time.  
- All fields are correctly formatted and positioned.  
- The window titled "QC Correction" includes a vertical scrollbar.  
- The functionality is tested and verified to meet the acceptance criteria.

# Handle Selection and Validation of Radio Button in Form

Type: QC

Title: Handle Selection and Validation of Radio Button in Form  
  
Acceptance Criteria:  
1. When the user navigates through the records and reaches the last record, the system should automatically navigate to the first record.  
2. When the user changes the radio button selection:  
 - If the selected option is 'N' and a specific flag is set, the system should display an alert prompting the user to select an error category first and prevent further actions.  
 - If the selected option is 'N' and another specific flag is set, the system should check a condition in the database and display an alert if the condition is met.  
 - If the selected option is 'Y' and specific flags are set, the system should reset certain fields to 'N'.  
  
Definition of Done:  
- The system correctly navigates to the first record when the last record is reached.  
- The system displays appropriate alerts based on the selected option and flag conditions.  
- The system updates the fields correctly based on the selected option and flag conditions.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- SELECT SOCIAL\_STATUS INTO v\_sector FROM AZBJ\_POLICY\_BASES\_EXT WHERE CONTRACT\_ID = azbj\_pk0\_acc.get\_contract\_id (:control.pol\_ref) AND TOP\_INDICATOR ='Y';

# Automatic Record Navigation on Down Arrow Key Press

Type: QC

Title: Automatic Record Navigation on Down Arrow Key Press  
  
Acceptance Criteria:  
1. When the user presses the down arrow key:  
 - If the current record is the last record, the system should navigate to the first record.  
 - If the current record is not the last record, the system should navigate to the next record.  
  
Definition of Done:  
- The system correctly identifies the last record.  
- The system navigates to the first record when the down arrow key is pressed on the last record.  
- The system navigates to the next record when the down arrow key is pressed on any record other than the last record.  
- The functionality is tested and verified to work as expected.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided logic does not involve direct database CRUD operations.

# Display Insurance Rider Details

Type: RIDERS1

Title: Display Insurance Rider Details  
  
Acceptance Criteria:  
1. The system should display the following details for each insurance rider:  
 - Cover Code  
 - Sum Insured  
 - Benefit Term  
 - Premium Term  
2. Each detail should be displayed in a read-only format.  
3. The details should be aligned and formatted consistently for easy readability.  
  
Definition of Done:  
1. The user can see the cover code, sum insured, benefit term, and premium term for each insurance rider.  
2. The information is displayed in a read-only format.  
3. The display is aligned and formatted consistently.  
4. The feature has been tested and verified for accuracy and usability.

# Display Pre-Print QC Information

Type: BLK\_PRE\_PRINT

Title: Display Pre-Print QC Information  
  
Acceptance Criteria:  
1. The pre-print QC information should be displayed in a read-only format.  
2. The user ID, address, telephone number, and mobile number fields should be visible and correctly formatted.  
3. The appointee's name, date of birth, relation, and gender should be displayed as read-only fields.  
4. The information should be presented in a clear and organized manner, with appropriate labels for each field.  
  
Definition of Done:  
1. The pre-print QC information is displayed on the screen.  
2. All fields are read-only and correctly formatted.  
3. The information is organized and labeled clearly.  
4. The feature has been tested and verified for accuracy and usability.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Not applicable as the provided XML content does not include any specific database queries or operations.

# View Fund Details in Read-Only Format

Type: FUND

Title: View Fund Details in Read-Only Format  
  
Acceptance Criteria:  
1. The fund details should be displayed in a grid format with the following columns:  
 - Fund ID  
 - Fund Description  
 - PR Apportionment  
2. Each column should have a fixed width and should be aligned to the start.  
3. The background color of the fields should be gray, and the text should be black.  
4. The font used should be Tahoma with a size of 8pt, plain style, and medium weight.  
5. The fields should be non-editable, meaning users should not be able to insert or update any information in these fields.  
  
Definition of Done:  
- The fund details are displayed in a grid format with the specified columns and properties.  
- The fields are non-editable and meet the specified design criteria.  
- The user can view the fund details without the ability to modify them.

# Manage Insurance Policy Covers

Type: RIDERS

Title: Manage Insurance Policy Covers  
  
Acceptance Criteria:  
1. The system should allow the user to input and save the cover code, which is a mandatory field.  
2. The system should allow the user to input and save the sum insured for the whole cover, which should be a numeric value.  
3. The system should allow the user to input and save the benefit term, which should be a numeric value.  
4. The system should allow the user to input and save the premium term, which should be a numeric value.  
  
Definition of Done:  
1. The user can successfully input and save the cover code, sum insured, benefit term, and premium term.  
2. The system validates that the cover code is provided and that the sum insured, benefit term, and premium term are numeric values.  
3. The data is stored in the database and can be retrieved for display and further processing.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- No specific DB queries provided in the XML content.

# Manage Fund Information

Type: FUND1

Title: Manage Fund Information  
  
Acceptance Criteria:  
1. The system should allow users to insert new fund records with the following details:  
 - Fund ID  
 - Fund Description  
 - PR Apportionment  
2. The system should allow users to update existing fund records with the same details.  
3. The input fields for Fund ID, Fund Description, and PR Apportionment should be displayed in a user-friendly manner with appropriate labels and input validation.  
4. The input fields should have a maximum length of 100 characters.  
5. The input fields should be styled consistently with the rest of the application, including font size, font style, and colors.  
  
Definition of Done:  
- The functionality to insert and update fund records is implemented and tested.  
- The input fields for Fund ID, Fund Description, and PR Apportionment are displayed correctly and are user-friendly.  
- Input validation is in place to ensure data integrity.  
- The feature is reviewed and approved by stakeholders.  
- The feature is deployed to the production environment.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include specific SQL queries or table references.

# User Interface for Reprinting Option Selection

Type: PHY\_COPY

Title: User Interface for Reprinting Option Selection  
  
Acceptance Criteria:  
1. The interface should have a radio button group for selecting the reprinting option.  
2. The interface should have a push button labeled "Ok" to confirm the selection.  
3. The "Ok" button should be clearly visible and accessible, with appropriate styling for usability.  
  
Definition of Done:  
1. The radio button group for reprinting options is implemented and functional.  
2. The "Ok" button is implemented, styled, and functional.  
3. The interface is tested for usability and accessibility.  
4. The functionality is reviewed and approved by stakeholders.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include any specific database queries or operations.

# Reprint Physical Move Documents

Type: PHY\_COPY

Title: Reprint Physical Move Documents  
  
Acceptance Criteria:  
1. The reprint functionality should be accessible through a radio button option.  
2. The reprint option should be initialized with a default value of "X".  
3. The reprint functionality should be displayed on a specific section of the user interface, ensuring it is easily accessible and visible to the user.  
  
Definition of Done:  
1. The reprint functionality is implemented and accessible via a radio button.  
2. The default value for the reprint option is set to "X".  
3. The user interface is updated to include the reprint option in the designated section.  
4. The feature is tested and verified to ensure it works as expected.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include any specific database queries.

# Update Physical Copy Status and Provide Feedback

Type: PHY\_COPY

Title: Update Physical Copy Status and Provide Feedback  
  
Acceptance Criteria:  
1. When the "Ok" button is pressed, the system should check if the selected option for physical copy status is either 'Y' or 'N'.  
2. If the selected option is valid ('Y' or 'N'), the system should update the `azbj\_phub\_qc\_status\_detail` table with the new status for the given policy number.  
3. If the update is successful, the system should clear the form and move the cursor to the policy reference field.  
4. If the update is not successful, the system should display a warning message "Data Not Updated."  
5. If the selected option is invalid, the system should display an error message "Choose any one of the above options."  
6. The system should commit the transaction after a successful update.  
  
Definition of Done:  
- The physical copy status is updated in the database when a valid option is selected.  
- Appropriate feedback messages are displayed based on the outcome of the update.  
- The form is cleared, and the cursor is moved to the policy reference field after a successful update.  
- The transaction is committed after a successful update.  
  
DB queries for Table reference CRUD operations only(With Usage):  
```sql  
UPDATE azbj\_phub\_qc\_status\_detail  
SET PHYSICAL\_COPY\_STATUS = :PHY\_COPY.PHY\_MOVE\_REPRINTING  
WHERE policy\_no = :control.pol\_ref;  
```

# User Authentication

Type: BLK\_AUTH

Title: User Authentication  
  
Acceptance Criteria:  
1. The user should be able to enter comments in a text area with a maximum length of 500 characters.  
2. The user should be able to enter their supervisor ID in a text field with a maximum length of 10 characters, which should be automatically converted to uppercase.  
3. The user should be able to enter their password in a text field, which should be concealed for security purposes.  
4. The user should be able to click an "Authenticate" button to submit their credentials for authentication.  
  
Definition of Done:  
1. The comments text area should be displayed with a label "Comments".  
2. The supervisor ID text field should be displayed with a label "Supervisor ID" and should convert input to uppercase.  
3. The password text field should be displayed with a label "Password" and should conceal the input.  
4. The "Authenticate" button should be displayed and functional, allowing the user to submit their credentials.  
5. The authentication window should be modal, preventing interaction with other windows until the authentication process is complete.  
6. The authentication window should have a title "Authentication" and should not be resizable, minimizable, or maximizable.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include any specific database queries.

