User Stories

# Validate and Display Flags and Statuses on New Form Instance

Type: AZBJ\_OTC\_QC

Detailed description: As a user, I need the system to validate and display various flags and statuses related to an application when a new form instance is created. This includes checking for OTC eligibility, SPG case status, KYC flags, and other relevant details. The system should also display appropriate messages and set visibility for specific fields based on the validation results.  
  
Acceptance criteria:  
1. The system should initialize various variables and set default values when a new form instance is created.  
2. The system should validate the user's authorization to use the OTC QC feature.  
3. The system should check for the presence of specific flags and statuses (e.g., OTC eligibility, SPG case status, KYC flags) and set the visibility of corresponding fields accordingly.  
4. The system should display appropriate messages if the user is not authorized to use the OTC QC feature.  
5. The system should log relevant information and errors during the validation process.  
  
Definition of Done:  
1. The system initializes all necessary variables and sets default values when a new form instance is created.  
2. The system validates the user's authorization to use the OTC QC feature and displays a message if the user is not authorized.  
3. The system checks for the presence of specific flags and statuses and sets the visibility of corresponding fields based on the validation results.  
4. The system logs relevant information and errors during the validation process.  
5. The system displays appropriate messages to the user based on the validation results.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- The system should execute the following queries to validate and retrieve necessary information:  
 - Check user authorization:  
 ```sql  
 SELECT COUNT(0) INTO v\_otc\_sub\_profile FROM user\_uw\_limits WHERE user\_id = USER AND uw\_code = 'OTC\_QC';  
 ```  
 - Check for SPG case status:  
 ```sql  
 SELECT CASE WHEN COUNT() > 0 THEN 'Y' ELSE 'N' END INTO v\_spg\_case FROM wip\_policy\_bases x, wip\_policy\_versions y WHERE x.contract\_id = y.contract\_id AND x.policy\_ref = :parameter.policy\_ref AND y.product\_id IN (SELECT char\_value FROM azbj\_system\_constants WHERE sys\_type = 'TELE\_MER' AND sys\_code = 'TELE\_MER\_APPR');  
 ```  
 - Check for OTC eligibility:  
 ```sql  
 SELECT CASE WHEN COUNT() > 0 THEN 'Y' ELSE 'N' END INTO v\_otc\_eligible\_case FROM azbj\_phub\_status\_tracker x WHERE x.application\_no = :dtls\_blk.appln\_no AND (x.status = 'OTC\_QC' OR (x.status = 'Tele\_Video\_Mer\_Recd' AND v\_spg\_case = 'Y' AND x.end\_time IS NULL)) AND NOT EXISTS (SELECT 1 FROM azbj\_phub\_status\_tracker y WHERE y.application\_no = x.application\_no AND y.status LIKE 'FR-AR%' AND y.end\_time IS NULL);  
 ```  
 - Check for KYC flag:  
 ```sql  
 SELECT kyc\_flag, bi\_no INTO v\_kyc\_flag, v\_bi\_no FROM azbj\_proposal\_appln\_det\_ext WHERE appln\_no = :dtls\_blk.appln\_no AND de\_flag = 'D2' AND ROWNUM < 2;  
 ```  
 - Retrieve contract ID:  
 ```sql  
 SELECT contract\_id INTO :control.contract\_id FROM wip\_policy\_bases WHERE policy\_ref = :dtls\_blk.proposal\_no;  
 ```  
 - Retrieve applicant details:  
 ```sql  
 SELECT ip\_title || '. ' || ip\_first\_name || ' ' || ip\_last\_name, ph\_title || '. ' || ph\_first\_name || ' ' || ph\_last\_name, product\_id INTO :dtls\_blk.la\_name, :dtls\_blk.ph\_name, :control.product\_id FROM azbj\_proposal\_appln\_det WHERE appln\_no = TO\_NUMBER(:dtls\_blk.appln\_no) AND de\_flag = 'D1';  
 ```  
 - Check for PhonePe flag:  
 ```sql  
 SELECT CASE WHEN COUNT() > 0 THEN 'Y' ELSE 'N' END INTO v\_phonepe\_flag FROM azbj\_nbtab\_activity\_dtls WHERE application\_no = :dtls\_blk.appln\_no AND MODULE\_FLAG = 'PHONEPE';  
 ```  
 - Retrieve E-KYC flag:  
 ```sql  
 SELECT E\_KYC INTO v\_e\_kyc FROM azbj\_proposal\_appln\_det\_ext WHERE appln\_no = :dtls\_blk.appln\_no AND de\_flag = 'D2' AND ROWNUM < 2;  
 ```  
 - Retrieve SISO flag:  
 ```sql  
 SELECT CASE WHEN COUNT(1) > 0 THEN 'Y' ELSE 'N' END INTO v\_siso\_flag FROM azbj\_appln\_siso\_dtls WHERE application\_no = :dtls\_blk.appln\_no;  
 ```  
 - Retrieve ENACH source:  
 ```sql  
 SELECT SOURCE INTO v\_enach\_source FROM azbj\_nach\_registration\_det WHERE appln\_no = :dtls\_blk.appln\_no;  
 ```  
 - Retrieve channel name:  
 ```sql  
 SELECT channel\_code INTO :DTLS\_BLK.Channel\_Name FROM azbj\_v\_agents WHERE reference\_code = v\_agent\_code AND ROWNUM < 2;  
 ```  
 - Retrieve account type:  
 ```sql  
 SELECT ACCOUNT\_TYPE INTO :DTLS\_BLK.ACCOUNT\_TYPE FROM azbj\_annuity\_prod\_det WHERE appln\_no = :dtls\_blk.appln\_no AND top\_indicator = 'Y';  
 ```  
 - Retrieve premium details:  
 ```sql  
 SELECT trans\_id INTO p\_activity\_id FROM bbu\_trans WHERE appl\_no = :dtls\_blk.appln\_no AND ROWNUM = 1;  
 SELECT TO\_NUMBER(param\_value) INTO v\_premium\_de\_qc FROM bbu\_val\_activity\_dtls WHERE application\_no = :dtls\_blk.appln\_no AND param\_id = 4 AND version\_no = (SELECT MAX(version\_no) FROM bbu\_val\_activity\_dtls WHERE activity\_id = p\_activity\_id);  
 SELECT log\_msg INTO v\_log\_msg FROM bbu\_log\_data WHERE application\_no = :dtls\_blk.appln\_no AND log\_msg LIKE 'SERVICE\_TAX=%' AND log\_time = (SELECT MAX(log\_time) FROM bbu\_log\_data WHERE application\_no = :dtls\_blk.appln\_no AND log\_msg LIKE 'SERVICE\_TAX=%') AND ROWNUM = 1;  
 SELECT SUM(amount) INTO v\_gross\_purchase\_price FROM azbj\_batch\_items WHERE application\_no = :dtls\_blk.appln\_no;  
 SELECT product\_package, annuity\_amount, annuity\_frequency, prm\_term, modes, policy\_option INTO v\_annuity\_type, v\_annuity\_amount, v\_annuity\_freq, v\_premium\_term, v\_mode, v\_annuity\_option FROM carequote\_bi WHERE request\_id = v\_bi\_no OR transaction\_id = v\_bi\_no;  
 ```  
 - Retrieve additional details:  
 ```sql  
 SELECT ext\_user, pasa\_flag, partial\_pasa\_flag, part\_kyc\_ekyc\_flag, pasa\_rt\_tag, cust\_type, retention\_flag INTO v\_ext\_user, v\_pasa\_flag, v\_partial\_pasa\_flag, v\_part\_kyc\_ekyc\_flag, v\_pasa\_rt\_tag, v\_cust\_type, v\_retention\_flag FROM azbj\_proposal\_appln\_det\_ext WHERE appln\_no = :dtls\_blk.appln\_no AND de\_flag = 'D2';  
 SELECT TO\_NUMBER(answer) INTO v\_bmi FROM azbj\_telemer\_answers WHERE appln\_no = :dtls\_blk.appln\_no AND sub\_question = 8;  
 SELECT COUNT(1) INTO v\_edu\_cnt FROM azbj\_telemer\_answers WHERE appln\_no = :dtls\_blk.appln\_no AND sub\_question = 3 AND (UPPER(answer) LIKE '%MAT%' OR UPPER(answer) LIKE '%10%' OR UPPER(answer LIKE '%12%');  
 SELECT COUNT(1) INTO v\_cnt\_advm FROM azbj\_telemer\_answers WHERE appln\_no = :dtls\_blk.appln\_no AND sub\_question BETWEEN 9 AND 38 AND answer = 'Y';  
 SELECT ckyc\_opted, ckyc\_edit, ekyc\_opted, ekyc\_edit, pkyc\_edit, scan\_doc\_opted, scan\_doc\_edit INTO v\_ckyc\_opted, v\_ckyc\_edit, v\_ekyc\_opted, v\_ekyc\_edit, v\_pkyc\_edit, v\_scan\_doc\_opted, v\_scan\_doc\_edit FROM azbj\_instab\_documents\_logs WHERE application\_no = :dtls\_blk.appln\_no;  
 SELECT COUNT() INTO v\_ocr\_cnt FROM azbj\_cq\_doc\_upload\_dtls WHERE APPLICATION\_NO = :dtls\_blk.appln\_no AND OCR\_FLAG = 'Y';  
 SELECT AWG\_WMM\_FLAG INTO v\_AWG\_WMM\_FLAG FROM azbj\_annuity\_prod\_det WHERE appln\_no = :dtls\_blk.appln\_no AND top\_indicator = 'Y' AND ROWNUM < 2;  
 SELECT penny\_drop\_ac\_verification INTO v\_penny\_drop\_ac\_verification FROM azbj\_proposal\_appln\_det\_ext WHERE appln\_no = :dtls\_blk.appln\_no AND de\_flag = 'D1';  
 SELECT comments INTO :dtls\_blk.comments FROM azbj\_uw\_comments a WHERE contract\_id = :control.contract\_id AND event\_no = (SELECT MAX(event\_no) FROM azbj\_uw\_comments b WHERE a.contract\_id = b.contract\_id);  
 SELECT UNIQUE doc\_type, doc\_status, doc\_desc, doc\_category, CASE WHEN doc\_category LIKE '%M68%' THEN 'M68' WHEN doc\_category LIKE '%M038%' THEN 'M038' WHEN doc\_category LIKE '%M234%' THEN 'M234' WHEN doc\_category LIKE '%AGE\_PROOF%' THEN 'M017' WHEN (doc\_category LIKE '%ADDRESS\_PROOF%' OR doc\_category LIKE '%ACS%') THEN 'M107' WHEN doc\_category LIKE '%ID\_PROOF%' THEN 'M108' WHEN doc\_category LIKE '%INCOME\_PROOF%' THEN 'M018' WHEN doc\_category LIKE '%RECENT\_PHOTOGRAPH%' THEN 'M106' WHEN doc\_category LIKE '%CAN\_CHEQUE%' THEN 'M646' WHEN doc\_category LIKE '%COPY\_OF\_PAN%' THEN 'M253' WHEN doc\_category LIKE '%FORM60%' THEN 'M505' WHEN doc\_category LIKE '%PREMIUM\_CHEQUE\_DD%' THEN 'M73' WHEN doc\_category LIKE '%SURROGATE\_INCOME\_PROOF%' THEN 'M750' WHEN doc\_category LIKE '%NACH%' THEN 'M679' WHEN doc\_category LIKE '%OTHERS%' THEN 'M021' END REQ\_CODE FROM azbj\_cq\_doc\_upload\_dtls WHERE application\_no = :dtls\_blk.appln\_no;  
 ```

# Display Test Number in Read-Only Format

Type: BLK\_REQS

Title: Display Test Number in Read-Only Format  
  
Acceptance Criteria:  
1. The test number should be displayed in a read-only field.  
2. The field should be visually distinct with a gray background and black text.  
3. The field should be positioned appropriately on the user interface for easy visibility.  
  
