User Stories

# Dynamic Field Visibility and State Adjustment

Type: MODULE

Title: Dynamic Field Visibility and State Adjustment  
  
Acceptance Criteria:  
1. When the tab page changes, the system should:  
 - Retrieve the name of the topmost tab page.  
 - Adjust the visibility and enabled state of the `CH\_GMI` field in the `COVERHEAD` section based on the product ID.  
 - Calculate and set the `ch\_sum\_assured` field in the `COVERHEAD` section based on the product ID and booking frequency.  
 - Disable the `MU\_ADD\_TEST` field in the `MED\_UW` section if the product ID is 309.  
 - Validate and set the `partner\_id` in the `insured\_person` section.  
 - Pass parameters to the `medical\_uw` form and call it.  
 - Populate the `further\_req` section with test codes and descriptions from the `AZBJ\_UW\_REQUIREMENTS\_RAISED` and `azbj\_system\_constants` tables.  
 - Populate the `MED\_UW` section with test codes and descriptions from the `AZBJ\_UW\_REQUIREMENTS\_RAISED` and `azbj\_system\_constants` tables.  
 - Adjust the visibility and enabled state of the `IP\_PAN\_ISSUANCE\_DATE` and `IP\_PAN\_ISSUACE\_DATE\_NA` fields in the `INSURED\_PERSON` section based on the age proof.  
 - Adjust the visibility and enabled state of the `PH\_PAN\_ISSUANCE\_DATE` and `PH\_PAN\_ISSUANCE\_DATE\_NA` fields in the `POLICY\_HOLDER` section based on the age proof.  
 - Adjust the visibility and enabled state of the `st\_customer` field in the `AGENTS` section based on the mailing state.  
 - Populate the `PH\_RELATION` list in the `POLICY\_HOLDER` section based on the existence of EIA.  
  
Definition of Done:  
- The system dynamically adjusts the visibility and enabled state of fields based on the product ID and other conditions.  
- The `medical\_uw` form is called with the correct parameters.  
- The `further\_req` and `MED\_UW` sections are populated with the correct test codes and descriptions.  
- The visibility and enabled state of fields in the `INSURED\_PERSON`, `POLICY\_HOLDER`, and `AGENTS` sections are correctly adjusted based on the specified conditions.  
- The `PH\_RELATION` list in the `POLICY\_HOLDER` section is correctly populated based on the existence of EIA.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- No direct database queries are provided in the XML content. The logic involves retrieving and setting values based on conditions and calling forms with parameters.

# Exit Button Functionality

Type: MODULE

Title: Exit Button Functionality  
  
Acceptance Criteria:  
1. When the "Exit" button is pressed, the system should hide the current view.  
2. The system should also hide the current window associated with the view.  
3. The system should then navigate to the main view.  
  
Definition of Done:  
- The "Exit" button is functional and performs the actions as described in the acceptance criteria.  
- The current view and window are hidden upon pressing the "Exit" button.  
- The user is navigated back to the main view.  
- 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 XML content does not include any database CRUD operations.

# Display and Manage Underwriting Comments in Case Summary

Type: SUMMARY\_UW\_DTLS

Title: Display and Manage Underwriting Comments in Case Summary  
  
Acceptance Criteria:  
1. The system should display underwriting comments, user IDs, and comment dates in the case summary section.  
2. Underwriting comments should have a maximum length of 500 characters.  
3. The user ID and comment date fields should be display-only and not editable by the user.  
4. The comment date should be formatted as "dd-mon-yy".  
5. The fields should be hidden from view by default.  
  
Definition of Done:  
1. The underwriting comments, user IDs, and comment dates are displayed correctly in the case summary section.  
2. The underwriting comments field allows up to 500 characters.  
3. The user ID and comment date fields are display-only and not editable.  
4. The comment date is formatted as "dd-mon-yy".  
5. The fields are hidden from view by default and can be made visible as needed.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include any specific database queries.

# Manage Unit-Linked Fund Portfolio

Type: UL\_AT\_PORTFOL

Title: Manage Unit-Linked Fund Portfolio  
  
Acceptance Criteria:  
1. The user should be able to view a list of funds in the portfolio.  
2. The user should be able to enter the percentage allocation for each fund.  
3. The system should automatically calculate and display the total fund value based on the entered percentages.  
4. The user should be able to mark a fund for deletion.  
5. The system should initialize the percentage allocation to 0 for new entries.  
6. The system should ensure that the total fund value is updated whenever the percentage allocation is modified.  
  
Definition of Done:  
1. The user interface displays a list of funds with their respective percentage allocations.  
2. The user can input and modify the percentage allocation for each fund.  
3. The total fund value is correctly calculated and displayed based on the entered percentages.  
4. The user can mark funds for deletion, and these funds are visually indicated as such.  
5. New fund entries have their percentage allocation initialized to 0.  
6. The system updates the total fund value dynamically as percentage allocations are changed.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include specific database queries.

# Manage Unit-Linked Fund Portfolio

Type: UL\_AT\_PORTFOL\_2

Title: Manage Unit-Linked Fund Portfolio  
  
Acceptance Criteria:  
1. The user should be able to view a list of unit-linked funds.  
2. The user should be able to add a new fund to the portfolio.  
3. The user should be able to update the details of an existing fund.  
4. The user should be able to delete a fund from the portfolio.  
5. The system should display the total value of the funds in the portfolio.  
6. The percentage allocation of each fund should be displayed and editable.  
7. The system should ensure that the total percentage allocation does not exceed 100%.  
  
Definition of Done:  
1. The functionality to view, add, update, and delete funds is implemented and tested.  
2. The total value of the funds is correctly calculated and displayed.  
3. The percentage allocation is editable and validated.  
4. The user interface is intuitive and user-friendly.  
5. All acceptance criteria are met and verified through testing.  
6. 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 SQL queries or table references.

# View and Manage Member Comments in Reinsurer Selection Screen

Type: RI\_COMMENTS\_MEMBERWISE

Detailed description: As a user, I want to view and manage comments related to individual members within a reinsurer selection screen, so that I can efficiently track and update relevant information.  
  
Acceptance criteria:  
1. The system should display a section for member comments with the following fields:  
 - Partner ID (numeric)  
 - IP Number (numeric)  
 - Member Name (text, up to 150 characters)  
 - RI Date (date)  
 - RI Comments (text, up to 500 characters)  
2. The section should display up to 3 records at a time.  
3. The fields should be read-only, meaning users cannot insert or update the information directly in this section.  
4. The section should include a scrollbar for navigating through the records.  
  
Definition of Done:  
1. The section for member comments is implemented and displays the specified fields.  
2. The fields are read-only and display up to 3 records at a time.  
3. A scrollbar is present and functional for navigating through the records.  
4. The section is integrated into the reinsurer selection screen and is visually 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 any specific database queries or operations.

# Manage Receipt Information and Perform Transaction Actions

Type: AZBJ\_TOPUP

Detailed description: As a user, I want to manage receipt information and perform various actions related to transactions in the system. This includes viewing and entering receipt details, calculating the total amount, and executing specific actions such as service requests, rejections, and handling suspected documents.  
  
Acceptance criteria:  
1. The user should be able to view and enter the receipt number and receipt date.  
2. The user should be able to view the amount associated with each transaction.  
3. The system should automatically calculate and display the sum of all amounts.  
4. The user should be able to perform actions such as:  
 - Initiating a service request.  
 - Rejecting a transaction.  
 - Marking a document as suspected.  
 - Handling scrutiny failure cases.  
 - Viewing re-insurance details.  
 - Managing simultaneous medicals.  
 - Viewing previous fake document details for insurance companies.  
  
Definition of Done:  
1. The receipt number and receipt date fields are functional and allow data entry.  
2. The amount field displays the correct value for each transaction.  
3. The total amount is correctly calculated and displayed.  
4. All action buttons (Service Request, Reject, Suspected Document, Scrutiny Failure Case, Re-Insurance Details, Simultaneous Medicals, Previous Fake Doc Details) are functional and trigger the appropriate actions.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Not applicable as the provided XML content does not include specific database queries or operations.

# Simultaneous Medical Data Processing Button

Type: AZBJ\_TOPUP

Title: Simultaneous Medical Data Processing Button  
  
Acceptance Criteria:  
1. When the "Simultaneous Medicals" button is pressed, the system should execute the `azbj\_simultaneous\_medical\_data` procedure.  
2. The button should be labeled "Simultaneous Medicals" and be located on the "MED\_UW" tab page.  
3. The button should have a gray background, black foreground, and use the "Tahoma" font in bold style.  
4. The button should be positioned at coordinates (501, 764) with a width of 186 and a height of 20.  
  
Definition of Done:  
- The "Simultaneous Medicals" button is visible and correctly positioned on the "MED\_UW" tab page.  
- Pressing the button triggers the `azbj\_simultaneous\_medical\_data` procedure.  
- The button's appearance matches the specified design attributes (color, font, size, etc.).  
  
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 table references.

# Manage Reinsurance Details

Type: AZBJ\_TOPUP

Detailed description: As a user, I want to access and manage reinsurance details for a policy, so that I can ensure the correct reinsurance type and details are recorded and processed.  
  
Acceptance criteria:  
1. When the "Re-Insurance Details" button is pressed, the system should:  
 - Set the proposal number in the reinsurance details to match the policy reference.  
 - Set the sum assured in the reinsurance details to match the cover head sum assured.  
 - Navigate to the reinsurance block and clear any existing records.  
 - Retrieve and display the first record in the reinsurance block.  
 - Execute the reinsurance logic to populate the reinsurance details.  
 - If the solution name in the cover head is between 1 and 2, execute additional reinsurance solution logic.  
 - Navigate to the reinsurance reference number field and display the first record.  
  
Definition of Done:  
- The reinsurance details are correctly populated based on the policy reference and sum assured.  
- The reinsurance block is navigated to and cleared before displaying the first record.  
- The reinsurance logic is executed to ensure the correct reinsurance details are populated.  
- Additional reinsurance solution logic is executed if the solution name is between 1 and 2.  
- The reinsurance reference number field is navigated to and the first record is displayed.  
  
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.

# View Previous Fake Document Details

Type: AZBJ\_TOPUP

Detailed description: As a user, I want to view the details of previous fake documents associated with an intermediary who has a past history of forged documents, so that I can review and take necessary actions.  
  
Acceptance criteria:  
1. When the user presses the "Previous Fake Doc Dtls for IC" button, the system should fetch and display the following details for each relevant document:  
 - Application Number  
 - Document Description  
 - Document Type  
 - Document Category  
 - Policy Reference Number  
2. The system should iterate through all records that match the criteria and display them sequentially.  
3. The criteria for fetching records are:  
 - The intermediary's code matches the agent's code.  
 - The intermediary has a comment indicating a past history of forged documents.  
4. The system should handle any exceptions gracefully without crashing.  
  
Definition of Done:  
- The button is functional and retrieves the correct data based on the specified criteria.  
- The data is displayed in a user-friendly manner.  
- The system handles exceptions without crashing.  
- The feature is tested and verified to work as expected.  
  
DB queries for Table reference CRUD operations only(With Usage):  
```sql  
SELECT pt.application\_no,  
 fd.document\_desc,  
 fd.document\_type,  
 fd.category,  
 fd.policy\_ref  
 FROM azbj\_uw\_comments uw,  
 azbj\_fake\_document fd,  
 azbj\_phub\_tracker pt  
 WHERE uw.contract\_id = fd.contract\_id  
 AND pt.proposal\_no = fd.policy\_ref  
 AND uw.policy\_no = fd.policy\_ref  
 AND uw.comments = 'Intermediary with Past History- Of Forged Documents.'  
 AND ic\_code = :agents.ag\_agent\_code;  
```

# Manage Suspected Documents for a Contract

Type: AZBJ\_TOPUP

Title: Manage Suspected Documents for a Contract  
  
Acceptance Criteria:  
1. When the "Suspected Document" button is pressed, the system should check if there are any existing records for the given contract ID in the suspected documents table.  
2. If no records are found, the system should:  
 - Enable the "Save" button and disable the "Update" button.  
 - Populate the document details from the system constants table where the system type is 'FAKE\_DOC'.  
3. If records are found, the system should:  
 - Retrieve the user who inserted the document and check if the current user is a supervisor or the same user.  
 - Enable the "Update" button and disable the "Save" button.  
 - Populate the document details from the suspected documents table for the given contract ID.  
 - If the document is marked as "FAKE", check if the document type exists in the further requirements table.  
 - If the document type exists in the further requirements table, disable editing for specific fields.  
4. Display an error message if the document has already been verified by another user.  
  