# Secure Password Entry in Authentication Form

Type: BLK\_AUTH

Title: Secure Password Entry in Authentication Form  
  
Acceptance Criteria:  
1. The password field should be concealed to ensure that the entered password is not visible.  
2. Upon entering the password, the visual attribute of the password field should change to indicate that the field has been interacted with.  
  
Definition of Done:  
1. The password field is implemented and concealed.  
2. The visual attribute changes upon interaction with the password field.  
3. The authentication form is displayed correctly with the password field in the specified position.  
4. The form is modal, ensuring that the user must complete the authentication before interacting with other parts of the application.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as there are no CRUD operations mentioned in the provided XML content.

# Supervisor ID Input Field

Type: BLK\_AUTH

Title: Supervisor ID Input Field  
  
Acceptance Criteria:  
- The input field for Supervisor ID should be clearly labeled.  
- The input field should accept a maximum of 10 characters.  
- The input field should automatically convert all input to uppercase.  
- Upon entering the Supervisor ID, the input field should visually indicate that the data has been entered correctly by changing its appearance.  
  
Definition of Done:  
- The input field for Supervisor ID is implemented and visible on the authentication screen.  
- The input field accepts up to 10 characters and converts them to uppercase.  
- The input field changes its visual attributes (e.g., color, font) upon data entry to indicate successful input.  
- The authentication screen is fully functional and accessible to users.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include any specific database queries.

# Implement Multi-line Comments Field in Authentication Module

Type: BLK\_AUTH

Title: Implement Multi-line Comments Field in Authentication Module  
  
Acceptance Criteria:  
1. The comments field should allow up to 500 characters.  
2. The comments field should be displayed with a specific visual style, including font type, size, and color.  
3. The comments field should be positioned correctly within the authentication module.  
4. The comments field should support multi-line input.  
5. Upon exiting the comments field, the visual style should be updated to a predefined attribute.  
  
Definition of Done:  
1. The comments field is implemented and allows up to 500 characters.  
2. The comments field is styled according to the specified visual attributes.  
3. The comments field is correctly positioned within the authentication module.  
4. The comments field supports multi-line input.  
5. The visual style of the comments field is updated upon exiting the field.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Not applicable as the provided XML content does not include any specific database queries.

# Supervisor Authentication and Approval

Type: BLK\_AUTH

Title: Supervisor Authentication and Approval  
  
Acceptance Criteria:  
1. The system should prompt the user to enter the Supervisor ID if it is not provided.  
2. The system should prompt the user to enter the Password if it is not provided.  
3. The system should validate the Supervisor ID and Password using an approval procedure.  
4. If the credentials are valid, the system should:  
 - Set an approval flag.  
 - Display an "Approved" message.  
 - Insert a record into the approval log with the policy reference, supervisor ID, comments, and the current date.  
 - Navigate to the next item for further processing.  
5. If the credentials are invalid, the system should display an error message with the reason for the failure.  
6. In case of any other errors during the authentication process, the system should display a generic error message.  
  
Definition of Done:  
- The user is able to authenticate using Supervisor ID and Password.  
- Appropriate messages are displayed for missing or invalid credentials.  
- Successful authentication results in an approval log entry and navigation to the next item.  
- Error handling is implemented for unexpected issues during the authentication process.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Insert into `azbj\_qc\_fail\_approval` table:  
 ```sql  
 INSERT INTO azbj\_qc\_fail\_approval (POLICY\_REF, SUP\_USER\_ID, SUP\_COMMENTS, CREATE\_DATE)  
 VALUES (:CONTROL.POL\_REF, :BLK\_AUTH.LOGIN, :BLK\_AUTH.REMARKS, SYSDATE);  
 ```  
  
- Update and insert into `azbj\_phub\_qc\_status\_detail` table:  
 ```sql  
 UPDATE azbj\_phub\_qc\_status\_detail  
 SET status = st, substatus = subst, ins\_date = SYSDATE  
 WHERE policy\_no = :control.pol\_ref;  
  
 IF SQL%NOTFOUND THEN  
 INSERT INTO azbj\_phub\_qc\_status\_detail(POLICY\_NO, CONTRACT\_ID, STATUS, SUBSTATUS, DE1, DE2, BBU, LOCK\_FLG, INS\_DATE)  
 VALUES (:control.pol\_ref, p\_con\_id, st, subst, p\_de1, p\_de2, p\_bbu, 'N', SYSDATE);  
 END IF;  
 ```  
  
- Update `azbj\_qc\_kclick\_data` table:  
 ```sql  
 UPDATE azbj\_qc\_kclick\_data  
 SET flg = 'N'  
 WHERE policy\_ref = :control.pol\_ref;  
 ```

# Manage Dispatch Details

Type: DISPATCH

Title: Manage Dispatch Details  
  
Acceptance Criteria:  
1. The system should allow the user to input and save the following dispatch details:  
 - Shipment Number (SHIP\_NO)  
 - Airway Bill Number (AWB\_NO)  
 - Delivery Type (DEL\_TYPE) with default value "I"  
 - Vendor (VENDOR)  
 - Destination Type (DES\_TYPE) with default value "CUST"  
 - Pickup Request Number (PICK\_REQ\_NO)  
 - Pickup Request Reference (PR\_REF) (read-only)  
 - Consignee Name (CONSIGNEE\_NAME) (read-only)  
 - Consignee Address Line 1 (CON\_ADD1) (read-only)  
 - Consignee Address Line 2 (CON\_ADD2) (read-only)  
 - Consignee Address Line 3 (CON\_ADD3) (read-only)  
 - Consignee City (CON\_CITY) (read-only)  
 - Consignee State (CON\_STATE) (read-only)  
 - Destination PIN (DES\_PIN) (read-only)  
 - Weight (WT)  
 - Weight Unit (GM) with default value "Grm"  
2. The system should provide a "Save" button to save the dispatch details.  
3. The system should display the vendor code (VEN\_CODE) as a read-only field.  
4. The system should fetch vendor details from the database using the following query:  
 ```sql  
 SELECT VENDOR\_CODE, VENDOR\_NAME  
 FROM azbj\_logistics\_vendor\_details  
 WHERE branch\_delete\_flag IS NULL  
 AND branch\_code = substr(user, 1, 3)  
 AND branch\_code IS NOT NULL  
 ```  
  
Definition of Done:  
- The user can input and save all required dispatch details.  
- The "Save" button functions correctly and saves the dispatch details.  
- The vendor code is displayed as a read-only field.  
- Vendor details are fetched correctly from the database using the specified query.

# Implement Dispatch Type Dropdown

Type: DISPATCH

Title: Implement Dispatch Type Dropdown  
  
Acceptance Criteria:  
1. The dispatch type field should display a dropdown list with at least two options.  
2. The default value of the dispatch type should be set to "CUST".  
3. The dispatch type field should be positioned correctly on the user interface.  
4. The dispatch type field should not be mandatory.  
  
Definition of Done:  
1. The dispatch type dropdown list is implemented and displays the correct options.  
2. The default value "CUST" is set for the dispatch type field.  
3. The dispatch type field is correctly positioned on the user interface.  
4. The dispatch type field is not marked as required.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include any specific database queries.

# User can select a delivery type from a predefined list

Type: DISPATCH

Title: User can select a delivery type from a predefined list  
  
Acceptance Criteria:  
1. The delivery type field should display a list of options for the user to select from.  
2. The list should contain at least two predefined delivery types.  
3. The default value for the delivery type should be set to "I".  
4. The delivery type field should be clearly visible and positioned appropriately on the dispatch form.  
5. The field should have a white background, black text, and use the Tahoma font with a medium weight and normal spacing.  
  
Definition of Done:  
1. The delivery type field is implemented and displays a list of predefined options.  
2. The default value is set to "I".  
3. The field is styled with the specified font and colors.  
4. The field is positioned correctly on the dispatch form.  
5. The functionality is tested and verified to meet the acceptance criteria.

# Implement GM Field with Predefined List

Type: DISPATCH

Title: Implement GM Field with Predefined List  
  
Acceptance Criteria:  
1. The GM field should display a list of three predefined values.  
2. The default value for the GM field should be "Grm".  
3. The GM field should be positioned at coordinates (497, 815) on the screen.  
4. The GM field should have a width of 39 and a height of 14.  
5. The GM field should have a maximum length of 10 characters.  
6. The GM field should use the "Tahoma" font with a size of 8, medium weight, and plain style.  
7. The GM field should have a white background and black foreground.  
  
Definition of Done:  
- The GM field is implemented with the specified list of values.  
- The default value "Grm" is set correctly.  
- The field is positioned and sized as specified.  
- The field's font and color properties are applied correctly.  
- The field is tested to ensure it meets all acceptance criteria.

# Vendor Selection via Double-Click

Type: DISPATCH

Title: Vendor Selection via Double-Click  
  
Acceptance Criteria:  
1. When the user double-clicks on the vendor field, a list of vendors should be displayed.  
2. The list of vendors should be fetched from the database table `azbj\_logistics\_vendor\_details`.  
3. The list should only include vendors where `branch\_delete\_flag` is NULL, `branch\_code` is not NULL, and the `branch\_code` matches the first three characters of the current user's branch code.  
  
Definition of Done:  
1. The vendor field should be interactive and respond to double-clicks.  
2. The list of vendors should be displayed correctly and should meet the specified criteria.  
3. The user should be able to select a vendor from the list, and the selected vendor's information should be populated in the vendor field.  
  