Definition of Done:  
1. The test number is displayed correctly in a read-only format.  
2. The field's appearance matches the specified design (gray background, black text).  
3. The field is positioned correctly on the user interface.  
4. The feature is tested and verified to ensure that the test number cannot be edited by the user.

# Manage Document Information

Type: CP\_ID

Title: Manage Document Information  
  
Acceptance Criteria:  
1. The system should display a list of documents with the following attributes:  
 - Document Name (read-only)  
 - Document Type (read-only)  
 - AML Document Type (read-only)  
 - Proof Type (read-only)  
 - Request Code (read-only)  
 - Document Received status (editable)  
  
2. The Document Received status should be a selectable list with predefined options.  
  
Definition of Done:  
- The user interface displays the document attributes as specified.  
- The Document Received status can be updated by the user.  
- All read-only fields are non-editable and correctly display the relevant information.  
- The layout and design of the interface are user-friendly and consistent with the rest of the application.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Not applicable as the provided XML content does not include specific CRUD operations or SQL queries.

# Indicate Document Received Status in Quality Control

Type: CP\_ID

Title: Indicate Document Received Status in Quality Control  
  
Acceptance Criteria:  
1. The user should be able to select from a predefined list of options to indicate the document received status.  
2. The document received status should be displayed on the quality control tab.  
3. The user should be able to update the document received status if necessary.  
4. The document received status should be saved and retrievable for future reference.  
  
Definition of Done:  
1. The user can see a field labeled "Doc Received" in the quality control section.  
2. The field should allow the user to select from a list of predefined options.  
3. The user should be able to save the selected option.  
4. The saved status should be retrievable and displayed correctly when the user revisits the quality control section.  
  
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.

# Manage Comments Section

Type: UW\_COMMENTS

Detailed description: As a user, I want to manage comments within the system, so that I can add, view, and exit the comments section efficiently.  
  
Acceptance criteria:  
1. The comments section should display a text area where users can input their comments.  
2. There should be a checkbox labeled "Click to add Comments" that, when checked, enables the comments text area for input.  
3. An "Add Comments" button should be available to save the entered comments.  
4. An "Exit" button should be present to allow users to exit the comments section.  
  
Definition of Done:  
- The comments text area is visible and allows for multiline input.  
- The checkbox correctly toggles the ability to add comments.  
- The "Add Comments" button saves the entered comments.  
- The "Exit" button successfully exits the comments section.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include specific database queries.

# Exit Button Functionality

Type: UW\_COMMENTS

Title: Exit Button Functionality  
  
Acceptance Criteria:  
1. When the "Exit" button is pressed, the comments section should be hidden.  
2. The system should navigate to the comments field in the details section.  
3. The system should retrieve the latest comment for the given contract ID from the database.  
4. If no comment is found, the comments field should be set to null.  
  
Definition of Done:  
- The "Exit" button hides the comments section.  
- The system navigates to the comments field in the details section.  
- The latest comment for the given contract ID is retrieved and displayed.  
- If no comment is found, the comments field is set to null.  
  
DB queries for Table reference CRUD operations only(With Usage):  
```sql  
-- Retrieve the latest comment for a specific contract  
SELECT comments  
INTO :DTLS\_BLK.COMMENTS  
FROM azbj\_uw\_comments a  
WHERE contract\_id = :control.contract\_id  
AND event\_no = (SELECT MAX(event\_no)   
 FROM azbj\_uw\_comments b   
 WHERE a.contract\_id = b.contract\_id);  
```

# Enable/Disable Comments Section Based on Checkbox State

Type: UW\_COMMENTS

Title: Enable/Disable Comments Section Based on Checkbox State  
  
Acceptance Criteria:  
1. When the checkbox is checked, the comments input field should be enabled.  
2. When the checkbox is unchecked, the comments input field should be disabled.  
3. The checkbox should have a label prompting the user to click to add comments.  
  
Definition of Done:  
- The checkbox functionality is implemented and tested.  
- The comments input field correctly toggles between enabled and disabled states based on the checkbox.  
- The user interface displays the checkbox with the appropriate label.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include any specific database queries.

# Add Comments Functionality

Type: UW\_COMMENTS

Title: Add Comments Functionality  
  
Acceptance Criteria:  
1. When the "Add Comments" button is pressed, the system should:  
 - Retrieve the current username.  
 - Determine the next event number for the given contract by checking the maximum event number in the `azbj\_uw\_comments` table and incrementing it by one. If no event number is found, it should default to 1.  
 - If the comment field is not empty, insert a new record into the `azbj\_uw\_comments` table with the following details:  
 - Event number  
 - Contract ID  
 - Policy number  
 - User ID  
 - Current date  
 - Comment text  
 - A flag set to 'Y'  
 - Clear the comment field after saving.  
 - Display a success message indicating that the comments were saved successfully.  
 - If the comment field is empty, it should simply clear the field.  
 - Commit the transaction.  
 - Refresh the comments block and clear any unsaved data.  
  
Definition of Done:  
- The "Add Comments" button functionality is implemented and tested.  
- The system correctly retrieves the next event number and handles cases where no previous event number exists.  
- Comments are saved to the database with all required fields.  
- Appropriate success and error messages are displayed.  
- The comments block is refreshed and cleared after the operation.  
- All acceptance criteria are met and verified through testing.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Retrieve the next event number:  
 ```sql  
 SELECT NVL(MAX(event\_no), 0) + 1  
 INTO v\_event\_no  
 FROM azbj\_uw\_comments  
 WHERE contract\_id = :control.contract\_id;  
 ```  
  
- Insert a new comment:  
 ```sql  
 INSERT INTO azbj\_uw\_comments  
 (event\_no, contract\_id, policy\_no, user\_id, comment\_date, comments, flag)  
 VALUES (v\_event\_no, :control.contract\_id, :dtls\_blk.proposal\_no, v\_user, SYSDATE, :uw\_comments.uw\_comment, 'Y');  
 ```

# User Input and Display for PAN Details

Type: AZBJ\_PAN\_DET

Title: User Input and Display for PAN Details  
  
Acceptance Criteria:  
1. The form should display fields for entering and viewing the following details:  
 - PAN Number  
 - PAN Status  
 - Name Match  
 - Date of Birth (DOB) Match  
2. Each field should have a clear and centered prompt for user guidance.  
3. The form should allow for a maximum of 3 records to be displayed at a time.  
4. The form should include a scrollbar for navigating through records.  
5. The form should be visually organized with appropriate spacing and alignment for ease of use.  
  
Definition of Done:  
- The form is implemented and tested to ensure that it meets the acceptance criteria.  
- The form is free of any Oracle Forms-specific terminology and can be used independently of any specific technology.  
- The form 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 specific database queries.

# Manage Image Details

Type: IMAGE\_DET

Title: Manage Image Details  
  
Acceptance Criteria:  
1. The system should allow users to input and display the following details for each image:  
 - Image Path  
 - Proposal Number  
 - Application Number  
 - Image Name  
 - Image Scan Time  
 - Image Size (in KB)  
 - Number of Pages  
2. The system should provide a button to view the image.  
3. The system should provide a button to hide the image details.  
  
Definition of Done:  
1. Users can successfully input and save image details.  
2. Users can view the image by clicking the "View Image" button.  
3. Users can hide the image details by clicking the "Hide" button.  
4. All fields and buttons are displayed correctly and are functional.  
5. The system handles up to 5 records of image details at a time.  
  
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.

# View Image Functionality

Type: IMAGE\_DET

Title: View Image Functionality  
  
Acceptance Criteria:  
1. When the "View Image" button is pressed, the system should:  
 - Retrieve the image path from the database.  
 - Construct a local file path using the proposal number and image name.  
 - Transfer the image file from the server to the local machine.  
 - Log the transfer details including the contract ID and application number.  
 - Open the image using the appropriate application based on the file type (e.g., PDF viewer for PDF files, image viewer for other image types).  
 - Handle any errors during the process by logging the error details and displaying an error message to the user.  
  
Definition of Done:  
- The "View Image" button functionality is implemented and tested.  
- The image is successfully transferred from the server to the local machine.  
- The image opens in the appropriate application based on its file type.  
- Error handling is in place, and errors are logged and displayed to the user.  
- The functionality is verified through user acceptance testing.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include specific CRUD operations.

# Navigate to Proposal Number Details on Hide Button Press

Type: IMAGE\_DET

Title: Navigate to Proposal Number Details on Hide Button Press  
  
Acceptance Criteria:  
- When the "Hide" button is pressed, the system should automatically navigate to the proposal number details section.  
  
Definition of Done:  
- The "Hide" button is functional and correctly navigates to the proposal number details section upon being pressed.  
- The navigation should be seamless and without any errors.  
- The user should be able to see the proposal number details immediately after pressing the "Hide" button.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include any specific database queries.

# View Application Details

Type: DEQC\_DISPLAY

Detailed description: As a user, I want to view the details of various applications, including their status, receipt information, and associated names, so that I can easily track and manage the applications.  
  