Definition of Done:  
- The "Suspected Document" button functionality is implemented as per the acceptance criteria.  
- The system correctly enables/disables buttons based on the presence of records and user roles.  
- Document details are accurately populated from the respective tables.  
- Appropriate error messages are displayed when necessary.  
- The functionality is tested and verified to work as expected.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Select count of records from `azbj\_fake\_document` where `contract\_id` matches the given contract ID.  
- Select distinct system constants from `azbj\_system\_constants` where `sys\_type` is 'FAKE\_DOC'.  
- Select distinct user who inserted the document from `azbj\_fake\_document` where `contract\_id` matches the given contract ID and `document\_code` is not null.  
- Select document details from `azbj\_fake\_document` where `contract\_id` matches the given contract ID.  
- Check if the document type exists in the `further\_req` table for the given contract ID.

# Reject Service Request for Top-Up Transaction

Type: AZBJ\_TOPUP

Detailed description: As a user, I want to be able to reject a service request for a top-up transaction, ensuring that the request is properly validated and updated in the system.  
  
Acceptance criteria:  
1. If the total amount for the top-up is not selected, the system should display a warning message: "Please Select the topup receipt" and focus on the receipt number field.  
2. If the top-up has been approved and the user ID does not start with 'P00', the system should display an error message: "Cannot Reject the request".  
3. If the user ID starts with 'P00' or the top-up has not been approved and the user ID does not start with 'P00':  
 - The system should update the status and approval status of the service request to 'R' (Rejected) in the database.  
 - The system should retrieve the service request number associated with the policy reference.  
 - The system should update the active flag to 'N' for the top-up cover details associated with the retrieved service request number.  
 - The system should display a warning message: "Service Request Rejected" and focus on the receipt number field.  
4. If no data is found for the service request, the system should display a warning message: "Please check the receipt".  
  
Definition of Done:  
- The reject functionality should be implemented and tested.  
- All acceptance criteria should be met.  
- The system should handle exceptions and display appropriate messages.  
- The changes should be reviewed and approved.  
  
DB queries for Table reference CRUD operations only (With Usage):  
```sql  
-- Update the status and approval status of the service request  
UPDATE AZBJ\_TPS\_SERVICE\_REQ  
SET STATUS = 'R',  
 APPROVAL\_STATUS = 'R'  
WHERE POLICY\_REF = :CONTROL.CN\_POLICY\_REF;  
  
-- Retrieve the service request number associated with the policy reference  
SELECT SERVICE\_REQ\_NO  
INTO V\_SERVICE\_REQ\_NO  
FROM AZBJ\_TPS\_SERVICE\_REQ  
WHERE POLICY\_REF = :CONTROL.CN\_POLICY\_REF  
AND STATUS = 'R';  
  
-- Update the active flag for the top-up cover details  
UPDATE AZBJ\_TOPUP\_COVER\_DETAILS  
SET ACTIVE\_FLAG = 'N'  
WHERE SERVICE\_REQ\_NO = V\_SERVICE\_REQ\_NO;  
```

# Handle Scrutiny Failure Cases

Type: AZBJ\_TOPUP

Title: Handle Scrutiny Failure Cases  
  
Acceptance Criteria:  
1. If both the verification number and sign card number are missing, the system should display a warning message: "Please enter Verification or Sign Card No and then do the Scrutiny Failure case".  
2. If a scrutiny record is found for the given application number, the system should navigate to the scrutiny failure document section.  
3. If no scrutiny failure record exists for the given contract ID, the system should:  
 - Enable the save button and disable the update button.  
 - Populate the scrutiny failure block with system constants related to scrutiny documents.  
4. If a scrutiny failure record exists for the given contract ID, the system should:  
 - Enable the update button and disable the save button.  
 - Populate the scrutiny failure block with existing scrutiny failure records for the contract ID.  
5. If no scrutiny record is found for the given application number, the system should display a warning message: "Scrutiny Not Present For This Application no."  
  
Definition of Done:  
- The functionality to handle scrutiny failure cases is implemented and tested.  
- The system correctly displays warning messages when required.  
- The system navigates to the appropriate sections and updates the scrutiny failure documents as per the acceptance criteria.  
- All related database operations are performed correctly.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Query to check if scrutiny record exists:  
 ```sql  
 SELECT 'Y'  
 INTO v\_scrutiny\_present  
 FROM azbj\_phub\_scrutiny\_prop  
 WHERE APPLICATION\_NO = TO\_CHAR(NVL(:INSURED\_PERSON.IP\_VERF\_NO, :INSURED\_PERSON.IP\_SIGN\_CARD\_NO));  
 ```  
  
- Query to count existing scrutiny failure records:  
 ```sql  
 SELECT COUNT()  
 INTO v\_present  
 FROM azbj\_scrutiny\_fail  
 WHERE contract\_id = :control.cn\_contract\_id;  
 ```  
  
- Query to populate scrutiny failure block with system constants:  
 ```sql  
 SELECT SYS\_CODE, SYS\_DESC, CHAR\_VALUE  
 FROM azbj\_system\_constants  
 WHERE sys\_type = 'SCRUT\_DOC';  
 ```  
  
- Query to populate scrutiny failure block with existing records:  
 ```sql  
 SELECT document\_desc, document\_type, document\_code, document\_checked, comments, scrutiny\_fail\_document  
 FROM azbj\_scrutiny\_fail  
 WHERE contract\_id = :control.cn\_contract\_id;  
 ```

# Service Request Initiation

Type: AZBJ\_TOPUP

Detailed description: As a user, I want to be able to initiate a service request for a specific contract, so that I can manage and track service requests efficiently.  
  
Acceptance criteria:  
1. When the "Service Request" button is pressed, the system should check if there are any existing proposals for the given contract ID.  
2. If there are no existing proposals, the system should display a message prompting the user to save the proposal first.  
3. If there are existing proposals, the system should create a parameter list and add necessary parameters such as policy reference, receipt number, premium amount, and insured person's date of birth.  
4. The system should then check if the receipt number is null. If it is, the system should prompt the user to select a receipt number.  
5. If the receipt number is not null, the system should check if there is an existing service request for the given receipt number and contract ID.  
6. If no existing service request is found, the system should set the form status to "NEW" and call the service request form.  
7. If an existing service request is found, the system should set the form status to "UPDATE" and call the service request form with the existing service request number.  
8. The system should handle any errors by displaying an appropriate message.  
  
Definition of Done:  
- The "Service Request" button functionality is implemented as per the acceptance criteria.  
- The system correctly checks for existing proposals and service requests.  
- Appropriate messages are displayed to the user based on the conditions.  
- The service request form is called with the correct parameters.  
- Error handling is implemented and tested.  
  
DB queries for Table reference CRUD operations only(With Usage):  
```sql  
-- Check for existing proposals  
SELECT COUNT(0)  
INTO V\_COUNT  
FROM AZBJ\_SEARCH\_PROPOSAL  
WHERE CONTRACT\_ID = :CONTROL.CN\_CONTRACT\_ID;  
  
-- Check for existing service request  
SELECT A.SERVICE\_REQ\_NO  
INTO v\_service\_req\_no  
FROM azbj\_tps\_TOPUP A, AZBJ\_TPS\_SERVICE\_REQ B  
WHERE perm\_rEcpt\_no = :azbj\_topup.perm\_receipt\_no  
AND A.SERVICE\_REQ\_NO = B.SERVICE\_REQ\_NO  
AND STATUS <> 'R';  
```

# View and Manage Previous Medical Records

Type: PREVIOUS\_MEDICALS

Title: View and Manage Previous Medical Records  
  
Acceptance Criteria:  
1. The system should display a list of previous medical records with the following details:  
 - Serial Number (Sl )  
 - Policy Reference (Policy Ref)  
 - Test Number (Test No)  
 - Description  
 - Test Done Date  
 - Date Received  
 - Received (Y/N)  
 - Life  
 - Medical Test ID  
 - Partner ID  
 - Supervisor Comments  
 - Underwriting Reason  
 - Request Count  
2. The user should be able to navigate through the records using up and down keys.  
3. The user should be able to view images associated with the medical records by clicking a "View Images" button.  
  
Definition of Done:  
- The list of previous medical records is displayed with all the specified details.  
- Navigation through records using up and down keys is functional.  
- The "View Images" button is present and functional, allowing users to view associated images.  
- All fields are displayed in a read-only format except for the "Received (Y/N)" and "Date Received" fields, which are editable.  
- The user interface is intuitive and matches the design specifications.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include any direct database operations or queries.

# View Images for Policy Reference

Type: PREVIOUS\_MEDICALS

Title: View Images for Policy Reference  
  
Acceptance Criteria:  
1. When the "View Images" button is pressed, the system should check if the policy reference in the previous medical records is not null.  
2. If the policy reference is not null, the system should generate a URL using the policy reference and encrypt it.  
3. The system should then open the generated URL in a web browser to display the related images.  
4. If there is an error in generating or opening the URL, the system should display a warning message indicating the issue.  
  
Definition of Done:  
- The "View Images" button functionality is implemented and tested.  
- The system correctly checks for a non-null policy reference.  
- The URL is generated and encrypted properly.  
- The URL opens in a web browser to display the images.  
- Appropriate error handling and warning messages are in place.  
- 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 any direct CRUD operations.

# Manage Product-Related Information

Type: COMBO

Title: Manage Product-Related Information  
  
Acceptance Criteria:  
1. The system should display the following fields for product-related information:  
 - Product Name  
 - Name of Member  
 - Application No  
 - Date of Birth  
 - Package Name  
 - Premium  
 - Premium Term  
 - Benefit Term  
 - Sum Assured  
 - Multiplier  
 - Master Policy No.  
 - Loan Amount  
 - Loan Number  
 - Loan Tenure  
  
2. The "Package Name" field should have a dropdown list populated with distinct package codes from the database, filtered based on the selected child product ID.  
  
Definition of Done:  
- The user can view all the specified fields on the interface.  
- The "Package Name" dropdown is populated correctly based on the child product ID.  
- The fields are displayed in a user-friendly manner, with appropriate labels and formatting.  
- The system should not allow updates or inserts for fields that are marked as display-only.  
- The system should handle data types correctly, ensuring that date fields, numeric fields, and text fields are validated appropriately.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- The dropdown for "Package Name" should be populated using the following query:  
 ```sql  
 SELECT DISTINCT PACKAGE\_CODE   
 FROM azbj\_package\_master a, azbj\_combo\_products b   
 WHERE a.product\_id = b.SECONDARY\_PRODUCT   
 AND UPPER(b.SECONDARY\_PRODUCT) = :combo.child\_product\_id;  
 ```

# Validate Premium Field and Disable Toolbar Buttons

Type: COMBO

Title: Validate Premium Field and Disable Toolbar Buttons  
  
Acceptance Criteria:  
1. When the "Premium" field is validated:  
 - The "Commit Form" and "Exit Form" buttons on the horizontal toolbar should be disabled.  
 - If the global loading status is 'F', the form status control should be set to 'Y'.  
  
Definition of Done:  
- The "Premium" field validation logic is implemented.  
- The "Commit Form" and "Exit Form" buttons are disabled during the validation process.  
- The form status control is updated based on the global loading status.  
- The functionality is tested and confirmed to work as expected.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include any database-specific queries.

# Package Name Selection and Validation

Type: COMBO

Title: Package Name Selection and Validation  
  
Acceptance Criteria:  
1. When the user double-clicks on the package name field, a list of values (LOV) should be displayed, allowing the user to select a package name.  
2. The system should validate the selected package name to ensure it does not contain the word "SINGLE".  
3. The system should check if the product ID associated with the package name is valid by querying the `azbj\_system\_constants` table where `sys\_type` is 'GRP' and `sys\_code` is 'GRP\_COMBO'.  
4. If the package name is invalid or the product ID is not valid, the system should display an error message: "Please select valid package name" and return the focus to the package name field.  
  
Definition of Done:  
- The LOV for package names is displayed upon double-clicking the package name field.  
- The package name is validated according to the specified criteria.  
- An error message is shown if the package name is invalid, and the focus is returned to the package name field.  
- The functionality is tested and verified to work as expected.  
  
DB queries for Table reference CRUD operations only(With Usage):  
```sql  
-- Query to validate product ID  
SELECT COUNT()   
INTO v\_count   
FROM azbj\_system\_constants  
WHERE sys\_type = 'GRP'  
 AND sys\_code = 'GRP\_COMBO'  
 AND num\_value = :control.cn\_product\_id;  
  
-- Query for LOV  
SELECT DISTINCT PACKAGE\_CODE   
FROM azbj\_package\_master a, azbj\_combo\_products b   
WHERE a.product\_id = b.SECONDARY\_PRODUCT  
 AND UPPER(b.SECONDARY\_PRODUCT) = :combo.child\_product\_id;  
```

# Validate Master Policy Number

Type: COMBO

Title: Validate Master Policy Number  
  
Acceptance Criteria:  
1. When the Master Policy Number is entered, the system should:  
 - Retrieve the combo number from the `azbj\_phub\_scrutiny\_prop` table using the application number.  
 - Retrieve the child product ID from the `azbj\_combo\_products` table using the combo number.  
 - Check if the child product is a group product.  
 - Retrieve the scheme type from the `azbj\_master\_policy\_contract` table using the Master Policy Number.  
 - Verify that the scheme type and child product ID exist in the `azbj\_grp\_product\_master` table.  
 - Check if rates are uploaded for the Master Policy Number in the `azbj\_grp\_life\_prem\_rate` table.  
  