DB queries for Table reference CRUD operations only(With Usage):  
```sql  
SELECT VENDOR\_CODE, VENDOR\_NAME  
FROM azbj\_logistics\_vendor\_details  
WHERE branch\_delete\_flag IS NULL  
AND branch\_code = substr(user, 1, 3)  
AND branch\_code IS NOT NULL;  
```

# Validate and Save Shipment Details

Type: DISPATCH

Title: Validate and Save Shipment Details  
  
Acceptance Criteria:  
1. The system should validate that the shipment number, AWB number, vendor, and weight fields are not null.  
2. If any of these fields are null, the system should prompt the user with an appropriate error message and focus on the respective field.  
3. If the shipment number does not exist in the shipment details table, the system should:  
 - Insert a new record into the shipments table with the provided details.  
 - Insert a new record into the shipment details table with the provided details.  
 - Insert a new record into the document details table with the provided details.  
 - Update the QC data table to set the flag to 'N' for the given policy reference.  
 - Commit the transaction.  
 - Display a success message indicating that the shipment has been dispatched successfully.  
 - Clear the form.  
4. If the shipment number already exists in the shipment details table, the system should:  
 - Update the QC data table to set the flag to 'N' for the given policy reference.  
 - Commit the transaction.  
 - Clear the form.  
 - Display an error message indicating that the shipment number has already been dispatched.  
  
Definition of Done:  
- The "Save" button functionality is implemented as per the acceptance criteria.  
- All validations and database operations are performed correctly.  
- Appropriate messages are displayed to the user based on the validation and operation results.  
- The form is cleared after successful or failed operations.  
- The feature is tested and verified to ensure it works as expected.  
  
DB queries for Table reference CRUD operations only(With Usage):  
```sql  
-- Check if shipment number exists  
SELECT COUNT() INTO cnt FROM azbj\_logistics\_shipment\_detail WHERE shipment\_id = :dispatch.ship\_no;  
  
-- Insert into shipments table  
INSERT INTO azbj\_logistics\_shipments  
 (shipment\_id, policy\_no, consignee\_name, consignee\_address, destination\_city, pincode, shipment\_through, pickup\_request\_no, shipment\_creation\_date, shipment\_create\_user, shipment\_status, consignee\_address1, consignee\_address2, consignee\_address5, consignee\_phone, consignee\_mobile)  
VALUES  
 (:dispatch.ship\_no, SUBSTR(:dispatch.ship\_no, 4, 10), :dispatch.consignee\_name, :dispatch.con\_add1, :dispatch.con\_city, :dispatch.des\_pin, CASE WHEN :dispatch.pick\_req\_no IS NULL THEN NULL WHEN :dispatch.pick\_req\_no IS NOT NULL THEN 'THROUGH REQUEST' END, :dispatch.pick\_req\_no, SYSDATE, user, 'CLOSE', :dispatch.con\_add2, :dispatch.con\_add3, :dispatch.con\_state, v\_tel, v\_mobile);  
  
-- Insert into shipment details table  
INSERT INTO azbj\_logistics\_shipment\_detail  
 (shipment\_id, policy\_no, consignee\_name, consignee\_address, destination\_city, pincode, shipment\_through, pickup\_request\_no, shipment\_create\_date, shipment\_create\_user, shipment\_status, shipment\_group\_status, pickup\_status, weight, declared\_value, awb\_no, pre\_alert\_instruction, vendor\_id, courier\_no, shipment\_send\_from, shipment\_send\_to, shipment\_send\_user, send\_date, consignee\_address1, consignee\_address2, consignee\_address5, consignee\_phone, consignee\_mobile)  
VALUES  
 (:dispatch.ship\_no, SUBSTR(:dispatch.ship\_no, 4, 10), :dispatch.consignee\_name, :dispatch.con\_add1, :dispatch.con\_city, :dispatch.des\_pin, CASE WHEN :dispatch.pick\_req\_no IS NULL THEN NULL WHEN :dispatch.pick\_req\_no IS NOT NULL THEN 'THROUGH REQUEST' END, :dispatch.pick\_req\_no, SYSDATE, user, 'INDIVIDUAL', 'INDIVIDUAL', 'CLOSE', :dispatch.wt, '', :dispatch.awb\_no, '', :dispatch.ven\_code, :dispatch.awb\_no, '', :dispatch.con\_city, user, SYSDATE, :dispatch.con\_add2, :dispatch.con\_add3, :dispatch.con\_state, v\_tel, v\_mobile);  
  
-- Insert into document details table  
INSERT INTO azbj\_logistics\_document\_detail  
 VALUES (:dispatch.ship\_no, '', :dispatch.awb\_no, TO\_NUMBER(SUBSTR(:dispatch.ship\_no, 1, 3)), 1, :dispatch.wt, :dispatch.wt);  
  
-- Update QC data table  
UPDATE AZBJ\_QC\_KCLICK\_DATA SET flg = 'N' WHERE POLICY\_REF = :GLOBAL.POLICY\_REF;  
  
-- Commit the transaction  
COMMIT;  
```

# View and Manage Underwriting Comments

Type: UW

Title: View and Manage Underwriting Comments  
  
Acceptance Criteria:  
1. The system should fetch the current user's profile and determine their permissions.  
2. If the user does not have a specific prefix in their username, the user ID field should be hidden.  
3. The system should display comments from the `azbj\_uw\_comments` table based on the user's profile:  
 - If the user has a specific prefix or is a supervisor, all comments for the policy should be displayed.  
 - If the user does not have the specific prefix and is not a supervisor, only comments with certain flags should be displayed.  
4. The system should also display comments from the `azbj\_ri\_co\_details` table for the policy.  
5. If no comments are found, an alert should be shown indicating that there are no UW comments for the policy number.  
6. The user should be redirected to the QC block if they acknowledge the alert.  
  
Definition of Done:  
- The user can view comments based on their profile and permissions.  
- The user ID field is hidden for users without the specific prefix.  
- Comments are fetched and displayed from both `azbj\_uw\_comments` and `azbj\_ri\_co\_details` tables.  
- An alert is shown if no comments are found, and the user is redirected to the QC block upon acknowledgment.  
  
DB queries for Table reference CRUD operations only(With Usage):  
```sql  
-- Fetch current user  
SELECT USER INTO :control.users FROM DUAL;  
  
-- Fetch user profile  
SELECT user\_profile INTO v\_uw\_profile FROM azbj\_user\_prof\_ref WHERE user\_name = :control.users;  
  
-- Fetch comments for users with specific prefix or supervisor profile  
SELECT FROM azbj\_uw\_comments WHERE policy\_no = :control.pol\_ref;  
  
-- Fetch comments for other users based on flags  
SELECT FROM azbj\_uw\_comments WHERE policy\_no = :control.pol\_ref AND 0 = (CASE WHEN :control.users LIKE 'UU%' AND v\_uw\_profile <> 'SUPERVISOR' THEN (CASE WHEN (user\_id LIKE 'P00%' AND NVL(flag, 'N') = 'Y') THEN 1 ELSE 0 END) WHEN NVL(flag, 'N') = 'N' THEN 0 ELSE 1 END);  
  
-- Fetch comments from azbj\_ri\_co\_details  
SELECT user\_id, co\_date, uw\_comments FROM azbj\_ri\_co\_details WHERE policy\_ref = :control.pol\_ref ORDER BY co\_date;  
```

# Manage Nominee Information in QC Module

Type: BLK\_NOMINEE

Detailed description: As a user, I want to manage nominee information within the QC module, so that I can efficiently handle nominee-related data.  
  
Acceptance criteria:  
1. The system should allow the user to input and update nominee information, including:  
 - MIS User: A text field with a maximum length of 50 characters.  
 - Scan Date: A date field.  
 - Flag: A numeric field.  
 - Insert Date: A date field.  
 - Flag1: A text field.  
 - Description: A display-only text field with a maximum length of 200 characters.  
 - OPUS Value: A read-only text field with a maximum length of 5000 characters.  
 - CI Nominee: A radio group with an initial value of 'N'.  
  
2. The system should display the nominee information in a structured format, with specific positions and styles for each field.  
  
Definition of Done:  
- The user can successfully input and update nominee information.  
- The nominee information is displayed correctly according to the specified positions and styles.  
- The read-only fields (Description and OPUS Value) are not editable by the user.  
- The radio group for CI Nominee initializes with the value 'N'.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include specific SQL queries or table references.

# Implement Radio Group for Nominee Selection in QC Section

Type: BLK\_NOMINEE

Title: Implement Radio Group for Nominee Selection in QC Section  
  
Acceptance Criteria:  
- The radio group should be initialized with a default value of "N".  
- The radio group should be positioned at coordinates (339, 134) on the form.  
- The radio group should have a width of 68 units and a height of 14 units.  
- The background color of the radio group should be gray, and the foreground color should be black.  
  
Definition of Done:  
- The radio group is visible in the QC section of the form.  
- The radio group is initialized with the default value "N".  
- The radio group is correctly positioned and sized according to the specified dimensions.  
- The radio group has the specified background and foreground colors.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Not applicable as the provided XML content does not include any database-specific queries.