Acceptance criteria:  
1. The application details should include the following fields:  
 - Application Number  
 - Application Status  
 - Receipt Number  
 - Receipt Date  
 - LA Name  
 - PH Name  
 - Change Description  
 - Proposal Number  
 - Partner Name  
 - Document Status  
 - Pending Documents  
 - A checkbox to mark the application as checked  
  
2. Each field should be displayed with appropriate labels and should be read-only where applicable.  
  
Definition of Done:  
1. The application details are displayed on a single screen.  
2. All fields are correctly labeled and aligned.  
3. Read-only fields cannot be edited by the user.  
4. The checkbox can be toggled between checked and unchecked states.  
5. The user interface is intuitive and user-friendly.  
  
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.

# Enable/Disable Action Buttons Based on Checkbox State

Type: DEQC\_DISPLAY

Detailed description: As a user, I want the system to enable or disable specific action buttons based on the selection of a checkbox in the DEQC\_DISPLAY section, so that I can perform relevant actions only when the checkbox is checked.  
  
Acceptance criteria:  
1. When the checkbox in the DEQC\_DISPLAY section is checked:  
 - The "LINKSAVE" button should be enabled.  
 - The "VIEW\_DOCS" button should be enabled.  
 - The "UPLOAD\_DOCS" button should be disabled.  
 - The "QC" button should be enabled.  
 - The "REJECT" button should be enabled.  
2. When the checkbox in the DEQC\_DISPLAY section is unchecked:  
 - The "LINKSAVE" button should be disabled.  
 - The "VIEW\_DOCS" button should be disabled.  
 - The "UPLOAD\_DOCS" button should be disabled.  
 - The "QC" button should be disabled.  
 - The "REJECT" button should be disabled.  
  
Definition of Done:  
- The system correctly enables or disables the action buttons based on the checkbox state.  
- The functionality is tested and verified to work as expected.  
- The user interface reflects the changes immediately upon checking or unchecking the checkbox.  
- The changes are documented and reviewed.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include any direct database queries.

# View and Manage Underwriter Comments

Type: SHOW\_UW\_COMMENTS

Detailed description: As a user, I want to view and manage underwriter comments in a dedicated section so that I can keep track of user comments along with their timestamps.  
  
Acceptance criteria:  
1. The section should display the following fields:  
 - User ID: Display-only field showing the ID of the user who made the comment.  
 - Date & Time: Display-only field showing the timestamp of the comment.  
 - UW Comments: Editable field where the underwriter's comments are displayed.  
2. The section should include the following buttons:  
 - Add Comments: Button to add new comments.  
 - Refresh: Button to refresh the comments section.  
3. The comments section should be scrollable if there are more comments than can be displayed at once.  
  
Definition of Done:  
1. The user can see the User ID and Date & Time fields as non-editable.  
2. The user can view and edit the UW Comments field.  
3. The user can add new comments by clicking the "Add Comments" button.  
4. The user can refresh the comments section by clicking the "Refresh" button.  
5. The comments section should be scrollable if necessary.  
  
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.

# Refresh Underwriter Comments

Type: SHOW\_UW\_COMMENTS

Title: Refresh Underwriter Comments  
  
Acceptance Criteria:  
1. When the refresh button is pressed, the system should clear the current list of underwriter comments.  
2. If the user ID starts with 'P00%' or the control profile is not equal to '1', the system should fetch all comments related to the contract ID from the AZBJ\_UW\_COMMENTS table.  
3. If the user ID does not start with 'P00%' and the control profile is equal to '1', the system should fetch only the comments with a flag 'N' related to the contract ID from the AZBJ\_UW\_COMMENTS table.  
4. The fetched comments should be displayed in the underwriter comments section, with each comment showing the user ID, comment date, and the comment text.  
5. The system should navigate to the first record of the underwriter comments after the refresh operation is completed.  
  
Definition of Done:  
- The refresh button functionality is implemented and tested.  
- The system correctly fetches and displays the comments based on the user profile and contract status.  
- The user can see the updated list of comments immediately after pressing the refresh button.  
- All acceptance criteria are met and verified through testing.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Fetch all comments for a contract:  
 ```sql  
 SELECT FROM AZBJ\_UW\_COMMENTS WHERE contract\_id = :control.contract\_id;  
 ```  
- Fetch comments with flag 'N' for a contract:  
 ```sql  
 SELECT FROM AZBJ\_UW\_COMMENTS WHERE contract\_id = :control.contract\_id AND flag = 'N';  
 ```

# View and Edit Underwriting Comments

Type: SHOW\_UW\_COMMENTS

Title: View and Edit Underwriting Comments  
  
Acceptance Criteria:  
1. When a user double-clicks on the comments field, an editor window should open.  
2. The editor window should display the current comments.  
3. The user should be able to edit the comments within the editor window.  
4. The changes made in the editor window should be saved back to the comments field.  
  
Definition of Done:  
1. The editor window opens upon double-clicking the comments field.  
2. The current comments are displayed in the editor window.  
3. The user can edit and save the comments.  
4. The updated comments are reflected in the comments field after closing the editor window.

# Add Comments Functionality

Type: SHOW\_UW\_COMMENTS

Title: Add Comments Functionality  
  
Acceptance Criteria:  
1. When the "Add Comments" button is pressed, the system should capture the current contract ID and user ID.  
2. The system should store these values globally for use in subsequent processes.  
3. The system should then open a form or interface where the user can input their comments related to the underwriting process.  
  
Definition of Done:  
- The "Add Comments" button is functional and triggers the process of capturing the contract ID and user ID.  
- The captured values are stored globally and can be accessed by other parts of the system.  
- A new form or interface is opened where the user can input their comments.  
- The process is tested and verified to ensure that comments can be added successfully.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include any specific database queries.

# View and Interact with Proposal Details

Type: DTLS\_BLK

Title: View and Interact with Proposal Details  
  
Acceptance Criteria:  
1. The user should be able to view the following fields:  
 - Proposal Number  
 - Life Assured Name  
 - Application Number  
 - Policy Holder Name  
 - Comments  
 - Refer to Supervisor  
 - TRL Score  
 - Income Estimate  
 - Cibil Score  
 - Account Type  
 - Channel Name  
 - KYC Status  
 - Existing Customer Status  
 - PASA Flag  
 - Partial PASA  
 - Penny Drop Successful  
 - Partial KYC  
 - OCR Flag  
 - Net Purchase Price  
 - Gross Purchase  
 - Annuity Amount  
 - Annuity Option  
 - Annuity Type  
 - Premium Term  
 - Annuity Frequency  
 - Customer Type  
 - Retention  
 - EDC Persistency Flag  
 - SISO Flag  
 - ENACH Status  
 - Medicals  
 - GSIP Flag  
 - WMM Flag  
  
2. The user should be able to perform the following actions:  
 - Submit the proposal details.  
 - Exit the form.  
 - View all related documents.  
 - Open the proposal form in an image viewer.  
 - Download KYC documents.  
  
Definition of Done:  
- The user can view all the specified fields with the correct data.  
- The user can perform all the specified actions without any errors.  
- The form should be user-friendly and intuitive.  
- All fields should be read-only except for the action buttons.  
- The form should be 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 specific SQL queries or database operations.

# Exit Button Functionality

Type: DTLS\_BLK

Title: Exit Button Functionality  
  
Acceptance Criteria:  
- When the user clicks the "Exit" button, the current form should close.  
- The system should handle any exceptions that occur during the exit process gracefully, ensuring that no error messages are displayed to the user.  
  
Definition of Done:  
- The "Exit" button is visible and clickable on the form.  
- Clicking the "Exit" button successfully closes the form.  
- Any potential errors during the exit process are managed without disrupting the user experience.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include any specific database CRUD operations.

# Implement Refer to Supervisor Dropdown in QC Section

Type: DTLS\_BLK

Title: Implement Refer to Supervisor Dropdown in QC Section  
  
Acceptance Criteria:  
1. The field should be a dropdown list with two options: "Yes" and "No".  
2. The default value of the dropdown should be "No".  
3. The field should be located in the QC section of the application.  
4. The field should be clearly labeled as "Refer to Supervisor".  
5. The field should be easily accessible and visible on the screen.  
  
Definition of Done:  
1. The dropdown field is implemented and visible in the QC section.  
2. The dropdown field has the options "Yes" and "No", with "No" as the default value.  
3. The field is labeled "Refer to Supervisor".  
4. The field is tested and confirmed to be working as expected.  
5. The changes are reviewed and approved by the 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.

# View Proposal Form

Type: DTLS\_BLK

Title: View Proposal Form  
  
Acceptance Criteria:  
1. When the "Proposal Form" button is pressed, the system should set a flag indicating that the proposal form is being viewed.  
2. The system should generate a URL for the document viewer using the application number.  
3. If the URL is successfully generated, the system should open the document viewer in a web browser.  
4. If there is an error in generating the URL, the system should display an error message indicating that the URL could not be checked.  
  
Definition of Done:  
- The "Proposal Form" button is functional and triggers the appropriate actions.  
- The URL for the document viewer is correctly generated and opened in a web browser.  
- Error handling is in place to notify the user if the URL cannot be generated.  
- The flag indicating the proposal form view is correctly set and reset as needed.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include any direct database CRUD operations.

# View All Documents Functionality

Type: DTLS\_BLK

Title: View All Documents Functionality  
  
Acceptance Criteria:  
1. When the "View All Documents" button is pressed, a temporary directory should be created on the local machine to store the documents.  
2. The system should navigate to the document details section and iterate through all records to check for available document paths.  
3. If a document path is found, the document should be copied from the server to the local temporary directory.  
4. If the document is a PDF, it should be opened using the default PDF viewer.  
5. If the document is an image, it should be opened using the default image viewer.  
6. The process should handle both server file paths and URLs, ensuring compatibility with different file sources.  
7. Any errors encountered during the process should be handled gracefully without interrupting the user experience.  
  
Definition of Done:  
- The "View All Documents" button functionality is implemented and tested.  
- The temporary directory is created successfully on the local machine.  
- Documents are copied from the server to the local machine and opened correctly based on their file type.  
- The system handles both server file paths and URLs.  
- Error handling is in place to manage any issues during the document retrieval and viewing process.  
- The feature 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 specific CRUD operations or SQL queries.