2. If any of the above validations fail, appropriate error messages should be displayed:  
 - "Error for child Product id in scrutiny"  
 - "Error for child Product id"  
 - "Child master policy no. is not valid"  
 - "Error in product master"  
 - "Please enter valid master policy no"  
 - "Rates are not uploaded for this Master policy no."  
  
Definition of Done:  
- The Master Policy Number validation logic is implemented and tested.  
- All error messages are displayed correctly based on the validation checks.  
- The system navigates back to the Master Policy Number field if the validation fails.  
- The validation logic is independent of any specific technology or Oracle Forms terminology.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Retrieve combo number:  
 ```sql  
 SELECT GSLCOMBO INTO v\_combo\_no  
 FROM azbj\_phub\_scrutiny\_prop  
 WHERE application\_no = TO\_CHAR(NVL(:insured\_person.ip\_verf\_no, :insured\_person.ip\_sign\_card\_no));  
 ```  
  
- Retrieve child product ID:  
 ```sql  
 SELECT SECONDARY\_PRODUCT INTO v\_child\_product\_id  
 FROM azbj\_combo\_products  
 WHERE combo\_no = v\_combo\_no;  
 ```  
  
- Retrieve scheme type:  
 ```sql  
 SELECT scheme\_type INTO v\_scheme\_type  
 FROM azbj\_master\_policy\_contract  
 WHERE master\_policy\_no = :combo.master\_policy\_no;  
 ```  
  
- Verify scheme type and child product ID:  
 ```sql  
 SELECT COUNT(1) INTO v\_count  
 FROM azbj\_grp\_product\_master  
 WHERE product\_id = v\_scheme\_type AND opus\_product\_code = v\_child\_product\_id;  
 ```  
  
- Check rates for Master Policy Number:  
 ```sql  
 SELECT COUNT() INTO v\_rate\_count  
 FROM azbj\_grp\_life\_prem\_rate  
 WHERE master\_policy\_no = :combo.master\_policy\_no;  
 ```

# View Case Summary Details

Type: CASE\_SUMMARY

Title: View Case Summary Details  
  
Acceptance Criteria:  
1. The case summary should include the following details:  
 - Gender  
 - Age  
 - Proposal Type  
 - Educational Qualification  
 - Occupation  
 - Frequency of payments  
 - Premium amount  
 - Riders Sum Assured  
 - Sum Assured  
 - Total Sum Assured (TASA)  
 - Success Indicator (SUC)  
2. The case summary should also include three text fields for additional case information.  
3. The user should be able to proceed with the case by clicking a "Proceed" button.  
4. The case summary should display the following financial review details:  
 - Financial Review Cycle  
 - Financial Review Freshness  
 - Financial Review Description  
  
Definition of Done:  
1. The case summary displays all the required demographic, proposal, educational, occupational, and financial details.  
2. The "Proceed" button is functional and allows the user to move forward with the case.  
3. The financial review details are visible and correctly populated.  
4. The user interface is intuitive and easy to navigate.  
5. All fields are correctly aligned and formatted for readability.  
6. The case summary is tested and verified to ensure all information is displayed accurately and the "Proceed" button works as expected.

# Proceed Button Functionality

Type: CASE\_SUMMARY

Title: Proceed Button Functionality  
  
Acceptance Criteria:  
1. When the "Proceed" button is pressed, the system should navigate to the item labeled 'Insured\_person.ip\_name'.  
2. The system should retrieve the name of the topmost tab page on the canvas named 'NBTABS'.  
3. The system should find the tab page ID using the retrieved tab page name.  
4. The system should retrieve the label of the identified tab page.  
5. If the identified tab page is visible, the system should set its visibility to false.  
6. The system should increment the variable `pk\_vars.v\_case\_summary\_ok` by 1.  
  
Definition of Done:  
- The "Proceed" button triggers the specified actions correctly.  
- The navigation to the item 'Insured\_person.ip\_name' is successful.  
- The tab page properties are retrieved and manipulated as described.  
- The visibility of the tab page is correctly updated based on its current state.  
- The counter `pk\_vars.v\_case\_summary\_ok` is incremented by 1 each time the button is pressed.  
  
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.

# Manage Allocation Details and Actions

Type: ALLOCATION

Title: Manage Allocation Details and Actions  
  
Acceptance Criteria:  
1. The system should display the following fields with appropriate labels:  
 - Main Allocation Percentage  
 - Promotional Allocation  
 - Topup Allocation  
 - Special Allocation End Date  
 - Campaign Code  
 - Campaign End Date  
 - Discount Type  
2. The system should allow the user to select a discount type from a list.  
3. The system should provide buttons for the following actions:  
 - Delete Fund  
 - Populate Allocation  
4. The system should display portfolio strategy information, which should be disabled for editing.  
5. The system should ensure that all fields and buttons are displayed on the "UNIT\_LINKED" tab page.  
  
Definition of Done:  
- All specified fields and buttons are displayed correctly on the "UNIT\_LINKED" tab page.  
- The user can view and interact with the fields and buttons as described in the acceptance criteria.  
- The portfolio strategy field is displayed but disabled for editing.  
- The user can perform the actions of deleting funds and populating allocations using the provided buttons.  
- The user interface is intuitive and user-friendly, with all elements properly aligned and labeled.  
  
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.

# Delete Fund Functionality

Type: ALLOCATION

Title: Delete Fund Functionality  
  
Acceptance Criteria:  
1. When the "Delete Fund" button is pressed, the system should navigate to the "SSO\_FUND" section.  
2. The system should then delete the current record from the "SSO\_FUND" section.  
  
Definition of Done:  
- The "Delete Fund" button is functional and performs the deletion of the fund record.  
- The system navigates to the "SSO\_FUND" section before performing the deletion.  
- The fund record is successfully removed from the database if it has not been approved.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- No specific DB queries provided in the XML content.

# Populate Allocation Button Functionality

Type: ALLOCATION

Title: Populate Allocation Button Functionality  
  
Acceptance Criteria:  
- When the "Populate Allocation" button is pressed, the system should execute the logic to check and update unit-linked allocations.  
- The button should be clearly labeled "Populate Allocation" and be easily accessible within the user interface.  
  
Definition of Done:  
- The "Populate Allocation" button is implemented and visible in the user interface.  
- Pressing the button triggers the logic to check and update unit-linked allocations.  
- The functionality is tested and verified to ensure that allocations are correctly populated and updated.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include specific database queries.

# Delete Fund Functionality in Auto Transfer Portfolio

Type: ALLOCATION

Title: Delete Fund Functionality in Auto Transfer Portfolio  
  
Acceptance Criteria:  
1. When the delete fund button is pressed, the system should check the total number of funds in the Auto Transfer portfolio.  
2. If there is only one fund in the portfolio, the system should display an error message: "Minimum one fund should be required under Auto Transfer!" and no deletion should occur.  
3. If there are multiple funds, the system should delete the selected fund(s) marked for deletion.  
4. The system should handle any exceptions gracefully without crashing.  
  
Definition of Done:  
1. The delete fund functionality is implemented and tested.  
2. The system correctly identifies the number of funds in the portfolio.  
3. The system displays the appropriate error message when attempting to delete the last remaining fund.  
4. The system successfully deletes the selected fund(s) when there are multiple funds.  
5. All exceptions are handled gracefully, ensuring the system remains stable.  
  
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.

# Delete Fund Functionality

Type: ALLOCATION

Detailed description: As a user, I want to be able to delete a fund from the Auto Transfer portfolio, ensuring that at least one fund remains in the portfolio.  
  
Acceptance criteria:  
1. When the "Delete Fund" button is pressed, the system should navigate to the "UL\_AT\_PORTFOL" section.  
2. The system should check the total number of records in the "UL\_AT\_PORTFOL" section.  
3. If there is only one fund in the portfolio, the system should display an error message: "Minimum one fund should be required under Auto Transfer!" and stop the deletion process.  
4. If there are multiple funds, the system should check each record to see if it is marked for deletion.  
5. If a record is marked for deletion and there are more than one fund, the system should delete the record.  
6. The system should continue this process until all records are checked.  
7. The system should handle any exceptions gracefully without crashing.  
  
Definition of Done:  
- The "Delete Fund" functionality works as described.  
- The error message is displayed correctly when there is only one fund.  
- Multiple funds can be deleted as long as at least one fund remains.  
- The system handles exceptions without crashing.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include specific database queries.

# Validate Employee Discount Type Selection

Type: ALLOCATION

Title: Validate Employee Discount Type Selection  
  
Acceptance Criteria:  
1. If the selected discount type is 'AB', 'BA', or 'BE' and the insured person's employee ID is null and the employee code variable is null, then the system should prompt a warning message: "Please Enter Employee Number for the Insured Person" and reset the discount type selection.  
2. If the selected discount type is 'AB', 'BA', or 'BE' and the agent code is not '3000000001' or '3000000007', and the in-house flag is 'N', and the web aggregator flag is 'N', then the system should reset the discount type selection and display an error message: "Employee Discount benefit is applicable through Direct Agents only".  
3. If the selected discount type is 'SE' and the agent code does not start with '51', then the system should reset the discount type selection and display an error message: "Stanchart Employee Discount benefit is applicable through Stanchart Bank Agents Only".  
4. The system should call a function to check UL allocations after validating the discount type selection.  
  
Definition of Done:  
- The discount type selection is validated based on the specified business rules.  
- Appropriate warning or error messages are displayed when the discount type selection does not meet the criteria.  
- The discount type selection is reset when the criteria are not met.  
- The function to check UL allocations is called after validation.  
- The functionality is tested and verified to ensure it works as expected.

# Delete Fund with Conditions

Type: ALLOCATION

Title: Delete Fund with Conditions  
  
Acceptance Criteria:  
1. The system should check if the product associated with the contract allows unlimited top-ups.  
2. If the product allows unlimited top-ups, the system should verify if there are any service requests for the contract that are not in 'R' (Rejected) or 'PA' (Pending Approval) status.  
3. If all service requests are in 'PU' (Pending Update) status, the system should display a warning message indicating that the fund cannot be deleted because the request is already approved.  
4. If there are no service requests in 'PU' status, the system should proceed to delete the fund.  
5. The system should navigate to the 'Unit\_linked' block and delete the record from the 'WIP\_AZBJ\_POLICY\_FUNDS\_REP' table if the 'CN\_WIP\_CONTINUE' control is set to 'T'.  
6. The system should delete the current record from the 'Unit\_linked' block.  
  
Definition of Done:  
- The user can attempt to delete a fund.  
- The system performs the necessary checks and either displays a warning message or proceeds with the deletion.  
- The fund is deleted from the 'WIP\_AZBJ\_POLICY\_FUNDS\_REP' table if applicable.  
- The current record is deleted from the 'Unit\_linked' block.  
- All conditions and checks are met as per the acceptance criteria.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- SELECT STATUS FROM AZBJ\_TPS\_SERVICE\_REQ WHERE CONTRACT\_ID = :CONTROL.CN\_CONTRACT\_ID AND STATUS NOT IN ('R','PA')  
- DELETE FROM WIP\_AZBJ\_POLICY\_FUNDS\_REP WHERE FUND\_ID = :unit\_linked.fund\_id

# Delete Fund from Auto Transfer Portfolio

Type: ALLOCATION

Title: Delete Fund from Auto Transfer Portfolio  
  
Acceptance Criteria:  
1. When the "Delete Fund" button is pressed, the system should navigate to the "UL\_AT\_PORTFOL" block.  
2. The system should check the total number of records in the "UL\_AT\_PORTFOL" block.  
3. If there is only one fund in the portfolio, the system should display an error message: "Minimum one fund should be required under Auto Transfer!" and exit the process.  
4. If there are multiple funds, the system should loop through the records.  
5. For each record where the "AT\_PORT\_DEL" field is marked as 'Y', the system should delete the record, provided there is more than one record remaining.  
6. The process should continue until all records are checked.  
7. The system should handle any exceptions gracefully without crashing.  
  
Definition of Done:  
- The "Delete Fund" button functionality is implemented as described.  
- The system correctly identifies and deletes the appropriate records.  
- An error message is displayed when attempting to delete the last remaining fund.  
- The process completes without unhandled exceptions.  
- 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 is specific to Oracle Forms and cannot be executed directly in the database without modification.