# Manage QC Process for Policies

Type: CONTROL

Title: Manage QC Process for Policies  
  
Acceptance Criteria:  
1. The system should allow users to view and update policy details.  
2. The system should allow users to submit QC details.  
3. The system should display relevant user comments and policy information.  
4. The system should handle different user profiles and permissions.  
5. The system should provide alerts if there are no comments for a policy.  
6. The system should update the status and substatus of policies in the database.  
7. The system should track changes and log relevant information.  
  
Definition of Done:  
1. Users can view and update policy details.  
2. Users can submit QC details.  
3. User comments and policy information are displayed correctly.  
4. User profiles and permissions are handled appropriately.  
5. Alerts are shown if there are no comments for a policy.  
6. Policy status and substatus are updated in the database.  
7. Changes are tracked and logged.  
  
DB queries for Table reference CRUD operations only(With Usage):  
```sql  
-- Update the status and substatus of a policy  
UPDATE azbj\_phub\_qc\_status\_detail  
SET status = st,  
 substatus = subst,  
 ins\_date = SYSDATE  
WHERE policy\_no = :control.pol\_ref;  
  
-- Insert a new record if the policy is not found  
INSERT INTO azbj\_phub\_qc\_status\_detail(POLICY\_NO, CONTRACT\_ID, STATUS, SUBSTATUS, DE1, DE2, BBU, LOCK\_FLG, INS\_DATE)  
VALUES (:control.pol\_ref, p\_con\_id, st, subst, p\_de1, p\_de2, p\_bbu, 'N', SYSDATE);  
  
-- Select application number and receipt number  
SELECT application\_no, perm\_receipt\_no  
INTO app1\_no, rcpt1\_no  
FROM azbj\_phub\_tracker  
WHERE proposal\_no = :control.pol\_ref;  
  
-- Update QC click data flag  
UPDATE azbj\_qc\_kclick\_data  
SET flg = 'N'  
WHERE policy\_ref = :control.pol\_ref;  
```

# Checkbox Functionality for Navigating and Updating Records

Type: CONTROL

Title: Checkbox Functionality for Navigating and Updating Records  
  
Acceptance Criteria:  
1. When the checkbox is checked, the system should navigate to the "QC" section.  
2. The system should iterate through all records in the "QC" section.  
3. For each record, if a specific field (ci) is marked as 'Y' and another field (flg) has a value of 10, the system should update the "ci" field to 'N'.  
4. The iteration should stop when the last record is reached.  
  
Definition of Done:  
- The checkbox functionality is implemented and tested.  
- The system correctly navigates to the "QC" section upon checking the checkbox.  
- The iteration through records and the conditional update of fields are functioning as expected.  
- All acceptance criteria are met and verified through testing.

# Dropdown for Reason Of Skip in Quality Control Module

Type: CONTROL

Title: Dropdown for Reason Of Skip in Quality Control Module  
  
Acceptance Criteria:  
1. The user should see a dropdown list labeled "Reason Of Skip" when interacting with the quality control module.  
2. The dropdown list should contain three predefined options for the reason of skipping.  
3. The selected reason should be stored in the database for future reference and reporting.  
4. The dropdown list should be positioned at the top left of the quality control section and should be easily accessible.  
  
Definition of Done:  
1. The dropdown list for "Reason Of Skip" is implemented and visible in the quality control module.  
2. The list contains exactly three predefined options.  
3. The selected option is saved to the database correctly.  
4. The feature is tested and verified to work as expected without any errors.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include specific SQL queries or database operations.

# Checkbox Functionality for Navigating and Updating Records in QC Block

Type: CONTROL

Detailed description: As a user, I want to ensure that when a specific checkbox is checked, the system will navigate to a particular data block, iterate through its records, and update certain fields based on specific conditions.  
  
Acceptance criteria:  
1. When the checkbox is checked, the system should navigate to the "QC" data block.  
2. The system should iterate through all records in the "QC" data block.  
3. For each record, if the field "ci" is set to 'Y' and the field "flg" is equal to 10, the system should update the "ci" field to 'N'.  
4. The iteration should stop when the last record in the "QC" data block is reached.  
  
Definition of Done:  
- The checkbox functionality is implemented and tested.  
- The system correctly navigates to the "QC" data block upon checking the checkbox.  
- The system iterates through all records in the "QC" data block and updates the "ci" field based on the specified conditions.  
- All acceptance criteria are met and verified through testing.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided logic is specific to Oracle Forms and cannot be executed directly in the database without modification.

# Exit Button Functionality

Type: CONTROL

Detailed description: As a user, I want to ensure that when I press the "Exit" button, the system updates the status of specific records in multiple tables and exits the form without validation, so that the data integrity is maintained and the form is closed properly.  
  
Acceptance criteria:  
1. When the "Exit" button is pressed, the system should update the `AZBJ\_QC\_KCLICK\_DATA` table to set the `flg` field to 'N' for the record where `POLICY\_REF` matches the current policy reference.  
2. The system should update the `azbj\_kclick\_qc\_details` table to set the `qc\_status` field to 'N' for records where `policy\_ref` matches the current policy reference and `qc\_status` is 'P'.  
3. The system should update the `azbj\_merging\_tat\_detail` table to set the `qc\_status` field to 'N' for records where `policy\_ref` matches the current policy reference and `qc\_status` is 'P'.  
4. The system should commit the changes to the database.  
5. The form should exit without performing any validation.  
  
Definition of Done:  
- The "Exit" button functionality is implemented as described.  
- The specified tables are updated correctly when the button is pressed.  
- The form exits without validation after the updates are committed.  
- The functionality is tested and verified to ensure data integrity and proper form closure.  
  
DB queries for Table reference CRUD operations only(With Usage):  
```sql  
UPDATE AZBJ\_QC\_KCLICK\_DATA  
SET flg = 'N'  
WHERE POLICY\_REF = :CONTROL.POL\_REF;  
  
UPDATE azbj\_kclick\_qc\_details  
SET qc\_status = 'N'  
WHERE policy\_ref = :CONTROL.POL\_REF AND qc\_status = 'P';  
  
UPDATE azbj\_merging\_tat\_detail  
SET qc\_status = 'N'  
WHERE policy\_ref = :CONTROL.POL\_REF AND qc\_status = 'P';  
  
COMMIT;  
```

# Update QC Status and Clear Form on Search Button Press

Type: CONTROL

Title: Update QC Status and Clear Form on Search Button Press  
  
Acceptance Criteria:  
1. When the search button is pressed, the system should update the `qc\_status` to 'N' for records in the `azbj\_kclick\_qc\_details` table where the `policy\_ref` matches the current policy reference and the `qc\_status` is 'P'.  
2. The system should also update the `qc\_status` to 'N' for records in the `azbj\_merging\_tat\_detail` table where the `policy\_ref` matches the current policy reference and the `qc\_status` is 'P'.  
3. After updating the records, the system should commit the changes to the database.  
4. The form should be cleared without validation after the updates are committed.  
  
Definition of Done:  
- The search button functionality is implemented and tested.  
- The `qc\_status` is correctly updated in both tables as per the acceptance criteria.  
- The form is cleared and ready for new input after the updates.  
- All changes are committed to the database successfully.  
  
DB queries for Table reference CRUD operations only(With Usage):  
```sql  
-- Update qc\_status in azbj\_kclick\_qc\_details  
UPDATE azbj\_kclick\_qc\_details  
SET qc\_status = 'N'  
WHERE policy\_ref = :CONTROL.POL\_REF AND qc\_status = 'P';  
  
-- Update qc\_status in azbj\_merging\_tat\_detail  
UPDATE azbj\_merging\_tat\_detail  
SET qc\_status = 'N'  
WHERE policy\_ref = :CONTROL.POL\_REF AND qc\_status = 'P';  
  
-- Commit the changes  
COMMIT;  
```

# Reset Button Functionality

Type: CONTROL

Title: Reset Button Functionality  
  
Acceptance Criteria:  
1. When the "Reset" button is pressed, the system should update the `AZBJ\_QC\_KCLICK\_DATA` table to set the `flg` field to 'N' for the record matching the current policy reference.  
2. The system should update the `azbj\_kclick\_qc\_details` table to set the `qc\_status` field to 'N' for records matching the current policy reference and having a `qc\_status` of 'P'.  
3. The system should update the `azbj\_merging\_tat\_detail` table to set the `qc\_status` field to 'N' for records matching the current policy reference and having a `qc\_status` of 'P'.  
4. After performing the updates, the system should commit the changes to the database.  
5. The form should be cleared without validation, and the 'control.sub' item should be enabled.  
  
Definition of Done:  
- The "Reset" button functionality is implemented and tested.  
- The specified tables are updated correctly based on the current policy reference.  
- The form is cleared and the 'control.sub' item is enabled after the updates.  
- All acceptance criteria are met and verified through testing.  
  