# Download KYC Document

Type: DTLS\_BLK

Title: Download KYC Document  
  
Acceptance Criteria:  
1. When the "DOWNLOAD\_KYC\_DOC" button is pressed, the system should:  
 - Retrieve the application number from the current record.  
 - Fetch the contract ID associated with the proposal number.  
 - Retrieve personal details (TAX\_ID, DATE\_OF\_BIRTH, SEX, FIRST\_NAME, MIDDLE\_NAME, SURNAME) of the applicant from the database.  
 - Format the date of birth in 'DD-MM-YYYY' format.  
 - Construct a JSON string with the retrieved details.  
 - Send the JSON string to a predefined URL for KYC validation.  
 - Display the response message from the KYC validation service.  
 - Navigate to the 'CP\_ID' block and iterate through the records.  
 - For each record, if the 'req\_code' is 'M107', 'M253', or 'M106', set the 'DOC\_RCVD' field to 'Y' and ensure it is editable.  
 - Handle any exceptions that occur during the process and display an appropriate error message.  
  
Definition of Done:  
- The "DOWNLOAD\_KYC\_DOC" button functionality is implemented as per the acceptance criteria.  
- The system successfully retrieves and processes the required data.  
- The KYC validation service is called, and the response is handled correctly.  
- The 'CP\_ID' block records are updated as specified.  
- Appropriate error handling is in place.  
- The feature is tested and verified to work as expected.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Retrieve contract ID:  
 ```sql  
 SELECT azbj\_pk0\_acc.get\_contract\_id(:DTLS\_BLK.PROPOSAL\_NO) INTO v\_contract\_id FROM dual;  
 ```  
  
- Retrieve personal details:  
 ```sql  
 SELECT TAX\_ID, DATE\_OF\_BIRTH, SEX, FIRST\_NAME, MIDDLE\_NAME, SURNAME  
 INTO v\_ph\_pan\_no, v\_ph\_dob, v\_ph\_sex, v\_ph\_FIRST\_NAME, v\_MIDDLE\_NAME, v\_SURNAME  
 FROM cp\_partners a, wip\_interested\_parties b  
 WHERE CONTRACT\_ID = v\_contract\_id  
 AND a.PART\_ID = b.PARTNER\_ID  
 AND b.IP\_NO = 2;  
 ```

# Validate Tele/Video Mer (Approved) Selection

Type: DTLS\_BLK

Title: Validate Tele/Video Mer (Approved) Selection  
  
Acceptance Criteria:  
1. When the user attempts to select the "Tele/Video Mer (Approved)" option, the system should check if the document has been verified.  
2. If the document has not been verified (i.e., `v\_tele\_video\_check` is 'N'), the system should display a warning message: "Please verify the document before selecting Yes or No option".  
3. If the document has not been verified, the selection should be reset to null.  
4. If an error occurs during this process, the system should display an error message with the SQL error code and message.  
  
Definition of Done:  
- The system correctly checks the verification status before allowing the selection of the "Tele/Video Mer (Approved)" option.  
- Appropriate warning and error messages are displayed as per the acceptance criteria.  
- The selection is reset to null if the document is not verified.  
- The feature is tested and validated 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 direct CRUD operations on database tables.

# View and Edit Comments in Quality Control Section

Type: DTLS\_BLK

Title: View and Edit Comments in Quality Control Section  
  
Acceptance Criteria:  
1. When a user double-clicks on the comments field, a new view should be displayed.  
2. The new view should allow the user to navigate to a specific comments section.  
3. The comments field should be multi-line and have a maximum length of 1000 characters.  
4. The comments field should be read-only and not allow updates directly in the main view.  
  
Definition of Done:  
1. The comments section is accessible via a double-click action.  
2. The new view for comments is displayed correctly and allows navigation to the specific comments section.  
3. The comments field supports multi-line text and enforces the maximum length constraint.  
4. The comments field is read-only in the main view, ensuring no direct updates are allowed.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include any specific database queries.

# Submit Application Form with Document Validation

Type: DTLS\_BLK

Detailed description: As a user, I want to be able to submit the application form after ensuring all required documents are received and validated, so that the application can proceed to the next stage of processing.  
  
Acceptance criteria:  
1. The system should check if the KYC documents have been downloaded before proceeding.  
2. The system should validate the presence of required documents such as age proof, identity proof, income proof, permanent address proof, and current address proof.  
3. If any required document is missing, the system should raise the appropriate request for the missing document.  
4. The system should update the status of the application based on the received documents and validation results.  
5. The system should log all actions and updates made during the submission process.  
6. If all required documents are received and validated, the system should proceed to update the application status and log the event.  
7. If any validation fails, the system should display an appropriate error message and prevent further processing.  
  
Definition of Done:  
1. The user can successfully submit the application form if all required documents are received and validated.  
2. The system raises appropriate requests for any missing documents.  
3. The application status is updated based on the validation results.  
4. All actions and updates are logged for audit purposes.  
5. The user is notified of any errors or missing information that prevents submission.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- The system should perform the following database operations:  
 - Select data from `azbj\_proposal\_appln\_det` to validate the application number.  
 - Insert records into `azbj\_otc\_testno` for received documents.  
 - Update `azbj\_proposal\_appln\_det\_ext` with remarks based on document validation.  
 - Insert records into `azbj\_aml\_nb\_records\_new` for AML checks.  
 - Update `azbj\_annuity\_prod\_det` with the FTR flag.  
 - Insert records into `customer.azbj\_ftr\_req\_dtls` for FTR requirements.  
 - Update `azbj\_phub\_tracker` with the proposal status.  
 - Insert records into `azbj\_auto\_issuance\_grp` for auto issuance.  
 - Insert records into `azbj\_pol\_activity\_log` for activity logging.  
 - Insert records into `azbj\_uw\_comments` for user comments.

# Manage Account Details

Type: ACCOUNT\_DET

Title: Manage Account Details  
  
Acceptance Criteria:  
1. The system should allow the user to input the following account details:  
 - Account Holder Name  
 - Bank Name  
 - Branch  
 - MICR  
 - IFSC Code  
 - Type of Account  
 - Account Number  
2. The system should validate the IFSC code by querying the bank details from the database.  
 - DB Query:   
 ```sql  
 SELECT BANK\_IFSC, BANK\_NAME, BANK\_BRANCH, BANK\_MICR  
 FROM azbj\_bank\_ifsc\_detail  
 WHERE BANK\_IFSC = :ACCOUNT\_DET.IFSC\_CODE  
 ```  
3. The system should provide options to specify the relationship with the premium payer and third-party payer.  
4. The system should display feedback on the success or failure of penny drop transactions.  
5. The system should allow the user to bypass bank details if necessary.  
  
Definition of Done:  
- The user can input and save all required account details.  
- The system validates the IFSC code against the database and retrieves the corresponding bank details.  
- The user can specify relationships with premium payers and third-party payers.  
- The system displays appropriate feedback for penny drop transactions.  
- The user can bypass bank details when needed.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- The provided query for validating the IFSC code is used to fetch bank details from the `azbj\_bank\_ifsc\_detail` table.

# Dropdown for IP Relation with Premium Payer

Type: ACCOUNT\_DET

Title: Dropdown for IP Relation with Premium Payer  
  
Acceptance Criteria:  
1. The user should be able to see a dropdown list labeled "Ip Relation with Premium payer" on the QC tab.  
2. The dropdown list should contain 18 predefined options for the relationship between the insurance policy and the premium payer.  
3. The dropdown list should be positioned appropriately on the screen for easy access and visibility.  
4. The dropdown list should be editable, allowing the user to select one of the predefined options.  
  
Definition of Done:  
1. The dropdown list is implemented and visible on the QC tab.  
2. The dropdown list contains 18 predefined options.  
3. The user can select an option from the dropdown list.  
4. The selected option is saved and can be retrieved accurately.  
  
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.

# Implement Relationship Type Dropdown

Type: ACCOUNT\_DET

Title: Implement Relationship Type Dropdown  
  
Acceptance Criteria:  
1. The relationship type should be selectable from a dropdown list containing predefined options.  
2. The dropdown list should be labeled "Relationship" and should be located on the "QC" tab of the user interface.  
3. The dropdown list should be positioned at the specified coordinates within the user interface.  
4. The selected relationship type should be saved and associated with the account details.  
  
Definition of Done:  
1. The dropdown list for selecting the relationship type is implemented and visible on the "QC" tab.  
2. The dropdown list contains the predefined options for relationship types.  
3. The selected relationship type is saved and correctly associated with the account details.  
4. The user interface is tested to ensure the dropdown list functions as expected and the data is saved correctly.  
  
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.

# Auto-fetch Bank Details using IFSC Code

Type: ACCOUNT\_DET

Title: Auto-fetch Bank Details using IFSC Code  
  
Acceptance Criteria:  
1. When the user enters an IFSC code and moves to the next field, the system should automatically retrieve the bank name, branch, and MICR code from the database.  
2. If the IFSC code is not found in the database, the system should prompt the user with a list of possible IFSC codes to choose from.  
  
Definition of Done:  
1. The IFSC code field should accept a maximum of 15 characters and convert all input to uppercase.  
2. The system should successfully fetch and display the bank name, branch, and MICR code when a valid IFSC code is entered.  
3. If the IFSC code is invalid, the system should display a list of possible IFSC codes for the user to select from.  
4. The user interface should be intuitive and user-friendly, with clear prompts and error messages.  
  
DB queries for Table reference CRUD operations only(With Usage):  
```sql  
SELECT BANK\_NAME, BANK\_BRANCH, BANK\_MICR  
 INTO :ACCOUNT\_DET.BANK\_NAME,  
 :ACCOUNT\_DET.ACC\_BRANCH,  
 :ACCOUNT\_DET.MICR  
 FROM azbj\_bank\_ifsc\_detail  
 WHERE BANK\_IFSC = :ACCOUNT\_DET.IFSC\_CODE;  
```  
  
```sql  
SELECT BANK\_IFSC, BANK\_NAME, BANK\_BRANCH, BANK\_MICR  
 FROM azbj\_bank\_ifsc\_detail  
 WHERE BANK\_IFSC = :ACCOUNT\_DET.IFSC\_CODE;  
```

# Auto-populate Bank Details Based on Account Type Selection

Type: ACCOUNT\_DET

Title: Auto-populate Bank Details Based on Account Type Selection  
  
Acceptance Criteria:  
1. When the user selects a type of account from the list, the system should automatically fetch and display the corresponding bank name, bank branch, and bank MICR code.  
2. If the system cannot find the bank details for the selected account type, it should clear any previously populated bank details.  
  