# Delete Fund Functionality

Type: ALLOCATION

Title: Delete Fund Functionality  
  
Acceptance Criteria:  
1. The system should check if the product associated with the fund allows unlimited top-ups.  
2. If the product allows unlimited top-ups, the system should verify if there are any service requests related to the contract that are not in 'R' (Rejected) or 'PA' (Pending Approval) status.  
3. If all service requests are either 'PU' (Pending Update) or approved, the system should display a warning message indicating that the fund cannot be deleted.  
4. If there are no pending or approved service requests, the system should proceed to delete the fund.  
5. The system should navigate to the 'Unit\_linked' block and delete the record from the 'WIP\_AZBJ\_POLICY\_FUNDS\_REP' table if the continuation status is 'T' (True).  
6. The system should delete the current record from the 'Unit\_linked' block.  
  
Definition of Done:  
- The user can successfully delete a fund only if the conditions specified in the acceptance criteria are met.  
- Appropriate warning messages are displayed when the fund cannot be deleted.  
- The fund is removed from the 'WIP\_AZBJ\_POLICY\_FUNDS\_REP' table and the 'Unit\_linked' block as per the conditions.  
- The functionality is tested and verified to ensure it works as expected.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- SELECT STATUS FROM AZBJ\_TPS\_SERVICE\_REQ WHERE CONTRACT\_ID = :CONTROL.CN\_CONTRACT\_ID AND STATUS NOT IN ('R','PA');  
- DELETE FROM WIP\_AZBJ\_POLICY\_FUNDS\_REP WHERE FUND\_ID = :unit\_linked.fund\_id;

# Manage Fund-Related Fields Based on Portfolio Strategy and Product Definitions

Type: ALLOCATION

Detailed description: As a user, I want the system to manage the enabling and disabling of fund-related fields based on the selected portfolio strategy and product definitions, so that the appropriate fields are editable or read-only depending on the business rules.  
  
Acceptance criteria:  
1. When the portfolio strategy is 'SO' and the product ID is one of the specified products ('NEW\_UG\_PLUS\_SP', 'WEALTH\_GAIN', 'WEALTH\_GAIN\_II', 'FLEXI\_ADVANTAGE', 'GUARANTEED\_MATURITY\_INSURANCE', 'FORTUNE\_GAIN', 'GOAL\_BASED\_SAVINGS', 'SMART\_WEALTH\_GOAL', 'SMART\_CHILD\_WEALTH\_GOAL', 'JOINT\_LIFE\_WEALTH'), the following fields should be enabled:  
 - Fund List  
 - Fund Allocation Percentage  
 - Portfolio Deletion Fund  
2. When the portfolio strategy is not 'SO' or the product ID is not one of the specified products, the above fields should be disabled.  
3. If the product ID is 'MONEY\_SECURE\_PLAN', the portfolio strategy should automatically be set to 'IS'.  
4. If the product ID is one of (307, 316, 331) and the portfolio strategy is one of ('TB', 'AT', 'CP'), the system should auto-populate funds.  
5. If the product ID is one of (307, 316, 331) and the portfolio strategy is one of ('TB', 'WL', 'CP'), the Portfolio Deletion Fund field should be disabled; otherwise, it should be enabled.  
6. If the product ID is 311 and the portfolio strategy is 'SO' and the package code does not contain 'SINGLE', the portfolio strategy should be set to null and an error message should be displayed indicating that the switching option is allowed only for single premium.  
  
Definition of Done:  
- The system correctly enables or disables the fund-related fields based on the portfolio strategy and product definitions.  
- The portfolio strategy is automatically set to 'IS' for the 'MONEY\_SECURE\_PLAN' product.  
- The system auto-populates funds for specific product IDs and portfolio strategies.  
- Appropriate error messages are displayed when business rules are violated.  
- All changes are tested and verified to ensure they meet the acceptance criteria.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- The procedure `AZBJ\_WOL\_FUND\_POPULATE` is used to populate fund details based on the portfolio strategy and product definitions. It retrieves data from the `azbj\_wheel\_of\_life` and `azbj\_tfv\_fund\_definition` tables and updates the `unit\_linked` block with the retrieved fund details.

# Delete Fund from Auto Transfer Portfolio

Type: ALLOCATION

Title: Delete Fund from Auto Transfer Portfolio  
  
Acceptance Criteria:  
1. When the "Delete Fund" button is pressed, the system should navigate to the "UL\_AT\_PORTFOL\_2" block.  
2. The system should check the total number of records in the "UL\_AT\_PORTFOL\_2" block.  
3. If there is only one fund in the portfolio, the system should display an error message: "Minimum one fund should be required under Auto Transfer!" and exit the process.  
4. If there are multiple funds, the system should loop through the records.  
5. For each record, if the "AT\_PORT\_DEL\_UL" field is marked as 'Y', the system should delete the record.  
6. The loop should continue until all records are processed or the last record is reached.  
7. The system should handle any exceptions gracefully without crashing.  
  
Definition of Done:  
- The "Delete Fund" button functionality is implemented as described.  
- The system correctly identifies and handles the minimum fund requirement.  
- The system successfully deletes the marked records.  
- Appropriate error messages are displayed when necessary.  
- The process completes without unhandled exceptions.  
  
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.

# Delete Fund with Validation

Type: ALLOCATION

Title: Delete Fund with Validation  
  
Acceptance Criteria:  
1. The system should check if the product associated with the contract allows unlimited top-ups.  
2. If the product allows unlimited top-ups, the system should verify if there are any service requests for the contract that are not in 'R' (Rejected) or 'PA' (Pending Approval) status.  
3. If all service requests are in 'PU' (Pending Update) status, the system should display a warning message indicating that the fund cannot be deleted because the request is already approved.  
4. If there are no service requests in 'PU' status, the system should proceed to delete the fund.  
5. The system should navigate to the 'Unit\_linked' section and delete the record from the 'WIP\_AZBJ\_POLICY\_FUNDS\_REP' table if the 'CN\_WIP\_CONTINUE' flag is set to 'T'.  
6. The system should delete the record from the current block.  
  
Definition of Done:  
- The user can attempt to delete a fund.  
- The system performs the necessary checks and validations.  
- Appropriate messages are displayed based on the conditions.  
- The fund is deleted if all conditions are met.  
- The record is removed from the 'WIP\_AZBJ\_POLICY\_FUNDS\_REP' table if applicable.  
- The record is deleted from the current block.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- SELECT STATUS FROM AZBJ\_TPS\_SERVICE\_REQ WHERE CONTRACT\_ID = :CONTROL.CN\_CONTRACT\_ID AND STATUS NOT IN ('R','PA')  
- DELETE FROM WIP\_AZBJ\_POLICY\_FUNDS\_REP WHERE FUND\_ID = :unit\_linked.fund\_id

# Dynamic Field Enable/Disable Based on Portfolio Strategy and Product Definition

Type: ALLOCATION

Title: Dynamic Field Enable/Disable Based on Portfolio Strategy and Product Definition  
  
Acceptance Criteria:  
1. When the portfolio strategy is 'SO' and the product definition is one of the specified products ('NEW\_UG\_PLUS\_SP', 'WEALTH\_GAIN', 'WEALTH\_GAIN\_II', 'FLEXI\_ADVANTAGE', 'GUARANTEED\_MATURITY\_INSURANCE', 'FORTUNE\_GAIN', 'GOAL\_BASED\_SAVINGS', 'SMART\_WEALTH\_GOAL', 'SMART\_CHILD\_WEALTH\_GOAL', 'JOINT\_LIFE\_WEALTH'), the fields 'FUNDLIST', 'FUNDAPOR', and 'PD\_DEL\_FUND' should be enabled.  
2. When the portfolio strategy is not 'SO' or the product definition is not one of the specified products, the fields 'FUNDLIST', 'FUNDAPOR', and 'PD\_DEL\_FUND' should be disabled.  
3. If the product definition is 'MONEY\_SECURE\_PLAN', the portfolio strategy should be automatically set to 'IS'.  
4. If the product ID is 307, 316, or 331 and the portfolio strategy is 'TB', 'AT', or 'CP', the system should call a procedure to auto-populate funds.  
5. If the product ID is 307, 316, or 331 and the portfolio strategy is 'TB', 'WL', or 'CP', the 'PB\_DEL\_FUND' field should be disabled; otherwise, it should be enabled.  
6. If the product ID is 311, the portfolio strategy is 'SO', and the package code does not contain 'SINGLE', the portfolio strategy should be set to null and an error message should be displayed indicating that the switching option is allowed only for single premium.  
  
Definition of Done:  
- The system dynamically enables or disables fields based on the selected portfolio strategy and product definition.  
- The portfolio strategy is automatically set to 'IS' for the 'MONEY\_SECURE\_PLAN' product.  
- The system calls the appropriate procedure to auto-populate funds when required.  
- The 'PB\_DEL\_FUND' field is enabled or disabled based on the product ID and portfolio strategy.  
- An error message is displayed when the switching option is not allowed for the selected package code.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- The procedure `AZBJ\_WOL\_FUND\_POPULATE` is used to populate fund details based on the product ID, benefit term, and cover code. It retrieves data from the `azbj\_wheel\_of\_life` and `azbj\_tfv\_fund\_definition` tables and populates the `unit\_linked` block with the retrieved fund details.

# View and Verify Address Details

Type: VERIFIED\_ADDRESS

Title: View and Verify Address Details  
  
Acceptance Criteria:  
1. The user should be able to see the following address fields:  
 - Door No  
 - Building Name  
 - Plot No/Street  
 - State  
 - Area  
 - Fax  
 - Country  
 - Pincode  
 - District  
 - Place  
 - Telephone  
 - Email Id  
2. The "Verified Address" button should be available but not visible to the user.  
3. The "Exit" button should be available to close the address verification section.  
  
Definition of Done:  
1. The address fields are displayed in a structured format with appropriate labels.  
2. The "Verified Address" button is present but hidden from the user interface.  
3. The "Exit" button is functional and allows the user to close the address verification section.  
4. The user interface is consistent with the design specifications and is user-friendly.  
5. All fields are non-editable and display only the address information.  
6. The layout and alignment of the fields are as per the design specifications.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- No specific DB queries provided in the XML content.

# Exit Button Functionality in Verified Address Section

Type: VERIFIED\_ADDRESS

Title: Exit Button Functionality in Verified Address Section  
  
Acceptance Criteria:  
1. When the "Exit" button is pressed, the system should navigate to the "Insured Person" section, specifically to the "Contact Email" field.  
2. The "Verified Address" section should be hidden from view.  
3. The window associated with the "Verified Address" section should be closed.  
  
Definition of Done:  
- The "Exit" button successfully navigates to the "Insured Person" section.  
- The "Verified Address" section is hidden.  
- The window for the "Verified Address" section is closed.  
- All changes are tested and verified to ensure they work as expected.

# Verify and Store Address Details

Type: VERIFIED\_ADDRESS

Detailed description: As a user, I want to verify the address of an insured person and store the verification details in the system, so that the address information is accurate and up-to-date.  
  
Acceptance criteria:  
1. When the "Verified Address" button is pressed, the system should:  
 - Navigate to the "VERIFIED\_ADDRESS.EXIT" item.  
 - Set the `br\_add\_ver` control to 'Y'.  
 - Check if an entry for the address verification already exists in the `azbj\_sign\_address\_verification` table for the given contract ID and column name 'ADDRESS'.  
 - If no entry exists, insert a new record into the `azbj\_sign\_address\_verification` table with the contract ID, column name 'ADDRESS', verification flag, user, current date, and a null column description.  
 - Commit the transaction.  
 - Retrieve and display the verified address details using the `azbj\_get\_verified\_address\_dtls` procedure, which includes fields such as door number, building name, plot number/street, area, state, place, district, pincode, country, fax, telephone, and email.  
 - Show the "WIN\_VER\_ADD" window and the "CAN\_VERIFIED\_ADDRESS" view.  
 - Handle any exceptions by displaying an error message with the SQL error message.  
  