DB queries for Table reference CRUD operations only(With Usage):  
```sql  
-- Update AZBJ\_QC\_KCLICK\_DATA table  
UPDATE AZBJ\_QC\_KCLICK\_DATA  
SET flg = 'N'  
WHERE POLICY\_REF = :CONTROL.POL\_REF;  
  
-- Update azbj\_kclick\_qc\_details table  
UPDATE azbj\_kclick\_qc\_details  
SET qc\_status = 'N'  
WHERE policy\_ref = :CONTROL.POL\_REF AND qc\_status = 'P';  
  
-- Update azbj\_merging\_tat\_detail table  
UPDATE azbj\_merging\_tat\_detail  
SET qc\_status = 'N'  
WHERE policy\_ref = :CONTROL.POL\_REF AND qc\_status = 'P';  
  
-- Commit the changes  
COMMIT;  
```

# Update Nominee Details

Type: CONTROL

Title: Update Nominee Details  
  
Acceptance Criteria:  
1. The system should check if the existing nominee details are present.  
2. If nominee details are missing, the system should display an error message: "Nominee Detail Not Found".  
3. The system should retrieve the contract ID associated with the policy reference.  
4. The system should prompt the user with a confirmation message: "Sure, want to update Nominee Name?".  
5. If the user confirms the update:  
 - The system should check if the new nominee name is different from the old nominee name.  
 - If different, the system should update the nominee name in the beneficiary repository.  
 - The system should log the update action in the policy activity log.  
 - The system should display a success message: "Nominee Name updated successfully".  
6. If the user cancels the update, the system should not make any changes.  
  
Definition of Done:  
- The nominee update functionality is implemented and tested.  
- The system correctly validates and prompts for confirmation before updating the nominee details.  
- The system logs the update action in the policy activity log.  
- Appropriate error and success messages are displayed to the user.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Update the nominee name in the beneficiary repository:  
 ```sql  
 UPDATE azbj\_beneficiary\_rep  
 SET ben\_NAME = :control.Nominee\_NAME  
 WHERE contract\_id = con\_id  
 AND top\_indicator = 'Y'  
 AND action\_code <> 'D'  
 AND BEN\_RELATION <> 'SPOUSE'  
 AND rownum < 2;  
 ```  
  
- Insert the update action into the policy activity log:  
 ```sql  
 INSERT INTO azbj\_pol\_activity\_log  
 VALUES (azbj\_policy\_log\_seq.NEXTVAL,   
 sysdate,  
 pme\_api.opus\_date,   
 user,   
 con\_id,   
 '',   
 '', :control.pol\_ref,   
 'NOM\_UPD',   
 'Nominee Name Updation in QC' || :control.Nominee\_NAME,   
 'INSERTED THROUGH QC FOR NOMINEE UPDATION' || :CONTROL.OLD\_NOMINEE,   
 NULL, pme\_api.opus\_date);  
 ```

# Enrich Proposal Details on Button Press

Type: CONTROL

Detailed description: As a user, I want to be able to enrich the details of a proposal by pressing a button, so that I can view and manage the associated documents and information efficiently.  
  
Acceptance criteria:  
1. When the button is pressed, the system should check if there is a scrutiny number associated with the given proposal number.  
2. If a scrutiny number is found, the system should verify if there are any enrichment requests for that scrutiny number.  
3. If enrichment requests exist, the system should generate a URL for the document management system and open the document in a web browser.  
4. If no enrichment requests are found, the system should display an error message indicating that no image has been uploaded.  
5. If no scrutiny number is found, the system should display an error message indicating that no scrutiny details are available.  
6. The system should also check if enrichment details are available for the given application number.  
7. If enrichment details are not available, an error message should be displayed.  
8. The system should retrieve the customer name associated with the contract ID.  
9. If the customer name is not found in the primary table, the system should attempt to retrieve it from a secondary table.  
10. The system should create a parameter list with the application number, form name, proposal number, and customer name, and then call another form to display the enrichment details.  
  
Definition of Done:  
- The button functionality is implemented and tested.  
- The system correctly checks for scrutiny numbers and enrichment requests.  
- Appropriate error messages are displayed when necessary.  
- The document management system URL is generated and opened correctly.  
- The customer name is retrieved from the appropriate table.  
- The parameter list is created and the form is called successfully.  
- All acceptance criteria are met and verified through testing.  
  
DB queries for Table reference CRUD operations only(With Usage):  
1. To check for scrutiny number:  
 ```sql  
 SELECT a.scrutiny\_no   
 FROM azbj\_phub\_scrutiny\_prop a, azbj\_phub\_tracker b  
 WHERE a.application\_no = b.application\_no  
 AND b.proposal\_no = :control.pol\_ref;  
 ```  
  
2. To count enrichment requests:  
 ```sql  
 SELECT COUNT()   
 FROM azbj\_phub\_scrutiny\_req  
 WHERE scrutiny\_no = :scrutiny\_no  
 AND req\_code = 'ENRICH';  
 ```  
  
3. To check for enrichment details:  
 ```sql  
 SELECT COUNT (ROWNUM)   
 FROM azbj\_data\_entrich\_details  
 WHERE application\_no = :application\_no;  
 ```  
  
4. To retrieve customer name from primary table:  
 ```sql  
 SELECT customer\_name\_text   
 FROM ocp\_interested\_parties  
 WHERE contract\_id = :contract\_id AND ip\_no = 1;  
 ```  
  
5. To retrieve customer name from secondary table:  
 ```sql  
 SELECT customer\_name\_text   
 FROM wip\_interested\_parties  
 WHERE contract\_id = :contract\_id AND ip\_no = 1;  
 ```

# Update QC Records on Checkbox Change

Type: CONTROL

Title: Update QC Records on Checkbox Change  
  
Acceptance Criteria:  
1. When the checkbox is checked, the system should navigate to the QC module.  
2. The system should iterate through all records in the QC module.  
3. For each record, if the 'ci' field is 'Y' and the 'flg' field is 10, the 'ci' field should be updated to 'N'.  
4. The iteration should stop when the last record is reached.  
  
Definition of Done:  
- The checkbox functionality is implemented and tested.  
- The system correctly navigates to the QC module and iterates through all records.  
- The 'ci' field is updated to 'N' for records meeting the specified conditions.  
- The iteration stops at the last record.  
- All changes are saved and verified through testing.

# Checkbox Functionality for Navigating and Updating Records in QC Section

Type: CONTROL

Title: Checkbox Functionality for Navigating and Updating Records in QC Section  
  
Acceptance Criteria:  
1. When the checkbox is checked, the system should navigate to the "QC" section.  
2. The system should iterate through all records in the "QC" section.  
3. For each record, if a specific field (ci) is marked as 'Y' and another field (flg) has a value of 10, the system should update the field (ci) to 'N'.  
4. The iteration should stop when the last record is reached.  
  
Definition of Done:  
- The checkbox functionality is implemented and tested.  
- The system correctly navigates to the "QC" section upon checking the checkbox.  
- The iteration through records and the conditional update of fields are functioning as expected.  
- All acceptance criteria are met and verified through testing.

# Checkbox Functionality for Navigating and Updating Records in QC Section

Type: CONTROL

Title: Checkbox Functionality for Navigating and Updating Records in QC Section  
  
Acceptance Criteria:  
1. When the checkbox is checked, the system should navigate to the "QC" section.  
2. The system should iterate through all records in the "QC" section.  
3. For each record, if a specific field (ci) is marked as 'Y' and another field (flg) has a value of 11, the system should update the field (ci) to 'N'.  
4. The iteration should stop when the last record is reached.  
  
Definition of Done:  
- The checkbox functionality is implemented and tested.  
- The system correctly navigates to the "QC" section upon checking the checkbox.  
- The system iterates through all records and updates the fields as per the specified conditions.  
- All acceptance criteria are met and verified through testing.

# View Images Button Functionality

Type: CONTROL

Title: View Images Button Functionality  
  
Acceptance Criteria:  
1. When the button is pressed, the system should:  
 - Retrieve the solution ID based on the application number or contract ID.  
 - Retrieve the parent contract ID if the solution ID is found and the parent flag is set to 'Y'.  
 - If a parent contract ID is found, get the policy reference number associated with it.  
 - Log the details of the solution ID, parent contract ID, and policy number.  
 - Generate a URL using the policy number or the policy reference number.  
 - Open the generated URL in a web browser.  
 - Display an error message if the URL cannot be generated.  
  
Definition of Done:  
- The button should be functional and perform all the steps outlined in the acceptance criteria.  
- The URL should open in a web browser if generated successfully.  
- An error message should be displayed if there is an issue with generating the URL.  
- The functionality should be tested and verified to ensure it meets the requirements.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Retrieve solution ID:  
 ```sql  
 SELECT solution\_id  
 INTO v\_sol\_id  
 FROM azbj\_solution\_appln\_bases  
 WHERE application\_no = PK\_VARS.v\_appln\_no OR contract\_id = azbj\_pk0\_acc.get\_contract\_id(:control.pol\_ref) AND solution\_id IS NOT NULL  
 AND ROWNUM = 1;  
 ```  
  
- Retrieve parent contract ID:  
 ```sql  
 SELECT contract\_id  
 INTO v\_parent\_cont\_id  
 FROM azbj\_solution\_appln\_bases a  
 WHERE solution\_id = v\_sol\_id AND PARENT\_FLAG = 'Y' AND EXISTS  
 (SELECT 1  
 FROM azbj\_solution\_appln\_bases b  
 WHERE a.solution\_id = b.solution\_id AND PARENT\_FLAG IS NULL);  
 ```  
  
- Log details:  
 ```sql  
 AZBJ\_NEW\_BBU\_UTILITIES.BBU\_INS\_LOG(:control.pol\_ref, :control.pol\_ref, 'PrePrintQC Log=>sol\_id: ' || v\_sol\_id || ' parent\_cont\_id: ' || v\_parent\_cont\_id || ' pol\_no: ' || v\_pol\_no);  
 ```  
  
- Generate URL:  
 ```sql  
 v\_url := azbj\_encrypt\_dms\_link('NB', 'PRINT\_QC', NVL(v\_pol\_no, :control.pol\_ref), NULL, NULL, NULL, NULL);  
 ```  
  
- Open URL:  
 ```sql  
 IF v\_url IS NOT NULL THEN  
 web.show\_document(v\_url);  
 END IF;  
 ```  
  
- Error handling:  
 ```sql  
 EXCEPTION  
 WHEN OTHERS THEN  
 azbj\_message('W', 'Please check the URL' || SQLERRM);  
 ```

# Checkbox Functionality for QC Section

Type: CONTROL

Title: Checkbox Functionality for QC Section  
  
Acceptance Criteria:  
1. When the checkbox is checked, the system should navigate to the "QC" section.  
2. The system should iterate through all records in the "QC" section.  
3. For each record, if a specific field (`ci`) is marked as 'Y' and another field (`flg`) has a value of 11, the system should update the `ci` field to 'N'.  
4. The iteration should stop when the last record is reached.  
  