Definition of Done:  
- The user can select the type of account from a list.  
- Upon selection, the bank name, bank branch, and bank MICR code fields are automatically populated.  
- If no matching bank details are found, the bank name, bank branch, and bank MICR code fields are cleared.  
  
DB queries for Table reference CRUD operations only(With Usage):  
```sql  
SELECT BANK\_NAME, BANK\_BRANCH, BANK\_MICR  
 INTO :ACCOUNT\_DET.BANK\_NAME,  
 :ACCOUNT\_DET.ACC\_BRANCH,  
 :ACCOUNT\_DET.MICR  
 FROM azbj\_bank\_ifsc\_detail  
 WHERE BANK\_IFSC = :ACCOUNT\_DET.IFSC\_CODE;  
```

# Bypass Bank Details Functionality

Type: ACCOUNT\_DET

Title: Bypass Bank Details Functionality  
  
Acceptance Criteria:  
1. When the "Bypass Bank Details" checkbox is checked, the following fields should be disabled:  
 - Account Number  
 - Account Holder Name  
 - Bank Name  
 - Account Relation  
 - Account Branch  
 - IFSC Code  
 - MICR  
 - Account TPP Relation  
2. When the "Bypass Bank Details" checkbox is unchecked, the above fields should be enabled.  
3. The system should clear the account details block and set the focus to the Account Number field when the checkbox state changes.  
  
Definition of Done:  
- The "Bypass Bank Details" checkbox functionality is implemented and tested.  
- The fields are correctly enabled or disabled based on the checkbox state.  
- The account details block is cleared and focus is set to the Account Number field when the checkbox state changes.  
- The feature 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 direct database queries.

# View Same Bank Account Details

Type: ACCOUNT\_DET

Title: View Same Bank Account Details  
  
Acceptance Criteria:  
1. When the button labeled "Same Bank Dtls" is pressed, the system should check if both the Account Number and IFSC Code fields are not empty.  
2. If either the Account Number or IFSC Code fields are empty, the system should display a warning message: "Please enter the Account No and IFSC Code.!" and prevent further action.  
3. If both fields are filled, the system should pass the Account Number, IFSC Code, form name, and IP part ID as parameters to another form that displays the same bank details.  
4. The system should handle any unexpected errors gracefully without crashing.  
  
Definition of Done:  
- The button functionality is implemented and tested.  
- The system correctly validates the presence of Account Number and IFSC Code.  
- The system displays the appropriate warning message when required fields are missing.  
- The system successfully passes the required parameters to the form displaying same bank details.  
- Error handling is in place to manage unexpected issues.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided logic does not include direct database CRUD operations.

# Search and Manage Records

Type: DEQC\_SAERCH

Title: Search and Manage Records  
  
Acceptance Criteria:  
1. The user should be able to input a "From Date" and "To Date" to specify the date range for the search.  
2. The user should be able to enter an "Application Number" to filter records by a specific application.  
3. The user should be able to select a "Partner Type" from a list to filter records by partner type.  
4. The user should be able to select a "Status" from a list to filter records by status.  
5. The user should be able to click a "Search" button to execute the search based on the provided criteria.  
6. The user should be able to click a "Clear" button to reset all search criteria fields.  
7. The user should be able to click a "Save" button to save the current state or data.  
8. The user should be able to click a "View Docs" button to view associated documents.  
9. The user should be able to click an "Upload Docs" button to upload new documents.  
10. The user should be able to click a "QC/Edit" button to perform quality control or edit the record.  
11. The user should be able to click a "UW Comments" button to view or add underwriter comments.  
12. The user should be able to click a "Reject" button to reject the record.  
13. The user should be able to enter a reason in a text field and click an "OK" button to confirm the reason.  
  
Definition of Done:  
1. The search functionality should be implemented and tested to ensure it filters records based on the specified criteria.  
2. The clear functionality should reset all input fields to their default state.  
3. The save functionality should save the current state or data as required.  
4. The view documents functionality should display associated documents.  
5. The upload documents functionality should allow users to upload new documents.  
6. The QC/Edit functionality should allow users to perform quality control or edit the record.  
7. The underwriter comments functionality should display or allow adding comments.  
8. The reject functionality should mark the record as rejected.  
9. The reason input and confirmation functionality should work as expected.  
10. All functionalities should be tested and verified to ensure they meet the acceptance criteria.

# Document Upload Functionality

Type: DEQC\_SAERCH

Title: Document Upload Functionality  
  
Acceptance Criteria:  
- When the "Upload Docs" button is pressed, the system should initiate the document upload process.  
- The system should call the appropriate forms or modules to handle the document upload.  
  
Definition of Done:  
- The "Upload Docs" button is visible and functional on the interface.  
- Pressing the button successfully triggers the document upload process.  
- The appropriate forms or modules are called to handle the document upload without errors.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include any specific database queries.

# Clear Button Functionality on Search Interface

Type: DEQC\_SAERCH

Title: Clear Button Functionality on Search Interface  
  
Acceptance Criteria:  
- When the clear button is pressed, all fields in the search form should be reset to their default values.  
- The form should not perform any validation checks when the clear button is pressed.  
  
Definition of Done:  
- The clear button is visible and accessible on the search interface.  
- Pressing the clear button successfully resets all form fields to their default state.  
- No validation is triggered when the clear button is pressed.

# Navigate to Reason Link on UW Comments Button Press

Type: DEQC\_SAERCH

Title: Navigate to Reason Link on UW Comments Button Press  
  
Acceptance Criteria:  
1. When the "UW Comments" button is pressed, the system should display the "Reason Link" section.  
2. The focus should automatically move to the "Reason Link" input field after the section is displayed.  
  
Definition of Done:  
- The "UW Comments" button is functional and triggers the display of the "Reason Link" section.  
- The focus is set to the "Reason Link" input field upon displaying the section.  
- The feature is tested and verified to work as expected in the user interface.

# Implement Partner Type Selection in Search Interface

Type: DEQC\_SAERCH

Title: Implement Partner Type Selection in Search Interface  
  
Acceptance Criteria:  
1. The partner type field should display a list of predefined partner types.  
2. The list should be easily accessible and visible on the search interface.  
3. The partner type field should be located on the search tab of the user interface.  
4. The field should not be mandatory, allowing users to perform searches without selecting a partner type.  
  
Definition of Done:  
1. The partner type field is implemented and displays a list of predefined partner types.  
2. The field is positioned correctly on the search tab of the user interface.  
3. The field is tested to ensure it is not mandatory and does not hinder the search functionality if left unselected.  
4. The user interface is reviewed and approved for usability and functionality.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include any specific SQL queries or database operations.

# View Documents Functionality

Type: DEQC\_SAERCH

Title: View Documents Functionality  
  
Acceptance Criteria:  
1. When the user clicks the "View Docs" button, the system should check if exactly one record is selected.  
2. If more than one record is selected, the system should display an error message indicating that only one record should be selected.  
3. If exactly one record is selected, the system should generate a URL for the document based on the application number of the selected record.  
4. The system should open the generated URL in a web browser.  
5. If the URL is not valid or cannot be generated, the system should display a warning message.  
  
Definition of Done:  
- The "View Docs" button functionality is implemented and tested.  
- The system correctly identifies if more than one record is selected and displays an appropriate error message.  
- The system generates the correct URL for the document based on the application number.  
- The URL is opened in a web browser when valid.  
- Appropriate warning messages are displayed if the URL cannot be generated or is invalid.  
- All acceptance criteria are met and verified through testing.

# Validate 'To Date' Field in Search Form

Type: DEQC\_SAERCH

Title: Validate 'To Date' Field in Search Form  
  
Acceptance Criteria:  
1. When the user enters a 'To Date' that results in a date difference of more than six months from the 'From Date', the system should disable the search functionality and display an error message indicating that the date difference should not be greater than six months.  
2. When the user enters a 'To Date' that is earlier than the 'From Date', the system should disable the search functionality and display an error message indicating that the 'To Date' should be greater than the 'From Date'.  
  
Definition of Done:  
- The 'To Date' field is validated upon entry.  
- Appropriate error messages are displayed when validation fails.  
- The search functionality is disabled when validation fails.  
- The system ensures that the 'To Date' is always greater than the 'From Date' and the date difference does not exceed six months.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include any CRUD operations directly executable in the database.

# Implement Status Dropdown in WEBSALES Tab

Type: DEQC\_SAERCH

Title: Implement Status Dropdown in WEBSALES Tab  
  
Acceptance Criteria:  
1. The status field should be a dropdown list containing predefined status options.  
2. The dropdown list should be positioned appropriately on the user interface for easy access.  
3. The status options should be displayed in uppercase letters.  
4. The status field should be clearly labeled as "Status" and should be easily identifiable.  
5. The status field should be located within the "WEBSALES" tab of the user interface.  
6. The status field should have a white background and black text for readability.  
7. The status field should be bold and use the Tahoma font for consistency with other UI elements.  
  
Definition of Done:  
1. The status dropdown list is implemented and contains the predefined status options.  
2. The status field is correctly positioned and labeled on the user interface.  
3. The status options are displayed in uppercase letters.  
4. The status field is located within the "WEBSALES" tab.  
5. The status field has a white background and black text.  
6. The status field uses the Tahoma font and is bold.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include any specific SQL queries or database operations.

# Submit Comment via Button Click

Type: DEQC\_SAERCH

Detailed description: As a user, I want to be able to submit a comment through a button click, so that the system can process and store the comment based on specific conditions and statuses.  
  
Acceptance criteria:  
1. If the comment field is empty, the system should prompt the user to enter a comment and prevent further actions.  
2. The system should increment a comment count variable each time a comment is successfully submitted.  
3. Based on the status variable, the system should:  
 - Set the status to 'REJECT' and navigate to the reject section if the status is 'R'.  
 - Set the status to 'LINK/SAVE' and navigate to the link/save section if the status is 'LS'.  
 - Set the status to 'PROPOSAL\_INVOKED' and navigate to the QC section if the status is 'QC'.  
 - For any other status, the system should call a procedure to handle comments.  
4. The procedure should:  
 - Retrieve the contract ID based on the application number.  
 - Generate a new event number for the comment.  
 - Insert the comment into the comments table with relevant details.  
 - Commit the transaction.  
  