Definition of Done:  
- The "Verified Address" button functionality is implemented and tested.  
- The system correctly navigates, sets controls, checks for existing entries, inserts new records, commits transactions, retrieves and displays address details, and handles exceptions as specified.  
- The "WIN\_VER\_ADD" window and "CAN\_VERIFIED\_ADDRESS" view are displayed correctly.  
- All acceptance criteria are met, and the feature is verified by QA.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Check for existing entry:  
 ```sql  
 SELECT COUNT(ROWNUM) INTO v\_cnt FROM azbj\_sign\_address\_verification  
 WHERE contract\_id = :control.cn\_contract\_id  
 AND column\_name = 'ADDRESS';  
 ```  
  
- Insert new record:  
 ```sql  
 INSERT INTO azbj\_sign\_address\_verification(contract\_id, column\_name, verified\_flag, verified\_user, verified\_date, column\_desc)  
 VALUES (:control.cn\_contract\_id, 'ADDRESS', :control.br\_add\_ver, USER, SYSDATE, NULL);  
 ```  
  
- Commit transaction:  
 ```sql  
 FORMS\_DDL('COMMIT');  
 ```  
  
- Retrieve verified address details:  
 ```sql  
 azbj\_get\_verified\_address\_dtls(  
 TO\_CHAR(NVL(:insured\_person.ip\_sign\_card\_no, :insured\_person.ip\_verf\_no)),  
 :VERIFIED\_ADDRESS.DOOR\_NO,  
 :VERIFIED\_ADDRESS.BUILDING\_NAME,  
 :VERIFIED\_ADDRESS.PLOT\_NO\_STREET,  
 :VERIFIED\_ADDRESS.AREA,  
 :VERIFIED\_ADDRESS.STATE,  
 :VERIFIED\_ADDRESS.PLACE,  
 :VERIFIED\_ADDRESS.DISTRICT,  
 :VERIFIED\_ADDRESS.PINCODE,  
 :VERIFIED\_ADDRESS.COUNTRY,  
 :VERIFIED\_ADDRESS.FAX,  
 :VERIFIED\_ADDRESS.TELEPHONE,  
 :VERIFIED\_ADDRESS.EMAIL  
 );  
 ```

# Manage Unit-Linked Funds

Type: UNIT\_LINKED

Title: Manage Unit-Linked Funds  
  
Acceptance Criteria:  
1. The system should display a list of unit-linked funds with the following attributes:  
 - Row Number  
 - Fund Number  
 - Fund Name  
 - Percentage Allocation  
 - Fund ID (read-only)  
 - Total Fund Value (calculated as the sum of percentage allocations)  
2. The system should allow the user to select a fund from a predefined list.  
3. The system should ensure that the total number of selected funds does not exceed four.  
4. The system should calculate and display the total fund value dynamically based on the percentage allocations of the selected funds.  
5. The system should fetch fund details based on specific criteria, including product ID, date range, and portfolio strategy.  
  
Definition of Done:  
- The user can view and manage unit-linked fund details.  
- The system restricts the selection to a maximum of four funds.  
- The total fund value is calculated and displayed correctly.  
- The fund list is fetched based on the defined criteria and displayed to the user.  
  
DB queries for Table reference CRUD operations only(With Usage):  
```sql  
SELECT fund\_short\_name, fund\_full\_name  
FROM AZBJ\_TFV\_FUND\_DEFINITION a, azbj\_cover\_funds b  
WHERE a.fund\_short\_name = b.fund\_name  
 AND b.product\_id = :control.cn\_product\_id  
 AND :susac.sa\_daterecd >= NVL(start\_date, '01-apr-2005')  
 AND :susac.sa\_daterecd <= NVL(end\_date, '01-jan-3000')  
 AND cover\_code = :covers.cv\_cover\_code  
 AND 1 = (CASE  
 WHEN :ALLOCATION.PORTFOLIO\_STRATEGY = 'SO'  
 AND a.FUND\_ID <> 'FVFD000025'  
 THEN 0  
 WHEN :ALLOCATION.PORTFOLIO\_STRATEGY <> 'SO'  
 THEN 1  
 ELSE 1  
 END);  
```

# Validate Apportionment Percentage for Cash Plus Pension Fund

Type: UNIT\_LINKED

Detailed description: As a user, I want to ensure that the apportionment percentage for the Cash Plus Pension Fund does not exceed 20% when certain product IDs are selected, so that the business rules are adhered to.  
  
Acceptance criteria:  
1. When the product ID is one of the following: 31, 32, 33, 34, 49, or 50, and the fund ID is 'NCPPF', the apportionment percentage should be validated.  
2. If the apportionment percentage exceeds 20%, an error message should be displayed: "Apportionment for Cash Plus Pension Fund cannot be more than 20".  
3. The form status should be updated to 'Y' after the validation check.  
  
Definition of Done:  
- The validation logic is implemented and tested.  
- The error message is displayed correctly when the conditions are met.  
- The form status is updated appropriately after the validation.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided logic is specific to form validation and does not involve direct CRUD operations on the database.

# Fund Selection on Double-Click

Type: UNIT\_LINKED

Title: Fund Selection on Double-Click  
  
Acceptance Criteria:  
1. When the user double-clicks on the "Fund Name" field, the system should check if the global loading status is 'F'.  
2. If the global loading status is 'F', the system should set the form status to 'Y'.  
3. The system should then validate the "Fund Name" field from the list.  
4. The system should navigate to the 'covers' block and then return to the 'UNIT\_LINKED' block.  
5. If the portfolio strategy is 'SO', the system should execute a query to fetch the fund details where the fund ID is 'FVFD000025'.  
6. If the portfolio strategy is not 'SO', the system should execute a query to fetch the fund details based on the product ID, date range, and cover code.  
7. The system should populate the fund list with the results of the query.  
8. The system should display the list of funds to the user.  
9. The user should be able to select a fund from the list, and the selected fund should be displayed in the "Fund Name" field.  
  
Definition of Done:  
- The user can double-click on the "Fund Name" field to open a list of available funds.  
- The system correctly validates and fetches the fund details based on the portfolio strategy and other criteria.  
- The list of funds is displayed to the user, and the user can select a fund from the list.  
- The selected fund is displayed in the "Fund Name" field.  
  
DB queries for Table reference CRUD operations only (With Usage):  
```sql  
SELECT fund\_short\_name, fund\_full\_name  
FROM AZBJ\_TFV\_FUND\_DEFINITION a, azbj\_cover\_funds b  
WHERE a.fund\_short\_name = b.fund\_name  
 AND b.product\_id = :control.cn\_product\_id  
 AND :susac.sa\_daterecd >= NVL(start\_date, '01-apr-2005')  
 AND :susac.sa\_daterecd <= NVL(end\_date, '01-jan-3000')  
 AND cover\_code = :covers.cv\_cover\_code  
 AND 1 = (CASE  
 WHEN :ALLOCATION.PORTFOLIO\_STRATEGY = 'SO' AND a.FUND\_ID <> 'FVFD000025' THEN 0  
 WHEN :ALLOCATION.PORTFOLIO\_STRATEGY <> 'SO' THEN 1  
 ELSE 1  
 END);  
```

# Manage Spouse Details

Type: SPOUSE\_DETAILS

Title: Manage Spouse Details  
  
Acceptance Criteria:  
1. The system should allow the user to input the spouse's name, which should be stored in uppercase and have a maximum length of 80 characters.  
2. The system should allow the user to input the spouse's surname, which should be stored in uppercase and have a maximum length of 40 characters.  
3. The system should allow the user to input the spouse's date of birth in the format "dd/mm/yyyy".  
4. The input fields for spouse details should be displayed on the "POLICY\_HOLDER" tab page.  
  
Definition of Done:  
1. The user can successfully input and save the spouse's name, surname, and date of birth.  
2. The input fields for spouse details are displayed correctly on the "POLICY\_HOLDER" tab page.  
3. The data is stored in the correct format and adheres to the specified length restrictions.  
4. The user interface is intuitive and user-friendly, with clear prompts for each input field.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include specific database queries or table references.

# Validate Spouse Surname Input

Type: SPOUSE\_DETAILS

User Story: Validate Spouse Surname Input  
  
Detailed Description:  
As a user, I want the system to validate the input for the spouse's surname field to ensure that certain conditions are met before allowing further actions. Specifically, if a certain premium frequency field is not null, the system should disable certain toolbar buttons to prevent form submission or exit until the validation is complete.  
  
Acceptance Criteria:  
1. When the user inputs a value in the spouse's surname field:  
 - If the premium frequency field (`CV\_FREQ\_PREM`) is not null:  
 - The "Commit Form" button should be disabled.  
 - The "Exit Form" button should be disabled.  
 - If the premium frequency field is null, no changes should be made to the toolbar buttons.  
  
Definition of Done:  
- The system correctly identifies when the premium frequency field is not null.  
- The "Commit Form" and "Exit Form" buttons are disabled when the premium frequency field is not null.  
- The system allows normal operation when the premium frequency field is null.  
- All changes are tested and verified to ensure they meet the acceptance criteria.  
  
DB Queries for Table Reference CRUD Operations:  
- Not applicable as the provided logic does not involve direct database CRUD operations.

# Validate Spouse Name Input

Type: SPOUSE\_DETAILS

User Story: Validate Spouse Name Input  
  
Detailed Description:  
As a user, I want to ensure that the "Spouse Name" field is validated correctly when inputting data, so that the system can handle specific conditions and update properties accordingly.  
  
Acceptance Criteria:  
1. When the "Spouse Name" field is validated:  
 - The system should set a specific value (e.g., 100) to a related field.  
 - If a certain condition is met (e.g., another field is not null), the system should disable specific form actions (e.g., commit and exit actions).  
  
Definition of Done:  
- The "Spouse Name" field validation logic is implemented.  
- The system correctly sets the related field value.  
- The system disables specific form actions based on the condition.  
- The functionality is tested and verified to work as expected.  
  
Explanation of Oracle Form Logic:  
- The `WHEN-VALIDATE-ITEM` trigger is used to execute logic when the "Spouse Name" field is validated.  
- The trigger sets a value to a related field and checks if another field is not null.  
- If the condition is met, it disables specific form actions to prevent further actions until the condition is resolved.  
  
DB Queries for Table Reference CRUD Operations:  
- Not applicable as the provided XML content does not include any direct database queries for CRUD operations.

# Validate Spouse Date of Birth

Type: SPOUSE\_DETAILS

User Story: Validate Spouse Date of Birth  
  
Detailed Description:  
As a user, I want to ensure that the date of birth entered for the spouse is valid and meets specific business rules, so that the data integrity and business requirements are maintained.  
  
Acceptance Criteria:  
1. The system should validate that the spouse's date of birth is not greater than the current date.  
2. The system should calculate the entry age based on the spouse's date of birth and the policy's inception date or effective date.  
3. If the calculated entry age is 18 or older and the appointee's name is not null, the system should clear the appointee's name, relation, and date of birth fields and display a warning message indicating that an appointee cannot be entered for a major.  
4. If the premium frequency is not null, the system should disable the form's commit and exit buttons to prevent further changes.  
  
Definition of Done:  
- The system correctly validates the spouse's date of birth against the current date.  
- The system accurately calculates the entry age and applies the necessary business rules.  
- The system displays appropriate warning messages and clears fields as required.  
- The form's commit and exit buttons are disabled when the premium frequency is not null.  
  
DB Queries for Table Reference CRUD Operations:  
- Not applicable as the provided logic is specific to Oracle Forms and cannot be executed directly in the database without modification.  
  
Explanation of Oracle Form Logic:  
- The `WHEN-VALIDATE-ITEM` trigger is used to validate the spouse's date of birth.  
- The trigger checks if the date of birth is greater than the current date and displays an error message if it is.  
- It calculates the entry age using a custom function and checks if the age is 18 or older.  
- If the age is 18 or older and the appointee's name is not null, it clears the appointee's details and displays a warning message.  
- It also checks if the premium frequency is not null and disables the form's commit and exit buttons to prevent further changes.

# Automatic Referral ID Validation and Population

Type: AGENTS

Detailed description: As a user, I want the system to automatically validate and populate the referral ID based on the agent's code and the bank's association, so that I can ensure the correct referral information is captured without manual intervention.  
  
Acceptance criteria:  
1. When a new instance of the AGENTS block is created, the system should check if the referral ID is null.  
2. If the referral ID is null and the bank association is 'SY', the system should prompt the user to select a referral ID from a predefined list.  
3. If the bank association contains 'Y', the system should populate a record group with a query and set the LOV (List of Values) for the referral ID based on the populated data.  
4. The system should enable or disable the validation from the list property of the referral ID based on the referral list flag.  
  
Definition of Done:  
- The system correctly prompts the user to select a referral ID when necessary.  
- The referral ID LOV is populated based on the bank association and agent's code.  
- The validation from the list property of the referral ID is set correctly based on the referral list flag.  
- The functionality is tested and verified to work as expected without any errors.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- The following query is used to populate the referral ID LOV when the bank association contains 'Y':  
 ```sql  
 Select branch\_code, branch\_name   
 from azbj\_syn\_bank\_branches   
 where br\_type like '3A6%'  
 ```

# Notification and Reset for Corporate Agents Distribution Channel

Type: AGENTS

Title: Notification and Reset for Corporate Agents Distribution Channel  
  
Acceptance Criteria:  
- When the user selects the 'Corporate Agents Distribution Channel' option, a message should be displayed indicating that this option is not available.  
- After displaying the message, the selection should automatically reset to the default value.  
  