Definition of Done:  
- The checkbox functionality is implemented and tested.  
- The system correctly navigates to the "QC" section upon checking the checkbox.  
- The system iterates through all records and updates the `ci` field based on the specified conditions.  
- All acceptance criteria are met and verified through testing.

# Update Dispatch Details Based on Checkbox Status

Type: CONTROL

Title: Update Dispatch Details Based on Checkbox Status  
  
Acceptance Criteria:  
1. When the checkbox is checked:  
 - Retrieve the contract ID using a reference value.  
 - Fetch and populate the dispatch address details from the `cp\_addresses` and `OCP\_INTERESTED\_PARTIES` tables.  
 - Fetch and populate the consignee name from the `CP\_PARTNERS` and `OCP\_INTERESTED\_PARTIES` tables. If not found, fetch from the `WIP\_INTERESTED\_PARTIES` table.  
 - Set the dispatch shipment number, reference, weight, delivery type, destination type, and GM value.  
 - Navigate to the dispatch shipment number field.  
  
2. When the checkbox is unchecked:  
 - Navigate to the dispatch block.  
 - Clear the current record.  
 - Reset the delivery type, destination type, and GM value to empty.  
  
Definition of Done:  
- The dispatch details are correctly populated when the checkbox is checked.  
- The dispatch details are cleared when the checkbox is unchecked.  
- Error handling is in place to manage any exceptions during the data retrieval process.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Fetch dispatch address details:  
 ```sql  
 SELECT address\_line1, address\_line2, address\_line3, postcode, address\_line4, address\_line5  
 INTO :dispatch.con\_add1, :dispatch.con\_add2, :dispatch.con\_add3, :DISPATCH.DES\_PIN, :DISPATCH.CON\_CITY, :DISPATCH.CON\_STATE  
 FROM cp\_addresses A, OCP\_INTERESTED\_PARTIES B  
 WHERE A.add\_id = B.MAILING\_ADDRESS\_ID  
 AND ACTION\_CODE <> 'D'  
 AND TOP\_INDICATOR = 'Y'  
 AND IP\_NO = 2  
 AND CONTRACT\_ID = :contract\_id1;  
 ```  
  
- Fetch consignee name:  
 ```sql  
 SELECT CASE WHEN NVL(PARTNER\_TYPE, 'A') != 'I' THEN  
 BEFORE\_TITLE || ' ' || FIRST\_NAME || ' ' || MIDDLE\_NAME || ' ' || SURNAME  
 ELSE  
 INSTITUTION\_NAME END  
 INTO :dispatch.CONSIGNEE\_NAME  
 FROM CP\_PARTNERS a, OCP\_INTERESTED\_PARTIES B  
 WHERE a.PART\_ID = b.partner\_id  
 AND b.contract\_id = :contract\_id1  
 AND ACTION\_CODE <> 'D'  
 AND TOP\_INDICATOR = 'Y'  
 AND IP\_NO = 2;  
 ```  
  
- Fetch consignee name fallback:  
 ```sql  
 SELECT CUSTOMER\_NAME\_TEXT  
 INTO :dispatch.CONSIGNEE\_NAME  
 FROM WIP\_INTERESTED\_PARTIES  
 WHERE CONTRACT\_ID = :contract\_id1  
 AND IP\_NO = 2;  
 ```

# Checkbox Functionality for Navigating and Updating Records in QC Section

Type: CONTROL

Detailed description: As a user, I want to ensure that when a specific checkbox is checked, the system will navigate to a particular section, iterate through records, and update certain fields based on specific conditions.  
  
Acceptance criteria:  
1. When the checkbox is checked, the system should navigate to the "QC" section.  
2. The system should iterate through all records in the "QC" section.  
3. For each record, if a specific field (ci) is marked as 'Y' and another field (flg) has a value of 11, the system should update the field (ci) to 'N'.  
4. The iteration should stop when the last record is reached.  
  
Definition of Done:  
- The checkbox functionality is implemented and tested.  
- The system correctly navigates to the "QC" section upon checking the checkbox.  
- The system iterates through all records and updates the fields as per the specified conditions.  
- All acceptance criteria are met and verified through testing.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided logic is specific to Oracle Forms and cannot be executed directly in the database without modification.

# Automatic Update of QC Records on Checkbox Selection

Type: CONTROL

Title: Automatic Update of QC Records on Checkbox Selection  
  
Acceptance Criteria:  
1. When the checkbox (CH3) is checked:  
 - The system should navigate to the QC module.  
 - The system should iterate through all records in the QC module.  
 - For each record, if the 'ci' field is 'Y' and the 'flg' field is 10, the system should update the 'ci' field to 'N'.  
 - The iteration should stop when the last record is reached.  
  
Definition of Done:  
- The system correctly navigates to the QC module when the checkbox is checked.  
- The system iterates through all records in the QC module.  
- The 'ci' field is updated to 'N' for records where 'ci' is 'Y' and 'flg' is 10.  
- The iteration stops at the last record.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Not applicable as the provided logic is specific to Oracle Forms and cannot be executed directly in the database without modification.

# Policy Reference Validation and Processing

Type: CONTROL

Title: Policy Reference Validation and Processing  
  
Acceptance Criteria:  
1. The system should check the length of the policy reference number.  
2. If the policy reference number is longer than 10 characters and starts with '101', it should be truncated to the last 10 characters.  
3. The system should verify if the policy reference number is already locked.  
 - If locked, an alert should be displayed indicating that the policy is already locked.  
 - If not locked, the system should update the status to 'Y' and record the current timestamp.  
4. The system should search for and retrieve beneficiary details based on the policy reference number.  
 - If the primary source does not have the details, it should check an alternative source.  
 - If no details are found, the beneficiary fields should be set to NULL.  
5. The system should retrieve the application number associated with the policy reference number.  
6. The system should retrieve the appointee gender based on the application number.  
  
Definition of Done:  
- The policy reference number is validated and processed according to the specified criteria.  
- Relevant alerts are displayed when necessary.  
- Beneficiary details are retrieved and updated correctly.  
- Application number and appointee gender are retrieved and displayed accurately.  
- All changes are committed to the database.  
  
DB queries for Table reference CRUD operations only(With Usage):  
```sql  
-- Check the length of the policy reference number  
SELECT LENGTH(:CONTROL.POL\_REF) INTO v\_length FROM dual;  
  
-- Verify if the policy reference number is already locked  
SELECT NVL(FLG, 'N') INTO QC\_status FROM AZBJ\_QC\_KCLICK\_DATA WHERE POLICY\_REF = :CONTROL.POL\_REF;  
  
-- Update the status and timestamp if the policy is not locked  
UPDATE AZBJ\_QC\_KCLICK\_DATA   
SET flg = 'Y', time\_stamp = SYSDATE   
WHERE POLICY\_REF = :CONTROL.POL\_REF;  
  
-- Retrieve beneficiary details from the primary source  
SELECT APP\_DOB, APP\_NAME, APP\_RELATION  
INTO :BLK\_PRE\_PRINT.BEN\_APP\_DOB, :BLK\_PRE\_PRINT.BEN\_NAME, :BLK\_PRE\_PRINT.BEN\_APP\_RELATION  
FROM azbj\_beneficiary\_rep  
WHERE CONTRACT\_ID = azbj\_pk0\_acc.get\_contract\_id(:CONTROL.POL\_REF)  
AND TOP\_INDICATOR = 'Y'  
AND ROWNUM = 1;  
  
-- Retrieve beneficiary details from the alternative source if not found in the primary source  
SELECT APP\_DOB, APP\_NAME, APP\_RELATION  
INTO :BLK\_PRE\_PRINT.BEN\_APP\_DOB, :BLK\_PRE\_PRINT.BEN\_NAME, :BLK\_PRE\_PRINT.BEN\_APP\_RELATION  
FROM wip\_azbj\_beneficiary\_rep  
WHERE CONTRACT\_ID = azbj\_pk0\_acc.get\_contract\_id(:CONTROL.POL\_REF)  
AND TOP\_INDICATOR = 'Y'  
AND ROWNUM = 1;  
  
-- Retrieve the application number associated with the policy reference number  
SELECT b.application\_no  
INTO v\_appln\_no  
FROM azbj\_phub\_scrutiny\_prop a, azbj\_phub\_tracker b  
WHERE a.application\_no = b.application\_no  
AND b.proposal\_no = :control.pol\_ref;  
  
-- Retrieve the appointee gender based on the application number  
SELECT appointee\_gender  
INTO :BLK\_PRE\_PRINT.appointee\_gender  
FROM azbj\_annuity\_prod\_det  
WHERE appln\_no = v\_appln\_no  
AND TOP\_INDICATOR = 'Y';  
```

# Automatic Update of QC Records on Checkbox Check

Type: CONTROL

Title: Automatic Update of QC Records on Checkbox Check  
  
Acceptance Criteria:  
1. When the checkbox is checked:  
 - The system should navigate to the QC module.  
 - The system should iterate through all records in the QC module.  
 - For each record, if the 'ci' field is 'Y' and the 'flg' field is 10, the system should update the 'ci' field to 'N'.  
 - The iteration should stop when the last record is reached.  
  