Definition of Done:  
- The button click functionality should be implemented and tested.  
- The system should correctly handle empty comment fields by prompting the user.  
- The comment count should increment correctly.  
- The system should navigate to the appropriate sections based on the status.  
- The procedure should correctly handle the retrieval of contract ID, generation of event number, insertion of comment, and committing the transaction.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Retrieve contract ID:  
 ```sql  
 SELECT CONT\_ID  
 INTO :control.contract\_id  
 FROM azbj\_batch\_items  
 WHERE APPLICATION\_NO = :deqc\_display.application\_number AND TRANSACTION\_TYPE='FRP';  
 ```  
- Generate new event number:  
 ```sql  
 SELECT NVL(MAX(event\_no) + 1, 1)  
 INTO v\_event\_no  
 FROM azbj\_uw\_comments  
 WHERE contract\_id = :control.contract\_id;  
 ```  
- Insert comment:  
 ```sql  
 INSERT INTO azbj\_uw\_comments  
 (event\_no, contract\_id,  
 policy\_no, move\_code,  
 policy\_status, user\_id, comment\_date,  
 comments, flag)  
 VALUES (v\_event\_no, :control.contract\_id,  
 :deqc\_display.proposal\_no, 'AZBJ\_WEB\_OTC',  
 azbj\_pkg\_var.v\_mst, USER, SYSDATE,  
 :deqc\_saerch.reason\_link, 'N');  
 ```  
- Commit transaction:  
 ```sql  
 COMMIT;  
 ```

# QC Button Functionality for DEQC Display

Type: DEQC\_SAERCH

Detailed description: As a user, I want to be able to select a case from the DEQC display and perform quality control (QC) actions on it, ensuring that only one case is selected at a time and that the selected case meets specific criteria before proceeding with further actions.  
  
Acceptance criteria:  
1. When the QC button is pressed, the system should navigate to the DEQC display block and count the number of selected cases.  
2. If more than one case is selected, the system should display an error message indicating that only one case can be selected and return the focus to the selection checkbox.  
3. The system should check if the selected case meets specific criteria, such as having a proposal status of 'PENDING\_FOR\_BBU' or 'PROPOSAL\_INVOKED' and no policy reference.  
4. If the selected case meets the criteria, the system should gather necessary parameters and call the appropriate form for further processing.  
5. If the selected case does not meet the criteria, the system should display a warning message indicating that the selected case is not allowed for DEQC.  
  
Definition of Done:  
- The QC button functionality is implemented and tested.  
- The system correctly identifies and handles the selection of multiple cases.  
- The system validates the selected case against the specified criteria.  
- Appropriate error and warning messages are displayed as needed.  
- The system successfully calls the appropriate form with the necessary parameters for further processing.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- The following SQL queries are used to validate and process the selected case:  
 ```sql  
 SELECT perm\_receipt\_no, proposal\_no, proposal\_status  
 INTO azbj\_pkg\_var.v\_receipt\_no, azbj\_pkg\_var.v\_policy\_ref, azbj\_pkg\_var.v\_proposal\_status  
 FROM azbj\_phub\_tracker  
 WHERE application\_no = :deqc\_display.application\_number;  
  
 SELECT TO\_NUMBER(product\_code)  
 INTO v\_product\_id  
 FROM azbj\_batch\_items  
 WHERE application\_no = :deqc\_display.application\_number  
 AND transaction\_type = 'FRP'  
 AND NVL(PRINT, 'X') <> 'C';  
  
 SELECT COUNT()  
 INTO v\_cnt  
 FROM azbj\_phub\_scrutiny\_prop  
 WHERE application\_no = :deqc\_display.application\_number;  
  
 INSERT INTO azbj\_landing\_form\_data (user\_id, appln\_no, start\_time, flag)  
 VALUES (USER, :deqc\_display.application\_number, SYSDATE, 'DEQC');  
  
 SELECT COUNT()  
 INTO v\_qc\_cnt  
 FROM azbj\_system\_constants  
 WHERE sys\_type = 'QC' AND sys\_code = 'QC\_RIGHTS';  
 ```

# Save Application Details and Update Status

Type: DEQC\_SAERCH

Title: Save Application Details and Update Status  
  
Acceptance Criteria:  
1. When the "Save" button is pressed, the system should check if the reason for the link is provided and if the comment count is greater than zero. If either condition is not met, the user should be prompted to enter comments.  
2. The system should navigate to the 'DEQC\_DISPLAY' section and process each record.  
3. For each record marked with 'Y':  
 - Retrieve the contract ID for the application number from the `azbj\_batch\_items` table.  
 - Call the `azbj\_auto\_issuance\_prc` procedure to issue the policy.  
 - If the policy number is generated, display a success message.  
 - Check for rule errors in the `bbu\_trans\_dtls` table and set the rule configuration flag accordingly.  
 - Check if all required documents are received.  
 - If there are no rule errors and all documents are received, update the status to 'PENDING\_FOR\_AUTO\_BBU' in the `wip\_policy\_versions`, `azbj\_phub\_tracker`, and `bbu\_trans` tables.  
 - If there are rule errors or documents are missing, update the status to 'FR-AR' in the `azbj\_phub\_tracker` and `wip\_policy\_versions` tables.  
4. Commit the changes to the database.  
5. Display appropriate messages if any applications have failed rule validation or if data is saved successfully.  
  
Definition of Done:  
- The "Save" button functionality is implemented as described.  
- The system correctly validates the input and processes the records.  
- The status updates and database commits are performed as per the conditions.  
- Appropriate messages are displayed to the user based on the outcome.  
- The functionality is tested and verified to ensure it meets the acceptance criteria.  
  
DB queries for Table reference CRUD operations only(With Usage):  
```sql  
-- Retrieve contract ID  
SELECT cont\_id  
INTO :control.contract\_id  
FROM azbj\_batch\_items  
WHERE application\_no = :deqc\_display.application\_number  
AND transaction\_type = 'FRP' AND ROWNUM = 1;  
  
-- Check for rule errors  
SELECT COUNT()  
INTO v\_rule\_error\_count  
FROM bbu\_trans\_dtls  
WHERE trans\_id = v\_activity\_id AND action\_id = 2  
AND rule\_config\_id IS NOT NULL  
AND version\_no = (SELECT MAX(version\_no)  
 FROM bbu\_trans\_dtls  
 WHERE trans\_id = v\_activity\_id);  
  
-- Update status to 'PENDING\_FOR\_AUTO\_BBU'  
UPDATE wip\_policy\_versions  
SET change\_description = 'PENDING\_FOR\_AUTO\_BBU',  
 contract\_status = 'I'  
WHERE contract\_id = :control.contract\_id;  
  
UPDATE azbj\_phub\_tracker  
SET proposal\_modif\_user = USER,  
 proposal\_status = 'PENDING\_FOR\_AUTO\_BBU',  
 proposal\_modif\_date = SYSDATE,  
 locking\_flag = 'N'  
WHERE application\_no = :deqc\_display.application\_number;  
  
UPDATE bbu\_trans  
SET proposal\_no = NVL(v\_policy\_no, p\_data.policy\_ref),  
 contract\_id = :control.contract\_id,  
 user\_id = USER  
WHERE appl\_no = :deqc\_display.application\_number  
AND version\_no = (SELECT MAX(version\_no) FROM bbu\_trans  
 WHERE appl\_no = :deqc\_display.application\_number);  
  
-- Update status to 'FR-AR'  
UPDATE azbj\_phub\_tracker  
SET proposal\_status = 'FR-AR',  
 locking\_flag = 'N'  
WHERE application\_no = :deqc\_display.application\_number;  
  
UPDATE wip\_policy\_versions  
SET change\_description = 'FR-AR'  
WHERE contract\_id = :control.contract\_id;  
```

# Search Application Details and Document Status

Type: DEQC\_SAERCH

Detailed description: As a user, I want to search for application details based on various criteria such as application number, partner type, and receipt date range, so that I can view the relevant application information and document status.  
  
Acceptance criteria:  
1. The search functionality should allow filtering by application number, partner type, and receipt date range.  
2. The search results should display application details including application number, status, receipt number, receipt date, applicant name, policyholder name, partner name, and document status.  
3. The document status should be determined based on the presence of specific document types (BI, PF, RP, M017, M202) associated with the application.  
4. If all required documents are present, the status should be "Documents Received". If some documents are missing, the status should be "Documents Not Received". If no documents are present, the status should be "All Documents Pending".  
5. The search results should be displayed in a grid format, with each record showing the relevant details.  
6. If no records match the search criteria, an appropriate message should be displayed to the user.  
  