Definition of Done:  
- The user receives a notification when selecting the 'Corporate Agents Distribution Channel' option.  
- The selection is reset to the default value after the notification is displayed.  
- 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 XML content does not include any direct database CRUD operations.

# Enable/Disable SAVE\_EXIT Action Based on Referred to FPU Checkbox

Type: AGENTS

Title: Enable/Disable SAVE\_EXIT Action Based on Referred to FPU Checkbox  
  
Acceptance Criteria:  
1. When the "Referred to FPU" checkbox is checked:  
 - The "SAVE\_EXIT" action should be enabled.  
2. When the "Referred to FPU" checkbox is unchecked:  
 - The "SAVE\_EXIT" action should be disabled.  
  
Definition of Done:  
- The "SAVE\_EXIT" action is correctly enabled or disabled based on the status of the "Referred to FPU" checkbox.  
- The functionality is tested and verified to ensure it works as expected.  
- The implementation is independent of any specific technology or Oracle Forms terminology.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include any direct database queries.

# Manage WSM Allowed Status for Agents

Type: AGENTS

Detailed description: As a user, I want to manage the status of an agent's WSM (Work Status Management) allowance through a checkbox, so that I can easily toggle between allowed and not allowed statuses based on specific conditions.  
  
Acceptance criteria:  
1. When the checkbox is unchecked (value 'N') and both the MU\_TESTNO and MU\_DESC fields in the MED\_UW section are not null, the system should:  
 - Automatically check the checkbox (set value to 'Y').  
 - Display a message asking if the user wants to convert to a non-medical case.  
 - Navigate to the MED\_UW section and clear its contents.  
 - Uncheck the checkbox (set value to 'N').  
 - Return focus to the checkbox.  
  
2. When the checkbox is checked (value 'Y'):  
 - If the MED\_UW.MU\_ADD\_TEST field is disabled, it should be enabled.  
  
3. When the checkbox is unchecked (value 'N'):  
 - If the MED\_UW.MU\_ADD\_TEST field is enabled, it should be disabled.  
  
Definition of Done:  
- The checkbox functionality should be implemented as described.  
- The system should correctly handle the conditions and actions specified in the acceptance criteria.  
- The user interface should reflect the changes in the checkbox and related fields without any errors.  
- The feature should be 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 direct database queries.

# Validate Agent Code Against Proposal Form

Type: AGENTS

Title: Validate Agent Code Against Proposal Form  
  
Description: As a user, I want to ensure that the agent code entered in the system is validated against the proposal form to prevent incorrect data entry.  
  
Acceptance Criteria:  
1. When the user enters the agent code, the system should validate it against the agent code from the proposal form.  
2. If the agent code does not match the one from the proposal form, the system should display an error message indicating that the agent code is incorrect and prompt the user to re-enter it.  
  
Definition of Done:  
1. The agent code validation logic is implemented and tested.  
2. The system displays an appropriate error message when the agent code does not match.  
3. The user is prompted to re-enter the agent code if it is incorrect.  
4. The functionality is tested and verified to work as expected.  
  
Dependencies:  
- This user story is dependent on the Proposal Form block for retrieving the correct agent code.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Not applicable as the provided XML content does not include any specific database queries.

# Validate Agent's Sub-Name Field

Type: AGENTS

Detailed description: As a user, I want the system to validate the agent's sub-name field to ensure it meets specific criteria based on the agent's code, so that data integrity is maintained.  
  
Acceptance criteria:  
1. When the user double-clicks on the agent's sub-name field, the system should check the agent's code. If the code matches certain criteria (e.g., not starting with '59D' and containing specific characters), the system should display a list of values. If the code starts with '59D', a different list of values should be shown.  
2. When the user presses a key to list values in the agent's sub-name field, the same validation logic as the double-click should be applied.  
3. When the user enters a value in the agent's sub-name field, the system should validate that if the agent's code starts with '5X9' and the sub-name is not null, the sub-name must be a numeric value of exactly 9 digits. If the value is not numeric or not 9 digits, an error message should be displayed, and the field should be cleared.  
  
Definition of Done:  
- The system correctly displays the appropriate list of values based on the agent's code when the user double-clicks or presses a key in the agent's sub-name field.  
- The system validates the agent's sub-name field according to the specified criteria and displays relevant error messages when the criteria are not met.  
- The system logs any errors encountered during the validation process for debugging purposes.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- The system should use the following query to fetch branch names for the list of values:  
 ```sql  
 SELECT BRANCH\_NAME   
 FROM azbj\_syn\_bank\_branches  
 WHERE br\_type LIKE 'N59DL%'  
 ```  
- This query is used to populate the list of values when the agent's code starts with '59D'.

# Validate Agent's Referral Name and Associated Details

Type: AGENTS

Detailed description: As a user, I want to validate the agent's referral name and associated details when an agent code is provided, so that I can ensure the correct agent information is displayed and stored.  
  
Acceptance criteria:  
1. When an agent code is provided, the system should:  
 - Retrieve the unique code and internal ID from the `dmt\_agents` table based on the provided agent code.  
 - Fetch the partner details from the `CP\_PARTNERS` table using the retrieved unique code.  
 - Concatenate and display the partner's title, first name, middle name, and surname in the agent's name field.  
 - If the agent's name is still null, set it to the partner's name.  
 - Retrieve and display the agent's license number and branch code from the `azbj\_agents\_ext` table using the internal ID.  
  
2. If the agent code is null or empty, the system should prompt the user to enter an agent code.  
  
Definition of Done:  
- The system correctly retrieves and displays the agent's referral name and associated details when a valid agent code is provided.  
- The system prompts the user to enter an agent code if it is null or empty.  
- All data retrievals and updates are performed accurately as per the specified logic.  
- The functionality is tested and verified to ensure it meets the acceptance criteria.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Retrieve unique code and internal ID:  
 ```sql  
 SELECT CUST\_PART\_UNIQUE\_CODE, int\_id   
 INTO AZBJ\_MC\_PART\_ID, azbj\_mn\_int\_id  
 FROM dmt\_agents  
 WHERE reference\_code = :ag\_agent\_code;  
 ```  
  
- Fetch partner details:  
 ```sql  
 SELECT   
 INTO AZBJ\_MC\_PARTNER   
 FROM CP\_PARTNERS   
 WHERE CP\_PARTNERS.PARTNER\_REF = AZBJ\_MC\_PART\_ID;  
 ```  
  
- Retrieve agent's license number and branch code:  
 ```sql  
 SELECT licence\_no, branch\_code   
 INTO :ag\_agent\_lic\_no, :ag\_branch\_code  
 FROM azbj\_agents\_ext  
 WHERE int\_id = azbj\_mn\_int\_id;  
 ```

# Implement Declined Reason Selection for Agents

Type: AGENTS

Title: Implement Declined Reason Selection for Agents  
  
Description: As a user, I want to select a reason for declining an agent's request from a predefined list, so that I can ensure consistency and accuracy in the reasons provided.  
  
Acceptance Criteria:  
1. The "Declined Reason" field should display a list of predefined reasons when the user interacts with it.  
2. The list of reasons should be fetched from the database table `azbj\_system\_constants` where `sys\_type` is 'DECLN\_TYPE'.  
3. The list should display both the `SYS\_CODE` and `SYS\_DESC` columns from the `azbj\_system\_constants` table.  
4. The list should be automatically positioned and titled "DECLINE REASONS".  
5. The "Declined Reason" field should not be visible by default.  
6. The list should be triggered by user interactions such as double-clicking or other specified actions.  
  
Definition of Done:  
1. The "Declined Reason" field is implemented and integrated into the user interface.  
2. The field correctly displays a list of predefined reasons when interacted with.  
3. The list is populated with data from the `azbj\_system\_constants` table based on the specified query.  
4. The field and list meet all specified acceptance criteria.  
5. The functionality is tested and verified to ensure it works as expected.  
  
DB queries for Table reference CRUD operations only (With Usage):  
```sql  
SELECT SYS\_CODE, SYS\_DESC   
FROM azbj\_system\_constants   
WHERE sys\_type = 'DECLN\_TYPE';  
```  
  
Block Name: AGENTS

# Implement IP Type Dropdown for Agents

Type: AGENTS

Title: Implement IP Type Dropdown for Agents  
  
Acceptance Criteria:  
1. The IP type field should be a dropdown list with two options.  
2. The default value of the IP type should be set to "1".  
3. The field should be visible and enabled for user interaction.  
4. The field should allow both insertion and updates.  
5. The field should be navigable using the keyboard and mouse.  
  
Definition of Done:  
1. The IP type dropdown list is implemented and displays two options.  
2. The default value of "1" is set for the IP type field.  
3. The field is visible and enabled for user interaction.  
4. The field allows insertion and updates.  
5. The field is navigable using both keyboard and mouse.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the field is not a database item.

# Validate Referral ID Field in Agents Section

Type: AGENTS

Title: Validate Referral ID Field in Agents Section  
  
Acceptance Criteria:  
1. If the "Referral ID" field is empty and the agent's code matches certain patterns (e.g., 'SY', 'CB', '%Y%'), the system should prompt the user to select a value from a list of values (LOV).  
2. The LOV displayed should be based on the agent's code:  
 - For 'SY', the "Syndicate" LOV should be shown.  
 - For 'CB', the "AZBJ\_CB\_LOV" should be shown.  
 - For codes containing 'Y', the "AZBJ\_XY\_LOV" should be shown.  
3. The system should dynamically populate the LOV based on the agent's code and other conditions.  
4. If the "Referral ID" field is not empty, the system should validate the entered value against the list.  
  
Definition of Done:  
- The "Referral ID" field validation logic is implemented and tested.  
- The LOVs are correctly displayed based on the agent's code.  
- The system dynamically populates the LOVs based on the agent's code.  
- The entered "Referral ID" is validated against the list if not empty.  
- All acceptance criteria are met, and the feature is tested and verified.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Query for "Syndicate" LOV:  
 ```sql  
 Select branch\_code, branch\_name   
 from azbj\_syn\_bank\_branches   
 where br\_type = (case when :agents.ag\_agent\_code like '522%' then 'B'   
 else substr(:agents.AG\_AGENT\_CODE,1,3) end)  
 ```  
  
- Query for "AZBJ\_CB\_LOV":  
 ```sql  
 select reference\_code,  
 DECODE(CP.PARTNER\_TYPE,'P',CP.FIRST\_NAME || ' ' ||CP.MIDDLE\_NAME|| ' ' ||CP.SURNAME,CP.INSTITUTION\_NAME) FULL\_NAME  
 from dmt\_agents a, azbj\_agents\_ext b, cp\_partners cp  
 where a.int\_id = b.int\_id  
 and a.CUST\_PART\_UNIQUE\_CODE = cp.partner\_ref  
 and recruited\_by = :agents.ag\_agent\_code  
 and (reference\_code like '220%' or reference\_code like '221%')  
 ```  
  
- Query for "AZBJ\_XY\_LOV":  
 ```sql  
 Select branch\_code, branch\_name   
 from azbj\_syn\_bank\_branches   
 where br\_type like '3A6%'  
 ```  
  
- Query for "AZBJ\_XY1\_LOV":  
 ```sql  
 Select branch\_name   
 from azbj\_syn\_bank\_branches   
 where br\_type like 'N59DR%'  
 ```

# Validate STM Code and Navigate to Next Item

Type: AGENTS

Detailed description: As a user, I want to ensure that the STM Code field in the AGENTS section is validated correctly and navigates to the appropriate next item based on certain conditions.  
  
Acceptance criteria:  
1. When the STM Code field is validated, it should check if the agent code starts with '1'. If it does, the STM Code should match the Unit Manager code. If not, an error message should be displayed prompting the user to re-enter the STM Code from the Proposal form.  
2. If the agent code and STM Code match the Unit Manager code, the fields AG\_AGENT\_CODE, AG\_AGENT\_NAME, AG\_UNIT\_MGR, and AG\_UNIT\_MGR\_NAME should be made visible.  
3. When navigating to the next item, if the INSURED\_PERSON.IP\_CP\_SEARCH field is enabled, the focus should move to it. If not, and if the INSURED\_PERSON.IP\_HEIGHT field is enabled, the focus should move to it instead.  
  
Definition of Done:  
- The STM Code field validation logic is implemented and tested.  
- The navigation logic to the next item is implemented and tested.  
- Error messages and visibility changes are correctly displayed and handled.  
- The functionality is tested and verified to work as expected without any Oracle Forms-specific terminology or dependencies.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- No specific database queries are mentioned in the provided XML content.