Definition of Done:  
- The system correctly navigates to the QC module when the checkbox is checked.  
- The system iterates through all records in the QC module.  
- The 'ci' field is updated to 'N' for records where 'ci' is 'Y' and 'flg' is 10.  
- The iteration stops at the last record.  
- The functionality is tested and verified to work as expected.

# Checkbox Functionality for QC Section Navigation and Record Update

Type: CONTROL

Title: Checkbox Functionality for QC Section Navigation and Record Update  
  
Acceptance Criteria:  
1. When the checkbox is checked, the system should navigate to the "QC" section.  
2. The system should iterate through all records in the "QC" section.  
3. For each record, if a specific field (ci) is marked as 'Y' and another field (flg) has a value of 11, the system should update the ci field to 'N'.  
4. The iteration should stop when the last record is reached.  
  
Definition of Done:  
- The checkbox functionality is implemented and tested.  
- The system correctly navigates to the "QC" section upon checking the checkbox.  
- The system iterates through all records and updates the ci field based on the specified conditions.  
- All acceptance criteria are met and verified through testing.

# Checkbox Functionality for QC Section Navigation and Record Update

Type: CONTROL

Title: Checkbox Functionality for QC Section Navigation and Record Update  
  
Acceptance Criteria:  
1. When the checkbox is checked, the system should navigate to the "QC" section.  
2. The system should start from the first record in the "QC" section.  
3. For each record in the "QC" section:  
 - If the "ci" field is marked as 'Y' and the "flg" field has a value of 11, the "ci" field should be updated to 'N'.  
4. The system should continue this process until the last record in the "QC" section is reached.  
  
Definition of Done:  
- The checkbox functionality is implemented and tested.  
- The system correctly navigates to the "QC" section when the checkbox is checked.  
- The system iterates through all records in the "QC" section and updates the "ci" field based on the specified conditions.  
- All acceptance criteria are met and verified through testing.

# Checkbox Functionality for Navigating and Updating Records in QC Section

Type: CONTROL

Title: Checkbox Functionality for Navigating and Updating Records in QC Section  
  
Acceptance Criteria:  
1. When the checkbox is checked, the system should navigate to the "QC" section.  
2. The system should iterate through all records in the "QC" section.  
3. For each record, if the "ci" field is 'Y' and the "flg" field is 11, the "ci" field should be updated to 'N'.  
4. The iteration should stop when the last record is reached.  
  
Definition of Done:  
- The checkbox functionality is implemented and tested.  
- The system correctly navigates to the "QC" section upon checking the checkbox.  
- The records in the "QC" section are updated based on the specified conditions.  
- All acceptance criteria are met and verified through testing.

# Exit Button Functionality

Type: CONTROL

Detailed description: As a user, I want to be able to exit the current screen and navigate to the QC screen when I press the "Exit" button, so that I can review and manage quality control data.  
  
Acceptance criteria:  
1. When the "Exit" button is pressed, the system should navigate to the QC screen.  
2. The system should hide the current screen and make the QC screen visible.  
3. The system should fetch the current user's profile and determine visibility settings based on the user's profile.  
4. If the user does not have the required profile, certain fields should be hidden.  
5. The system should load and display comments related to the policy number from the database.  
6. If no comments are found, an alert should be shown indicating that there are no comments for the given policy number.  
7. The system should handle exceptions and ensure that the user is informed if no data is found.  
  
Definition of Done:  
- The "Exit" button successfully navigates to the QC screen.  
- The current screen is hidden, and the QC screen is visible.  
- User profile is fetched and used to determine field visibility.  
- Comments related to the policy number are loaded and displayed correctly.  
- An alert is shown if no comments are found.  
- All exceptions are handled gracefully, and the user is informed appropriately.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Fetch current user:  
 ```sql  
 SELECT USER INTO :control.users FROM DUAL;  
 ```  
  
- Fetch user profile:  
 ```sql  
 SELECT user\_profile INTO v\_uw\_profile FROM azbj\_user\_prof\_ref WHERE user\_name = :control.users;  
 ```  
  
- Fetch comments for policy number:  
 ```sql  
 SELECT FROM azbj\_uw\_comments WHERE policy\_no = :control.pol\_ref;  
 ```  
  
- Fetch additional comments for policy number:  
 ```sql  
 SELECT user\_id, co\_date, uw\_comments FROM azbj\_ri\_co\_details WHERE policy\_ref = :control.pol\_ref ORDER BY co\_date;  
 ```

# View and Manage Underwriter Comments

Type: CONTROL

Title: View and Manage Underwriter Comments  
  
Acceptance Criteria:  
1. When the user clicks the "UW Comments" button, the system should check if a policy number is entered.  
2. If a policy number is entered, the system should display the underwriter comments section.  
3. If no policy number is entered, the system should prompt the user to enter a policy number.  
4. The system should retrieve and display comments based on the user's profile and policy number.  
5. If the user has a specific profile (e.g., 'SUPERVISOR'), additional comments should be displayed.  
6. The system should handle cases where no comments are found and notify the user accordingly.  
  
Definition of Done:  
- The "UW Comments" button functionality is implemented and tested.  
- The system correctly checks for the presence of a policy number.  
- The underwriter comments section is displayed when a policy number is entered.  
- The system prompts the user to enter a policy number if it is missing.  
- Comments are retrieved and displayed based on the user's profile and policy number.  
- The system handles and notifies the user when no comments are found.  
- All acceptance criteria are met and verified through testing.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Retrieve user profile:  
 ```sql  
 SELECT user\_profile  
 INTO v\_uw\_profile  
 FROM azbj\_user\_prof\_ref  
 WHERE user\_name = :control.users;  
 ```  
  
- Retrieve comments for a policy number:  
 ```sql  
 SELECT   
 INTO v\_comments  
 FROM azbj\_uw\_comments  
 WHERE policy\_no = :control.pol\_ref;  
 ```  
  
- Retrieve additional comments based on user profile:  
 ```sql  
 SELECT user\_id, co\_date, uw\_comments  
 INTO v\_comments  
 FROM azbj\_ri\_co\_details  
 WHERE policy\_ref = :control.pol\_ref  
 ORDER BY co\_date;  
 ```

# Skip Application Process

Type: CONTROL

Title: Skip Application Process  
  
Acceptance Criteria:  
1. If the reason for skipping is not provided, an error message should be displayed: "Please Select Reason For Skipping Application".  
2. If the reason for skipping is "OTHERS" and comments are not provided, an error message should be displayed: "Please Enter comments".  
3. Upon skipping, the system should set a global approval flag to 'N' and call a specific form for error handling.  
4. If the approval flag is set to 'Y' after the form call:  
 - Update the `AZBJ\_PRINT\_QC\_DISCARD` table to set `BLOCK\_FLAG` to 'N' for the current policy reference where `BLOCK\_FLAG` is 'Y'.  
 - Insert a new record into the `AZBJ\_PRINT\_QC\_DISCARD` table with the policy reference, block flag set to 'Y', the reason for skipping, the current date, the current user, and the comments concatenated with the approval flag.  
 - Commit the transaction and exit the form.  
  
Definition of Done:  
- The skip functionality is implemented and tested.  
- Error messages are displayed correctly when required fields are not provided.  
- The global approval flag is set and handled correctly.  
- Database updates and inserts are performed as specified.  
- The form exits correctly after the process is completed.  
  
DB queries for Table reference CRUD operations only(With Usage):  
```sql  
-- Update query to set BLOCK\_FLAG to 'N' for the current policy reference  
UPDATE AZBJ\_PRINT\_QC\_DISCARD  
SET BLOCK\_FLAG = 'N'  
WHERE POLICY\_REF = :CONTROL.POL\_REF  
AND BLOCK\_FLAG = 'Y';  
  
-- Insert query to add a new record with the provided details  
INSERT INTO AZBJ\_PRINT\_QC\_DISCARD  
(POLICY\_REF, BLOCK\_FLAG, BLOCK\_REASON, DELINK\_DATE, DELINK\_USER, BLOCK\_COMMENTS)  
VALUES(:CONTROL.POL\_REF, 'Y', :control.QC\_SKIP\_REASON, sysdate, USER, v\_approval\_yn || ' - ' || :CONTROL.DELINK\_COMMENTS);  
```

# View Document Associated with Policy Reference

Type: CONTROL

As a user, I want to view a document associated with a specific policy reference, so that I can verify the document's details and ensure it is not corrupted.  
  