Definition of Done:  
1. The search functionality is implemented and integrated into the application.  
2. The search results are displayed correctly based on the provided criteria.  
3. The document status logic is correctly implemented and displays the appropriate status.  
4. The user interface is tested and verified to ensure it meets the acceptance criteria.  
5. Any errors or exceptions during the search process are handled gracefully and appropriate messages are displayed to the user.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- The following SQL queries are used to fetch the application details and document counts:  
  
```sql  
SELECT apt.application\_no appno, apt.proposal\_status appstatus,  
 abi.perm\_receipt\_no receiptno, abi.perm\_receipt\_date receiptdate,  
 apad.ip\_title || ' ' || apad.ip\_first\_name || ' ' || apad.ip\_middle\_name || ' ' || apad.ip\_last\_name laname,  
 apad.ph\_title || ' ' || apad.ph\_first\_name || ' ' || apad.ph\_middle\_name || ' ' || apad.ph\_last\_name phname,  
 azsc.sys\_desc partnername, apt.received\_user recuser  
FROM azbj\_batch\_items abi, azbj\_phub\_tracker apt,  
 azbj\_proposal\_appln\_det apad, azbj\_system\_constants azsc  
WHERE apt.application\_no = NVL (:deqc\_saerch.application\_no, apt.application\_no)  
 AND abi.agent\_code = NVL (TRIM (:deqc\_saerch.partner\_type), abi.agent\_code)  
 AND abi.perm\_receipt\_date BETWEEN NVL (:deqc\_saerch.from\_date, abi.perm\_receipt\_date)  
 AND NVL (:deqc\_saerch.TO\_DATE, abi.perm\_receipt\_date)  
 AND To\_Number(apt.application\_no) = apad.appln\_no  
 AND apad.appln\_no = To\_Number(abi.application\_no)  
 AND apt.application\_no = abi.application\_no  
 AND apt.agent\_code = apad.agent\_code  
 AND apt.agent\_code = abi.agent\_code  
 AND apad.agent\_code = abi.agent\_code  
 AND abi.agent\_code = azsc.char\_value  
 AND apt.agent\_code = azsc.char\_value  
 AND apad.agent\_code = azsc.char\_value  
 AND apt.perm\_receipt\_no = abi.perm\_receipt\_no  
 AND apt.perm\_receipt\_no IS NOT NULL  
 AND abi.perm\_receipt\_no IS NOT NULL  
 AND azsc.sys\_type = 'OTC'  
 AND azsc.sys\_code = 'OTC\_WEB\_PARTNERS'  
 AND de\_flag = 'D2'  
 AND proposal\_status IN ('PENDING\_FOR\_BBU', 'PROPOSAL\_INVOKED', 'PROPOSAL\_UPDATED')  
 AND apt.proposal\_no IS NULL;  
  
SELECT COUNT (0)  
INTO v\_status\_cnt  
FROM azbj\_cq\_doc\_upload\_dtls  
WHERE application\_no = TO\_CHAR (i.appno)  
 AND UPPER (doc\_type) NOT IN ('BI', 'PF');  
  
SELECT COUNT (0)  
INTO v\_bidoc\_cnt  
FROM azbj\_cq\_doc\_upload\_dtls  
WHERE application\_no = TO\_CHAR (i.appno)  
 AND UPPER (doc\_type) = 'BI';  
  
SELECT COUNT (0)  
INTO v\_pfdoc\_cnt  
FROM azbj\_cq\_doc\_upload\_dtls  
WHERE application\_no = TO\_CHAR (i.appno)  
 AND UPPER (doc\_type) = 'PF';  
  
SELECT COUNT (0)  
INTO v\_photodoc\_cnt  
FROM azbj\_cq\_doc\_upload\_dtls  
WHERE application\_no = TO\_CHAR (i.appno)  
 AND UPPER (doc\_type) = 'RP';  
  
SELECT COUNT (0)  
INTO v\_ageproofdoc\_cnt  
FROM azbj\_cq\_doc\_upload\_dtls  
WHERE application\_no = TO\_CHAR (i.appno)  
 AND UPPER (doc\_type) = 'M017';  
  
SELECT COUNT (0)  
INTO v\_addproofdoc\_cnt  
FROM azbj\_cq\_doc\_upload\_dtls  
WHERE application\_no = TO\_CHAR (i.appno)  
 AND UPPER (doc\_type) = 'M202';  
```

# Manage and Verify PAN Card Details

Type: CONTROL

Title: Manage and Verify PAN Card Details  
  
Acceptance Criteria:  
1. The system should display the current date and user information on the toolbar.  
2. The system should provide a button to exit the form.  
3. The system should have hidden fields for `PRODUCT\_ID` and `CONTRACT\_ID`.  
4. The system should display an editable field for the address, highlighted in yellow, indicating if it has been edited.  
5. The system should provide a button to save bank details.  
6. The system should allow users to enter and update their first, middle, and last names, ensuring the text is in uppercase.  
7. The system should display the status of the PAN card.  
8. The system should provide a radio group for PAN card verification, defaulting to "Y".  
9. The system should provide a button to validate the entered PAN card details.  
10. The system should have a list item for phone numbers associated with the PAN card, which should be disabled and hidden by default.  
11. The system should allow users to enter the PAN card number, ensuring the text is in uppercase.  
12. The system should provide a list item for partner type.  
13. The system should provide a checkbox to indicate if the PAN card is not available.  
14. The system should provide a checkbox for RRB bank account, which should be disabled and hidden by default.  
15. The system should provide a button to view previous policy PAN details.  
16. The system should allow users to enter and update their date of birth, ensuring the text is in uppercase and formatted as DD/MM/RRRR.  
  
Definition of Done:  
- All fields and buttons are implemented as per the acceptance criteria.  
- The system accurately captures and displays the required information.  
- The system performs validation checks where necessary.  
- The user interface is intuitive and user-friendly.  
- The functionality is tested and verified to be working correctly.  
  
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 Exit Button Functionality

Type: CONTROL

Title: Implement Exit Button Functionality  
  
Acceptance Criteria:  
- When the user presses the "Exit" button, the current form should close immediately.  
- The button should be visually identifiable with a label and an icon.  
- The button should be located at a specific position on the toolbar for easy access.  
  
Definition of Done:  
- The "Exit" button is implemented and visible on the toolbar.  
- Pressing the "Exit" button successfully closes the current form.  
- The button has the specified visual properties (icon, label, color, size, and position).  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include any database CRUD operations.

# Validate 'Partner Type' Field

Type: CONTROL

Title: Validate 'Partner Type' Field  
  
Acceptance Criteria:  
1. When the user attempts to submit the form, the system should check if the 'Partner Type' field is empty.  
2. If the 'Partner Type' field is empty, the system should display a warning message: 'Please enter partner.'  
  
Definition of Done:  
1. The 'Partner Type' field validation is implemented.  
2. The warning message is displayed correctly when the field is empty.  
3. The form cannot be submitted if the 'Partner Type' field is empty.  
4. The functionality is tested and verified to work as expected.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as there are no direct database operations mentioned in the provided XML content.

# View Previous Policy Details

Type: CONTROL

Title: View Previous Policy Details  
  
Acceptance Criteria:  
1. When the button is clicked, the system should check if a parameter list named 'Param1' already exists.  
2. If 'Param1' exists, it should be destroyed and a new parameter list named 'Param1' should be created.  
3. The system should add the following parameters to the parameter list:  
 - 'PAR\_PH\_PART\_ID' with the value of the current policyholder's part ID.  
 - 'PAR\_PAN\_CARD\_NO' with the value of the current policyholder's PAN card number.  
 - 'PAR\_MODULE' with the value 'BBU'.  
 - 'PAR\_PAN\_PH\_NAME' with the value of the policyholder's name.  
 - 'PAR\_PAN\_PH\_DOB' with the value of the policyholder's date of birth.  
4. The system should then call a form named 'AZBJ\_OLD\_POLICY\_DTLS' with the created parameter list.  
5. After the form is called, the system should set a variable indicating that the PAN has been validated to 'Y'.  
  
Definition of Done:  
- The button should be visible and enabled for user interaction.  
- Clicking the button should trigger the described sequence of actions.  
- The form 'AZBJ\_OLD\_POLICY\_DTLS' should open with the provided parameters.  
- The variable indicating PAN validation should be updated to 'Y' after the form is called.  
- The functionality should be tested and verified to ensure it works as expected.

# Validate PH\_NO\_PAN\_LOV List Item Change

Type: CONTROL

Title: Validate PH\_NO\_PAN\_LOV List Item Change  
  
Acceptance Criteria:  
1. When the "PH\_NO\_PAN\_LOV" list item is changed:  
 - If the "ph\_no\_pan\_card" field is set to 'Y' (Yes), the system should check if a value is selected in the "PH\_NO\_PAN\_LOV" list.  
 - If no value is selected in the "PH\_NO\_PAN\_LOV" list, the system should display an error message: "Please select one of the LOV for pan card not received flag."  
2. The system should handle exceptions gracefully:  
 - If a form trigger failure occurs, the system should raise the appropriate error.  
 - For any other exceptions, the system should raise the appropriate error.  
  
Definition of Done:  
- The "PH\_NO\_PAN\_LOV" list item change functionality is implemented and tested.  
- The system correctly validates the selection based on the specified conditions.  
- Appropriate error messages are displayed when conditions are not met.  
- Exception handling is implemented as specified.  
- 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 CRUD operations or SQL queries.

# Verify PAN Card Status During Quality Check

Type: CONTROL

Title: Verify PAN Card Status During Quality Check  
  
Acceptance Criteria:  
1. When the PAN card verification status is set to 'N' (Not Verified), the system should display an error message.  
2. The error message should indicate that the PAN card has been received but not verified.  
3. The user should be instructed to raise the FRAR (Further Review and Action Required) again if the PAN card is not verified.  
  
Definition of Done:  
- The system correctly identifies when the PAN card verification status is 'N'.  
- An appropriate error message is displayed to the user when the PAN card is not verified.  
- The user is instructed to raise the FRAR again if the PAN card is not verified.  
- The functionality is tested and verified to ensure it works as expected.

# Manage PAN Card Fields Based on Checkbox Selection

Type: CONTROL

Detailed description: As a user, I want to manage the visibility and enablement of the PAN card-related fields based on the selection of a checkbox indicating the availability of a PAN card, so that I can ensure the correct fields are displayed and editable based on the user's input.  
  
Acceptance criteria:  
1. When the checkbox for "Pan Card not Available" is checked:  
 - The field for selecting PAN card options (PH\_NO\_PAN\_LOV) should become visible and enabled.  
 - The PAN card input field (PAN\_CARD) should be disabled.  
 - The PAN card input field and related fields (first name, middle name, last name, and status) should be cleared.  
  
2. When the checkbox for "Pan Card not Available" is unchecked:  
 - The field for selecting PAN card options (PH\_NO\_PAN\_LOV) should become hidden and its value should be cleared.  
 - The PAN card input field (PAN\_CARD) should be enabled.  
  
Definition of Done:  
- The functionality should be implemented and tested to ensure that the visibility and enablement of the fields behave as described in the acceptance criteria.  
- The user interface should reflect the changes immediately upon checking or unchecking the checkbox.  
- All related fields should be cleared or reset as specified when the checkbox state changes.  
- The implementation should be independent of any specific technology or platform.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided logic does not involve direct database CRUD operations.

# Save Bank Details Functionality

Type: CONTROL

Detailed description: As a user, I want to save bank details associated with a specific policy reference, so that the bank information is accurately stored in the system.  
  
Acceptance criteria:  
1. When the "SAVE BANK DETAILS" button is pressed, the system should check if there are existing bank details for the given policy reference.  
2. If existing bank details are found, they should be deleted.  
3. The system should then insert the new bank details into the `azbj\_account\_details` table with the following fields:  
 - POLICY\_REF  
 - CONTRACT\_ID  
 - PARTNER\_ID  
 - ACCOUNT\_NO  
 - COLL\_BRANCH  
 - IFSC\_CODE  
 - ACC\_HOLDER\_NAME  
 - BANK\_NAME  
 - PAYEE\_RELATION  
 - MICR  
 - TIME\_STAMP  
 - userid  
 - Pay\_mode  
 - ACC\_TPP\_RELATION  
 - RRB\_BANK\_ACCOUNT  
 - ip\_rel\_with\_pp  
4. If any error occurs during the process, an error message should be displayed.  
  