# Streamline Policy Issuance Based on FMR/MER Cleanliness

Type: AGENTS

Title: Streamline Policy Issuance Based on FMR/MER Cleanliness  
  
Acceptance Criteria:  
1. When the checkbox for "Clean FMR/MER" is checked, the system should navigate to the 'med\_uw' section and count the number of records.  
2. If there is only one record and the 'mu\_testno' field is null, the record count should be set to zero.  
3. If the record count is not equal to one, the checkbox should be automatically unchecked.  
4. If there is exactly one record and the 'mu\_testno' field is either '1a', 'FMR', or 'MER', and the 'mu\_res\_recd' field is 'Y', and the 'clean\_fmr' checkbox is checked, and the branch code does not start with 'P00', then a global flag 'clean\_fmr\_flag' should be set to 'Y'.  
5. If any of the above conditions are not met, the global flag 'clean\_fmr\_flag' should be set to 'N' and the 'clean\_fmr' checkbox should be unchecked.  
  
Definition of Done:  
- The policy issuance process should be validated based on the cleanliness of the FMR/MER records.  
- The system should automatically handle the checkbox state and global flag based on the specified conditions.  
- The functionality should be tested and verified to ensure it works as expected without any Oracle Forms-specific dependencies.

# Employee ID Validation

Type: AGENTS

Title: Employee ID Validation  
  
Acceptance Criteria:  
1. The Employee ID should not contain any special characters (e.g., !, @, , etc.).  
2. The Employee ID should not contain any alphabetic characters.  
3. The length of the Employee ID should be exactly 6 digits if the agent code starts with '59C', '59S', '5X9', or '59L'.  
4. The length of the Employee ID should be exactly 4 digits if the agent code starts with '59C', '59D', '59S', '5X9', '52S', or '59L'.  
5. If the Employee ID does not meet the above criteria, appropriate error messages should be displayed to the user.  
  
Definition of Done:  
- The validation rules for the Employee ID are implemented and tested.  
- Error messages are displayed correctly when the validation rules are not met.  
- The system allows the user to proceed only when the Employee ID meets all the validation criteria.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- The system should fetch the branch code and branch name from the `azbj\_syn\_bank\_branches` table based on the agent code.  
- The query to fetch branch details should be:  
 ```sql  
 SELECT branch\_code, branch\_name   
 FROM azbj\_syn\_bank\_branches   
 WHERE br\_type = (CASE   
 WHEN :agents.ag\_agent\_code LIKE '522%' THEN 'B'   
 ELSE SUBSTR(:agents.AG\_AGENT\_CODE, 1, 3)   
 END);  
 ```

# Proceed Button Functionality on Agent Grid

Type: AGENTS

Detailed description: As a user, I want to be able to proceed to the next section of the application by pressing a button labeled "Proceed" on the Agent Grid screen, so that I can continue my workflow efficiently.  
  
Acceptance criteria:  
1. When the "Proceed" button is pressed, the application should navigate to the "MED\_UW" section.  
2. The system should set a control flag named `CN\_CHECKED\_AGENT\_GRID` to 'Y' to indicate that the Agent Grid has been checked.  
  
Definition of Done:  
1. The "Proceed" button is visible and functional on the Agent Grid screen.  
2. Pressing the "Proceed" button successfully navigates the user to the "MED\_UW" section.  
3. The control flag `CN\_CHECKED\_AGENT\_GRID` is set to 'Y' upon pressing the button.  
4. 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 XML content does not include any specific database CRUD operations.

# Validate Agent Code and Retrieve Agent Details

Type: AGENTS

Title: Validate Agent Code and Retrieve Agent Details  
  
Acceptance Criteria:  
1. When an Agent Code is entered and is not null, the system should:  
 - Retrieve the unique code and internal ID from the `dmt\_agents` table based on the entered Agent Code.  
 - Fetch the partner details from the `CP\_PARTNERS` table using the retrieved unique code.  
 - Populate the agent's name by concatenating the partner's title, first name, middle name, and surname.  
 - If the agent's name is still null, use the partner's name.  
 - Retrieve the agent's license number and branch code from the `azbj\_agents\_ext` table using the internal ID.  
2. If the Agent Code is null or empty, the system should display an error message prompting the user to enter an Agent Code.  
3. When the Agent Code starts with '5X9%', the system should:  
 - Enable and make the sub-name field navigable.  
 - Display a list of values for the user to select a lead.  
4. If the Agent Code does not start with '5X9%', the system should:  
 - Display a list of values for the user to select a lead if certain conditions are met.  
  
Definition of Done:  
- The system correctly validates the Agent Code and retrieves the necessary details.  
- Error messages are displayed appropriately when the Agent Code is null or empty.  
- The sub-name field is enabled and navigable when the Agent Code starts with '5X9%'.  
- The list of values for selecting a lead is displayed based on the conditions specified.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Retrieve unique code and internal ID:  
 ```sql  
 SELECT CUST\_PART\_UNIQUE\_CODE, int\_id   
 INTO AZBJ\_MC\_PART\_ID, azbj\_mn\_int\_id  
 FROM dmt\_agents  
 WHERE reference\_code = :ag\_agent\_code;  
 ```  
  
- Fetch partner details:  
 ```sql  
 SELECT   
 INTO AZBJ\_MC\_PARTNER   
 FROM CP\_PARTNERS   
 WHERE CP\_PARTNERS.PARTNER\_REF = AZBJ\_MC\_PART\_ID;  
 ```  
  
- Retrieve agent's license number and branch code:  
 ```sql  
 SELECT licence\_no, branch\_code   
 INTO :ag\_agent\_lic\_no, :ag\_branch\_code  
 FROM azbj\_agents\_ext  
 WHERE int\_id = azbj\_mn\_int\_id;  
 ```  
  
- List of values for selecting a lead:  
 ```sql  
 SELECT branch\_code || '-' || BRANCH\_NAME branch\_name, branch\_code  
 FROM azbj\_syn\_bank\_branches  
 WHERE br\_type LIKE '5X9L%'  
 UNION ALL  
 SELECT 'NOSUBCODE', 'NOSUBCODE' FROM DUAL;  
 ```

# Navigate to Data Block and Display First Record

Type: AGENTS

User Story: Navigate to Data Block and Display First Record  
  
Title: As a user, I want to navigate to the data block and view the first record when I press the "Agents" button, so that I can quickly access the initial data entry in the data block.  
  
Acceptance Criteria:  
1. When the "Agents" button is pressed, the system should navigate to the data block.  
2. The system should automatically display the first record in the data block.  
  
Definition of Done:  
1. The "Agents" button is functional and correctly navigates to the data block.  
2. The first record in the data block is displayed upon pressing the "Agents" button.  
3. The feature is tested and verified to ensure it works as expected.  
  
SQL Query for Reference:  
- Not applicable as the provided XML content does not include any specific database queries.

# View Underwriting Details

Type: AGENTS

Title: View Underwriting Details  
  
Acceptance Criteria:  
- When the user presses the "Show U/W Details <Alt+1>" button, the system should execute the `azbj\_clean\_mer` procedure to clean up or prepare data for display.  
- The button should be accessible via the Alt+1 keyboard shortcut.  
  
Definition of Done:  
- The button labeled "Show U/W Details <Alt+1>" is present on the interface.  
- Pressing the button triggers the `azbj\_clean\_mer` procedure.  
- The Alt+1 keyboard shortcut successfully triggers the button action.  
- The underwriting details are displayed to the user after the button is pressed.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include any specific CRUD operations.

# Agent Code Validation and Details Retrieval

Type: AGENTS

Detailed description: As a user, I want to validate the agent code entered in the system to ensure it is correct and retrieve associated agent details, so that I can maintain accurate and up-to-date agent information.  
  
Acceptance criteria:  
1. When an agent code is entered, the system should check if the code is not null.  
2. The system should verify if the agent code matches specific criteria in the `azbj\_system\_constants` table for 'WEB\_AGGR' and 'IN\_HOUSE' types.  
3. If the agent code is valid, the system should retrieve the agent's unique code and internal ID from the `dmt\_agents` table.  
4. The system should fetch the agent's details from the `CP\_PARTNERS` table using the unique code.  
5. The agent's name should be constructed from the retrieved details and displayed.  
6. The system should retrieve the agent's license number, branch code, and license expiration date from the `azbj\_agents\_ext` table using the internal ID.  
7. If no data is found for the agent, an error message should be displayed, and the agent code should be cleared.  
8. The system should determine the unit manager for the agent based on specific conditions and retrieve the manager's name from the `DM\_V\_AGENT\_ASSIGNMENTS` table.  
9. If the agent code starts with '100%', the system should set a specific property for the agent.  
10. The form status should be updated to 'Y' upon successful validation.  
  
Definition of Done:  
- The agent code validation logic is implemented and tested.  
- The system retrieves and displays the correct agent details based on the entered agent code.  
- Error handling is in place for invalid or non-existent agent codes.  
- The unit manager determination logic is implemented and tested.  
- The form status is updated correctly after validation.  
  
DB queries for Table reference CRUD operations only(With Usage):  
1. Check WEB\_AGGR and IN\_HOUSE criteria:  
 ```sql  
 SELECT 'Y'  
 INTO pk\_vars.v\_web\_aggr  
 FROM azbj\_system\_constants  
 WHERE sys\_type = 'WEB\_AGGR' AND sys\_code = 'WEB\_AGGR\_DISC'  
 AND :agents.ag\_agent\_code LIKE char\_value || '%' AND ROWNUM = 1;  
 ```  
  
 ```sql  
 SELECT 'Y'  
 INTO pk\_vars.v\_inhouse\_flag  
 FROM azbj\_system\_constants  
 WHERE sys\_type = 'IN\_HOUSE' AND sys\_code = 'IN\_HOUSE\_AGENTS'  
 AND :agents.ag\_agent\_code LIKE char\_value || '%' AND ROWNUM = 1;  
 ```  
  
2. Retrieve agent unique code and internal ID:  
 ```sql  
 SELECT CUST\_PART\_UNIQUE\_CODE, int\_id  
 INTO AZBJ\_MC\_PART\_ID, azbj\_mn\_int\_id  
 FROM dmt\_agents  
 WHERE reference\_code = :ag\_agent\_code;  
 ```  
  
3. Fetch agent details from CP\_PARTNERS:  
 ```sql  
 SELECT   
 INTO AZBJ\_MC\_PARTNER  
 FROM CP\_PARTNERS  
 WHERE CP\_PARTNERS.PARTNER\_REF = AZBJ\_MC\_PART\_ID;  
 ```  
  
4. Retrieve agent's license details:  
 ```sql  
 SELECT licence\_no, branch\_code, license\_end\_date  
 INTO :ag\_agent\_lic\_no, :ag\_branch\_code, :ag\_lic\_exp\_date  
 FROM azbj\_agents\_ext  
 WHERE int\_id = azbj\_mn\_int\_id;  
 ```  
  
5. Determine unit manager and retrieve manager's name:  
 ```sql  
 SELECT PARTNER\_NAME  
 INTO :agents.ag\_unit\_mgr\_name  
 FROM DM\_V\_AGENT\_ASSIGNMENTS  
 WHERE REFERENCE\_CODE = :agents.ag\_unit\_mgr  
 AND :CONTROL.CN\_EFFECTIVE\_DATE >= START\_DATE  
 AND :CONTROL.CN\_EFFECTIVE\_DATE <= NVL(END\_DATE, :CONTROL.CN\_EFFECTIVE\_DATE);  
 ```

# View Document for Selected Policy and Document Type

Type: AGENTS

Title: View Document for Selected Policy and Document Type  
  
Acceptance Criteria:  
1. If the document type is not selected, the system should display an error message prompting the user to select a document type.  
2. The system should generate a URL for the document using the policy reference and document type.  
3. If the URL is successfully generated, the system should open the document in a web browser.  
4. If there is an error in generating the URL, the system should display a warning message with the error details.  
  
Definition of Done:  
- The user can select a document type from a predefined list.  
- The system validates that a document type is selected before attempting to generate the URL.  
- The system generates the URL and opens the document in a web browser if the URL is valid.  
- Appropriate error and warning messages are displayed as per the acceptance criteria.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Query to fetch document types for selection:  
 ```sql  
 SELECT DOC\_TYPE, DOC\_DESC  
 FROM azbj\_dms\_doc\_type\_master  
 ORDER BY SR\_NO;  
 ```

# Thumb Impression Checkbox Functionality

Type: AGENTS

Title: Thumb Impression Checkbox Functionality  
  
Acceptance Criteria:  
- The checkbox should have two states: checked (Y) and unchecked (N).  
- The checkbox should be enabled and allow both insert and update operations.  
- When the checkbox is checked, the system should set a control status to 'Y'.  
  