Acceptance Criteria:  
1. When the user presses the "View Document" button, the system should retrieve the file path of the document associated with the given policy reference from the database.  
2. The system should delete any existing PDF files in the temporary directory on the client machine.  
3. The system should construct the client and server paths for the document based on the policy reference and the retrieved file path.  
4. The system should copy the document from the server to the client machine.  
5. The system should verify the file size of the copied document to ensure it is not corrupted.  
6. If the file size is zero, the system should display an error message indicating that the image is corrupted and cannot be loaded.  
7. The system should open the document using the default web browser.  
8. The system should delete the temporary PDF file from the client machine after it has been opened.  
9. The "View Document" button should be hidden after the process is completed.  
  
Definition of Done:  
- The user can successfully view the document associated with the policy reference.  
- The system handles errors gracefully, displaying appropriate messages when necessary.  
- Temporary files are cleaned up after the document is viewed.  
- The "View Document" button is hidden after the document is opened.  
  
DB queries for Table reference CRUD operations only (With Usage):  
```sql  
SELECT file\_path  
INTO vfilename  
FROM azbj\_kclick\_qc\_details  
WHERE policy\_ref = :policy\_ref  
AND QC\_STATUS = 'P'  
AND file\_path IS NOT NULL;  
```

# Submit Policy Number for QC Verification

Type: CONTROL

Detailed description: As a user, I want to be able to submit a policy number for quality control (QC) verification. When the "Submit" button is pressed, the system should validate the policy number, check for existing QC records, and update the QC status based on the verification results. If the policy number is not found or is invalid, an error message should be displayed. The system should also handle specific conditions such as checking for bot users, handling different QC statuses, and updating various related tables accordingly.  
  
Acceptance criteria:  
1. The system should validate that the policy number is not null.  
2. If the policy number is valid, the system should:  
 - Check if the user is a bot and set the appropriate flag.  
 - Retrieve and count existing QC records for the policy number.  
 - If no QC records exist, fetch user details and insert new QC details.  
 - Loop through QC records to check for errors and update the QC status.  
 - If errors are found, update the QC status to "QC FAIL" and handle specific error conditions.  
 - If no errors are found, update the QC status to "QC PASS".  
3. The system should display appropriate messages and alerts based on the QC status and user actions.  
4. The system should update related tables and commit the changes to the database.  
  
Definition of Done:  
- The "Submit" button functionality is implemented and tested.  
- The system correctly validates the policy number and handles different QC statuses.  
- Appropriate messages and alerts are displayed to the user.  
- Related tables are updated, and changes are committed to the database.  
- The functionality is tested and verified to work as expected.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Insert into `azbj\_phub\_qc\_detail`:  
 ```sql  
 INSERT INTO azbj\_phub\_qc\_detail (policy\_no, contract\_id, desc\_type, opus\_value, correct\_flg, ins\_user, ins\_date, flg, err\_status, flg1, flg2, flg3, flg4, flg5, flg6)  
 VALUES (:control.pol\_ref, con\_id, :qc.DESC, :qc.opusval, ci, USER, SYSDATE, :qc.flg, err, flg6, flg7, flg8, flg9, flg10, flg11);  
 ```  
  
- Update `azbj\_phub\_qc\_detail`:  
 ```sql  
 UPDATE azbj\_phub\_qc\_detail  
 SET correct\_flg = ci, ins\_date = SYSDATE, opus\_value = :qc.opusval, ins\_user = USER  
 WHERE policy\_no = :control.pol\_ref AND desc\_type = :qc.DESC;  
 ```  
  
- Insert into `azbj\_phub\_qc\_status\_detail`:  
 ```sql  
 INSERT INTO azbj\_phub\_qc\_status\_detail (POLICY\_NO, CONTRACT\_ID, STATUS, SUBSTATUS, DE1, DE2, BBU, LOCK\_FLG, INS\_DATE)  
 VALUES (:control.pol\_ref, p\_con\_id, st, subst, p\_de1, p\_de2, p\_bbu, 'N', SYSDATE);  
 ```  
  
- Update `azbj\_kclick\_qc\_details`:  
 ```sql  
 UPDATE AZBJ\_KCLICK\_QC\_DETAILS  
 SET QC\_STATUS = 'Y', QC\_PASS\_USER = USER, QC\_PASS\_TIME = SYSDATE  
 WHERE POLICY\_REF = :control.pol\_ref;  
 ```  
  
- Update `azbj\_merging\_tat\_detail`:  
 ```sql  
 UPDATE azbj\_merging\_tat\_detail  
 SET QC\_STATUS = 'Y'  
 WHERE POLICY\_REF = :control.pol\_ref AND QC\_STATUS = 'P';  
 ```  
  
- Delete from `balic.ddt\_124`:  
 ```sql  
 DELETE FROM balic.ddt\_124  
 WHERE field\_269 = v\_application;  
 ```  
  
- Update `customer.azbj\_policy\_versions\_printed`:  
 ```sql  
 UPDATE customer.azbj\_policy\_versions\_printed  
 SET merging\_status = NULL  
 WHERE proposal\_no = :control.pol\_ref;  
 ```  
  
- Insert into `customer.AZBJ\_DYNAMIC\_VALUES`:  
 ```sql  
 INSERT INTO customer.AZBJ\_DYNAMIC\_VALUES (SEQ\_NO, APPLN\_NO, POLICY\_NO, PROPOSAL\_NO, FIELD\_VAL, INSERT\_DATE, INSERT\_USER, COL1, COL2, COL3, col4)  
 VALUES (v\_SEQ\_NO, v\_app\_no, :control.pol\_ref, :control.pol\_ref, v\_field\_val, SYSDATE, USER, v\_nextnavigationblock, v\_next\_item\_1, v\_value, v\_cnt\_item\_block);  
 ```  
  
- Update `azbj\_qc\_kclick\_data`:  
 ```sql  
 UPDATE azbj\_qc\_kclick\_data  
 SET flg = 'N'  
 WHERE policy\_ref = :control.pol\_ref;  
 ```  
  
- Update `azbj\_phub\_qc\_status\_detail`:  
 ```sql  
 UPDATE azbj\_phub\_qc\_status\_detail  
 SET status = st, substatus = subst, ins\_date = SYSDATE  
 WHERE policy\_no = :control.pol\_ref;  
 ```  
  
- Insert into `azbj\_phub\_qc\_status\_detail` if not found:  
 ```sql  
 INSERT INTO azbj\_phub\_qc\_status\_detail (POLICY\_NO, CONTRACT\_ID, STATUS, SUBSTATUS, DE1, DE2, BBU, LOCK\_FLG, INS\_DATE)  
 VALUES (:control.pol\_ref, p\_con\_id, st, subst, p\_de1, p\_de2, p\_bbu, 'N', SYSDATE);  
 ```  
  
- Update `AZBJ\_TR\_POLICY\_SCHEDULE`:  
 ```sql  
 UPDATE AZBJ\_TR\_POLICY\_SCHEDULE  
 SET print\_flag = 'Y'  
 WHERE policy\_ref = :control.pol\_ref AND contract\_id = con\_id;  
 ```  
  
- Update `AZBJ\_POLICY\_SCHEDULE`:  
 ```sql  
 UPDATE AZBJ\_POLICY\_SCHEDULE  
 SET print\_flag = 'Y'  
 WHERE policy\_ref = :control.pol\_ref AND contract\_id = con\_id;  
 ```

# Policy Search and Validation

Type: POL\_SEARCH

User Story: Policy Search and Validation  
  
Detailed Description:  
As a user, I want to search for a policy using the policy reference number and validate various details related to the policy, such as policy status, dates, and other attributes, so that I can ensure the accuracy and completeness of the policy information.  
  
Acceptance Criteria:  
1. The system should allow the user to input a policy reference number and retrieve the corresponding policy details.  
2. The system should validate the existence of the policy reference number. If the policy does not exist, an appropriate message should be displayed.  
3. The system should retrieve and display the following details for the policy:  
 - Date of Proposal  
 - Policy Status  
 - Place of Death  
 - Cause of Death  
 - Date of Information  
 - Docket Received On  
 - All Required Received On  
 - Ensured To  
 - Verified By  
 - Followed By  
 - Death Claim Start Date  
 - Assignment  
 - Reserves  
 - Sum at Risk  
 - Revised Sum Assured  
4. The system should check if the policy has passed the Quality Check (QC). If the policy has passed QC, certain fields should be enabled for further actions.  
5. The system should handle various scenarios such as:  
 - Policies with multiple covers  
 - Policies with joint life or individual life  
 - Policies with different premium terms and benefit terms  
6. The system should display appropriate messages and handle exceptions gracefully, ensuring that the user is informed of any issues during the process.  
  
Definition of Done:  
- The user can input a policy reference number and retrieve the policy details.  
- The system validates the policy reference number and displays an appropriate message if the policy does not exist.  
- The system retrieves and displays all required policy details accurately.  
- The system checks the QC status of the policy and enables/disables fields based on the QC status.  
- The system handles various scenarios and exceptions, providing appropriate messages to the user.  
- The functionality is tested and verified to ensure accuracy and completeness.  
  
DB Queries for Table Reference CRUD Operations:  
- Retrieve policy details based on the policy reference number.  
- Validate the existence of the policy reference number.  
- Retrieve and display various policy attributes.  
- Check the QC status of the policy.  
- Handle scenarios with multiple covers, joint life, and different premium/benefit terms.  
  
Note: The specific SQL queries and database operations are not included as they are dependent on the database schema and structure. The focus is on the functional requirements and user interactions.