Definition of Done:  
- The "SAVE BANK DETAILS" button functionality is implemented.  
- The system correctly checks for existing bank details and deletes them if found.  
- New bank details are successfully inserted into the `azbj\_account\_details` table.  
- Appropriate error handling is in place, and error messages are displayed when necessary.  
- The feature is tested and verified to work as expected.  
  
DB queries for Table reference CRUD operations only(With Usage):  
```sql  
-- Check for existing bank details  
SELECT COUNT()  
INTO v\_present  
FROM azbj\_account\_details  
WHERE policy\_ref = :DTLS\_BLK.proposal\_no;  
  
-- Delete existing bank details if found  
DELETE FROM azbj\_account\_details  
WHERE policy\_ref = :DTLS\_BLK.proposal\_no;  
  
-- Insert new bank details  
INSERT INTO azbj\_account\_details (  
 POLICY\_REF,  
 CONTRACT\_ID,  
 PARTNER\_ID,  
 ACCOUNT\_NO,  
 COLL\_BRANCH,  
 IFSC\_CODE,  
 ACC\_HOLDER\_NAME,  
 BANK\_NAME,  
 PAYEE\_RELATION,  
 MICR,  
 TIME\_STAMP,  
 userid,  
 Pay\_mode,  
 ACC\_TPP\_RELATION,  
 RRB\_BANK\_ACCOUNT,  
 ip\_rel\_with\_pp  
) VALUES (  
 :DTLS\_BLK.proposal\_no,  
 :control.contract\_id,  
 pk\_vars.ip\_part\_id,  
 :ACCOUNT\_DET.ACC\_NO,  
 :ACCOUNT\_DET.ACC\_BRANCH,  
 :ACCOUNT\_DET.IFSC\_CODE,  
 :ACCOUNT\_DET.ACC\_HOLDER\_NAME,  
 :ACCOUNT\_DET.BANK\_NAME,  
 :ACCOUNT\_DET.ACC\_RELATION,  
 :ACCOUNT\_DET.MICR,  
 SYSDATE,  
 USER,  
 v\_pay\_mode,  
 :ACCOUNT\_DET.ACC\_TPP\_RELATION,  
 :CONTROL.RRB\_BANK\_ACCOUNT,  
 :ACCOUNT\_DET.ip\_rel\_with\_pp  
);  
```

# PAN Card Number Entry and Validation

Type: CONTROL

Detailed description: As a user, I want to be able to enter a PAN Card number in the system and have it automatically retrieve and display the associated first name, middle name, surname, and date of birth from the database. Additionally, the system should validate the PAN Card number against previous policies and provide appropriate feedback.  
  
Acceptance criteria:  
1. When a PAN Card number is entered, the system should:  
 - Retrieve the first name, middle name, surname, and date of birth from the `cp\_partners` table based on the `part\_id`.  
 - The `part\_id` should be determined based on the value of `ip\_ph` (either `ip\_part\_id` or `ph\_part\_id`).  
  
2. If no data is found for the entered PAN Card number, the system should handle the exception gracefully without crashing.  
  
3. The system should validate the entered PAN Card number against previous policies:  
 - Check if the PAN Card number exists in the `ocp\_policy\_bases`, `ocp\_interested\_parties`, and `cp\_partners` tables.  
 - Ensure the policy is active (`top\_indicator = 'Y'` and `action\_code <> 'D'`).  
 - Exclude specific PAN Card numbers (`'AG/NRI/60A'`, `'AG/NRI/61A'`).  
 - If the PAN Card number is found in previous policies, enable the "Previous Policy Details" field and set a validation flag to 'N'.  
 - If the PAN Card number is not found, disable the "Previous Policy Details" field and set the validation flag to 'Y'.  
  
Definition of Done:  
- The system retrieves and displays the correct first name, middle name, surname, and date of birth based on the entered PAN Card number.  
- The system validates the PAN Card number against previous policies and provides appropriate feedback.  
- The "Previous Policy Details" field is enabled or disabled based on the validation result.  
- The system handles exceptions gracefully without crashing.  
  
DB queries for Table reference CRUD operations only(With Usage):  
```sql  
-- Query to retrieve user details based on PAN Card number  
SELECT first\_name, middle\_name, surname, DATE\_OF\_BIRTH  
FROM cp\_partners  
WHERE part\_id = CASE WHEN :CONTROL.ip\_ph = 'IP' THEN pk\_vars.ip\_part\_id  
 ELSE pk\_vars.ph\_part\_id  
 END;  
  
-- Query to validate PAN Card number against previous policies  
SELECT COUNT(1)  
INTO v\_pan\_cnt  
FROM ocp\_policy\_bases a, ocp\_interested\_parties b, cp\_partners c  
WHERE a.contract\_id = b.contract\_id  
 AND b.partner\_id = c.part\_id  
 AND c.tax\_id = :control.pan\_card  
 AND a.top\_indicator = 'Y'  
 AND c.part\_id <> pk\_vars.ph\_part\_id  
 AND a.action\_code <> 'D'  
 AND b.top\_indicator = 'Y'  
 AND b.action\_code <> 'D'  
 AND c.tax\_id NOT IN ('AG/NRI/60A', 'AG/NRI/61A')  
 AND rownum = 1;  
```

# Validate PAN Card Details

Type: CONTROL

Title: Validate PAN Card Details  
  
Acceptance Criteria:  
1. The system should prompt the user to enter the PAN card number if it is not provided.  
2. The system should validate that the PAN card number is exactly 10 characters long.  
3. The system should prompt the user to select the partner type if it is not provided.  
4. The system should prompt the user to enter the first name and date of birth as per the PAN card if they are not provided.  
5. The system should calculate the age based on the date of birth provided.  
6. The system should validate the PAN card number format using a regular expression.  
7. The system should prompt the user to confirm the PAN details before proceeding.  
8. The system should call an external API to validate the PAN card details and log the response.  
9. The system should display appropriate messages based on the validation results, such as invalid PAN number, name matching percentage, etc.  
10. The system should save the PAN details if the validation is successful.  
  
Definition of Done:  
- The user is prompted to enter the PAN card number if it is missing.  
- The user is prompted to select the partner type if it is missing.  
- The user is prompted to enter the first name and date of birth as per the PAN card if they are missing.  
- The system validates the PAN card number format and length.  
- The system calculates the age based on the date of birth provided.  
- The system prompts the user to confirm the PAN details before proceeding.  
- The system calls an external API to validate the PAN card details and logs the response.  
- The system displays appropriate messages based on the validation results.  
- The system saves the PAN details if the validation is successful.  
  
DB queries for Table reference CRUD operations only (With Usage):  
```sql  
-- Query to fetch date of birth from azbj\_proposal\_appln\_det table  
SELECT ip\_dob, ph\_dob  
FROM azbj\_proposal\_appln\_det  
WHERE appln\_no = :DTLS\_BLK.APPLN\_NO  
 AND de\_flag = 'D2';  
  
-- Fallback query to fetch date of birth if the first query fails  
SELECT ip\_dob, ph\_dob  
FROM azbj\_proposal\_appln\_det  
WHERE appln\_no = :DTLS\_BLK.APPLN\_NO  
 AND de\_flag = 'D1';  
  
-- Query to validate PAN card number format  
SELECT 'Y'  
INTO v\_ver\_pan  
FROM DUAL  
WHERE REGEXP\_LIKE (UPPER(:CONTROL.pan\_card), '^[A-Z]{3}[C,P,H,F,A,T,B,L,J,G][A-Z][0-9]{4}[A-Z]');  
  
-- Query to fetch father name from azbj\_partner\_extn table  
SELECT father\_name  
INTO v\_father\_name  
FROM azbj\_partner\_extn  
WHERE part\_id = CASE WHEN :CONTROL.ip\_ph = 'IP' THEN pk\_vars.ip\_part\_id ELSE pk\_vars.ph\_part\_id END;  
  
-- Query to fetch PAN details from AZBJ\_PAN\_DTLS table  
FOR i IN (SELECT FROM AZBJ\_PAN\_DTLS WHERE pan\_seq\_no = v\_pan\_seq)  
LOOP  
 -- Process each record  
END LOOP;  
  
-- Query to fetch partner details from cp\_partners table  
SELECT TRIM(first\_name), TRIM(middle\_name), TRIM(surname)  
INTO v\_fname, v\_mname, v\_lname  
FROM cp\_partners  
WHERE part\_id = pk\_vars.ip\_part\_id;  
```

# Reject Applications from DEQC\_SEARCH Module

Type: DEQC\_SEARCH

Title: Reject Applications from DEQC\_SEARCH Module  
  
Acceptance Criteria:  
1. When the "Reject" button is pressed, the system should:  
 - Navigate to the DEQC\_DISPLAY module and count the number of records.  
 - Loop through each record and check if the 'ch' field is marked as 'Y'.  
 - If 'ch' is 'Y', append the application number to a list and increment a counter.  
 - If the 'reason\_link' field is empty or a specific package variable is zero, prompt the user to enter comments and halt the process.  
 - Call a procedure to auto-reject the application.  
 - Attempt to retrieve the contract ID from the azbj\_batch\_items table based on the application number and transaction type.  
 - If the 'reason\_link' field is not empty, insert a new comment into the azbj\_uw\_comments table with the appropriate details.  
 - Commit the transaction.  
 - Display a message indicating the number of records rejected and the list of rejected application numbers.  
  
Definition of Done:  
- The "Reject" button functionality is implemented and tested.  
- The system correctly navigates, processes records, and handles exceptions as described.  
- Appropriate messages are displayed to the user.  
- All database operations (select, insert) are performed correctly and committed.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Retrieve contract ID:  
 ```sql  
 SELECT CONT\_ID INTO :control.contract\_id   
 FROM azbj\_batch\_items   
 WHERE APPLICATION\_NO = :deqc\_display.application\_number   
 AND TRANSACTION\_TYPE = 'FRP';  
 ```  
  
- Insert new comment:  
 ```sql  
 INSERT INTO azbj\_uw\_comments  
 (event\_no, contract\_id, policy\_no, move\_code, policy\_status, user\_id, comment\_date, comments, flag)  
 VALUES (v\_event\_no, :control.contract\_id, :deqc\_display.proposal\_no, 'AZBJ\_WEB\_OTC', azbj\_pkg\_var.v\_mst, USER, SYSDATE, :deqc\_saerch.reason\_link, 'N');  
 ```