Definition of Done:  
- The checkbox is visible and functional on the user interface.  
- The checkbox correctly reflects the thumb impression status of the policy holder.  
- The control status is updated to 'Y' when the checkbox is checked.  
- The checkbox is styled with a gray background, black foreground, and uses the Tahoma font in bold.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include specific database queries.

# Navigate to specific item on button press

Type: AGENTS

Title: Navigate to specific item on button press  
  
Acceptance Criteria:  
- When the "Premium Recd." button is pressed, the system should navigate to the item associated with 'susac.dummy1'.  
  
Definition of Done:  
- The button labeled "Premium Recd." is visible on the "Policy Holder" tab.  
- Pressing the "Premium Recd." button successfully navigates the user to the item associated with 'susac.dummy1'.  
- The navigation should be seamless and without errors.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include any CRUD operations or SQL queries.

# Document Type Selection

Type: AGENTS

Title: Document Type Selection  
  
Acceptance Criteria:  
1. When the user double-clicks on the "Document Type" field, a list of document types should be displayed.  
2. The list of document types should be fetched from the database using the query:   
 ```sql  
 SELECT DOC\_TYPE, DOC\_DESC  
 FROM azbj\_dms\_doc\_type\_master  
 ORDER BY SR\_NO;  
 ```  
3. The list should be displayed in a pop-up window with a width of 360 and a height of 180.  
4. The "Document Type" field should be a text input field with a maximum length of 250 characters.  
5. The field should be positioned at coordinates (575, 5) on the form and should have a width of 135 and a height of 20.  
6. The field should be labeled "Document Type" with the label positioned appropriately.  
  
Definition of Done:  
- The "Document Type" field is implemented and functional.  
- Double-clicking on the field displays a list of document types fetched from the database.  
- The list of document types is displayed in a pop-up window with the specified dimensions.  
- The field and its label are positioned correctly on the form.  
- The field meets all specified criteria, including maximum length and label text.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- The query to fetch document types is:  
 ```sql  
 SELECT DOC\_TYPE, DOC\_DESC  
 FROM azbj\_dms\_doc\_type\_master  
 ORDER BY SR\_NO;  
 ```

# Validate BI Received Date against BI Date

Type: AGENTS

Title: Validate BI Received Date against BI Date  
  
Acceptance Criteria:  
1. If the "BI Received Date" is not null and the "BI Date" is not null, the system should check if the "BI Received Date" is earlier than the "BI Date".  
2. If the "BI Received Date" is earlier than the "BI Date", the system should display an error message: "BI Received date should not be less than BI Date" and the focus should remain on the "BI Received Date" field.  
3. If any exception occurs during this validation, the "BI Date" should be set to null.  
  
Definition of Done:  
- The validation logic is implemented and tested.  
- The error message is displayed correctly when the "BI Received Date" is earlier than the "BI Date".  
- The focus remains on the "BI Received Date" field if the validation fails.  
- The "BI Date" is set to null in case of any exceptions during validation.  
- All acceptance criteria are met and verified through testing.

# Manage Assignment Flag for Agents

Type: AGENTS

Title: Manage Assignment Flag for Agents  
  
Acceptance Criteria:  
1. When the assignment flag is unchecked, the system should prompt the user with a confirmation message: "Assignment flag is marked at Scrutiny/DEQC level, do you really want to remove the flag?".  
2. If the user confirms the removal of the flag, the assignment status should remain unchecked. If the user cancels, the assignment status should revert to checked.  
3. If the assignment flag is checked and the policyholder has opted for an Electronic Insurance Account (EIA), the system should automatically uncheck the assignment flag and display a message: "PH has opted for Electronic insurance account and Assignment flag also checked. Please take confirmation from PH as eIA facility is not allowed for assignment cases".  
  
Definition of Done:  
- The assignment flag functionality is implemented and tested.  
- The confirmation and error messages are displayed as per the specified conditions.  
- The assignment status is correctly updated based on user input and policyholder's EIA status.  
- 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 specific database queries.

# Notify user when PAN card verification status is 'Not Verified'

Type: AGENTS

Title: Notify user when PAN card verification status is 'Not Verified'  
  
Acceptance Criteria:  
1. When the PAN card verification status is changed to 'Not Verified', the system should display a message indicating that the PAN card has been received but not verified.  
2. The message should instruct the user to raise the FRAR again.  
3. The system should handle any exceptions gracefully without crashing.  
  
Definition of Done:  
1. The user is notified with a clear message when the PAN card verification status is 'Not Verified'.  
2. The message instructs the user to raise the FRAR again.  
3. The system handles exceptions without causing a failure.  
  
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.

# Automatic Update of 'Referred to CMO' Status Based on Medical Requirements

Type: AGENTS

Title: Automatic Update of 'Referred to CMO' Status Based on Medical Requirements  
  
Acceptance Criteria:  
1. When the 'Referred to CMO' checkbox is changed:  
 - The system should navigate to the 'MED\_UW' section and check each record.  
 - If any record in 'MED\_UW' does not have the medical requirements received, the 'Referred to CMO' status should be set to 'N' (No).  
 - If all records in 'MED\_UW' have the medical requirements received, the 'Referred to CMO' status should be set to 'Y' (Yes).  
 - If the medical requirements are not received, an error message should be displayed: 'Medical requirements are not received, Kindly check..!'  
  
2. When validating the 'Referred to CMO' checkbox:  
 - If the checkbox is set to 'Y' (Yes), the 'SAVE\_EXIT' action in the 'END\_MOVEMENT1' section should be enabled.  
 - If the checkbox is set to 'N' (No), the 'SAVE\_EXIT' action in the 'END\_MOVEMENT1' section should be disabled.  
  
Definition of Done:  
- The system correctly updates the 'Referred to CMO' status based on the receipt of medical requirements.  
- The appropriate error message is displayed when medical requirements are not received.  
- The 'SAVE\_EXIT' action is enabled or disabled based on the 'Referred to CMO' status.  
- All changes are tested and verified to work as expected.

# Dynamic Question Generation Based on Checkbox Selection

Type: AGENTS

Title: Dynamic Question Generation Based on Checkbox Selection  
  
Acceptance Criteria:  
1. When the checkbox is selected, the system should:  
 - Calculate the age of the insured person based on their date of birth and the inception date or effective date.  
 - Navigate to the questions block and clear any existing data.  
 - Populate the questions block with questions from the database, ordered by question number.  
 - Customize the question descriptions based on the insured person's details, such as age, weight, education, employment status, and annual income.  
 - Retrieve and display any previously saved answers for each question.  
  
2. When the checkbox is deselected, the system should:  
 - Navigate to the questions block and clear any existing data.  
  
Definition of Done:  
- The system correctly calculates and displays the insured person's age.  
- The questions block is populated with the correct questions and customized descriptions.  
- The insured person's details are accurately retrieved and displayed in the questions.  
- Any previously saved answers are correctly retrieved and displayed.  
- The questions block is cleared when the checkbox is deselected.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Retrieve questions for the questions block:  
 ```sql  
 SELECT FROM AZBJ\_questions\_FOR\_BBU ORDER BY q\_no;  
 ```  
  
- Retrieve education details:  
 ```sql  
 SELECT EDUCATION INTO EDU FROM AZBJ\_PARTNER\_EXTN WHERE PART\_ID = :insured\_person.ip\_part\_id;  
 ```  
  
- Retrieve screen value for education:  
 ```sql  
 SELECT screen\_value INTO EDU1 FROM inf\_dnm\_poplists WHERE poplist\_code='EDUCATION' AND internal\_value = EDU;  
 ```  
  
- Retrieve employment status:  
 ```sql  
 SELECT screen\_value INTO occ FROM inf\_dnm\_poplists WHERE poplist\_code='EMPLOYMENT\_STATUS' AND internal\_value = :insured\_person.IP\_OCC\_STATUS;  
 ```  
  
- Retrieve previously saved answers:  
 ```sql  
 SELECT answer INTO :bbu\_q.q\_bbu FROM AZBJ\_answer\_FOR\_BBU WHERE contract\_id = :control.cn\_contract\_id AND q\_no = i.q\_no;  
 ```

# Implement Dropdown Menu for Image Selection

Type: AGENTS

Title: Implement Dropdown Menu for Image Selection  
  
User Story:  
As a user, I want to view a dropdown menu that lists available images so that I can select an image from the options provided.  
  
Acceptance Criteria:  
1. The dropdown menu should display a list of images.  
2. The dropdown menu should have a maximum length of 300 characters.  
3. The dropdown menu should be positioned at coordinates (451, 7) on the screen.  
4. The dropdown menu should have a width of 90 units and a height of 23 units.  
5. The dropdown menu should use the "Tahoma" font with a size of 8 points, plain style, and demilight weight.  
6. The background color of the dropdown menu should be white, and the foreground color should be black.  
7. The dropdown menu should display one item at a time.  
  
Definition of Done:  
1. The dropdown menu is implemented and displays the list of images.  
2. The dropdown menu meets all specified acceptance criteria.  
3. The dropdown menu is tested and verified to function correctly.  
4. The user story is reviewed and approved.  
  
SQL Query for Reference:  
- Not applicable as the provided XML content does not include any specific database queries or table references.

# Implement Signature Decision List

Type: AGENTS

Title: Implement Signature Decision List  
  
Acceptance Criteria:  
1. The decision options for the signature status should be presented as a list with three elements.  
2. The decision list should be visible and enabled for user interaction.  
3. The decision list should be positioned appropriately within the user interface.  
4. The decision list should be labeled "Signature Decision" and should be clearly visible to the user.  
  
Definition of Done:  
1. The decision list is implemented and displays three options for the signature status.  
2. The decision list is functional and allows users to select an option.  
3. The decision list is correctly positioned and labeled within the user interface.  
4. The feature has been tested and verified to meet the acceptance criteria.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include any specific database queries.

# Navigate to Incomplete Proposal Details

Type: AGENTS

Detailed description: As a user, I want to view and navigate to the incomplete proposal details section when I press the "Incomplete PF Details" button, so that I can review and manage incomplete proposals efficiently.  
  
Acceptance criteria:  
1. When the "Incomplete PF Details" button is pressed, the system should display the view for incomplete proposal details.  
2. The system should navigate to the block containing the incomplete proposal details.  
  
Definition of Done:  
1. The "Incomplete PF Details" button is functional and triggers the display of the incomplete proposal details view.  
2. The system successfully navigates to the block containing the incomplete proposal details upon button press.  
3. The user can see and interact with the incomplete proposal details section.  
  
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 Manage Underwriting Limits

Type: AGENTS

Title: View and Manage Underwriting Limits  
  
Acceptance Criteria:  
1. When the "Underwriting Limits" button is pressed, the system should display the underwriting limits view.  
2. The system should navigate to the underwriting limits section.  
3. The system should retrieve and display the following details for each user from the `user\_uw\_limits` table:  
 - User ID  
 - Underwriting Code  
 - Underwriting Description  
 - Minimum Success Rate  
 - Maximum Success Rate  
 - Minimum Age  
 - Maximum Age  
 - Minimum Code Value  
 - Maximum Code Value  
 - Start Date  
 - End Date  
4. The system should handle any errors that occur during the retrieval process and display an appropriate error message.  
  
Definition of Done:  
- The "Underwriting Limits" button is functional and triggers the display of the underwriting limits view.  
- The system successfully navigates to the underwriting limits section.  
- The system retrieves and displays the required details from the `user\_uw\_limits` table for each user.  
- Error handling is implemented, and appropriate error messages are displayed when necessary.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- The following query is used to retrieve user underwriting limits:  
 ```sql  
 SELECT user\_id, uw\_code, uw\_desc, min\_suc, max\_suc, min\_age, max\_age,   
 min\_code\_val, max\_code\_val, start\_date, end\_date  
 FROM user\_uw\_limits  
 WHERE user\_id = :BOILER.USERID;  
 ```

# Check and Display Duplicate Contact Number

Type: AGENTS

Title: Check and Display Duplicate Contact Number  
  
Acceptance Criteria:  
1. When the "Duplicate Contact No." button is pressed, the system should check if a duplicate contact number is found.  
2. If a duplicate contact number is found, the system should display the "Duplicate Contact No." view and navigate to the "Duplicate Contact No." section.  
3. If no duplicate contact number is found, the system should display a message indicating that no duplicate contact number was found.  
  
Definition of Done:  
- The "Duplicate Contact No." button functionality is implemented.  
- The system correctly checks for duplicate contact numbers.  
- The appropriate view is displayed if a duplicate is found.  
- An error message is shown if no duplicate is found.  
- All acceptance criteria are met and tested.  
  
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 Image Functionality

Type: AGENTS

Title: View Image Functionality  
  
Acceptance Criteria:  
1. When the "View Image" button is pressed, the system should retrieve the image path and proposal number from the relevant data fields.  
2. The system should construct a client path using the proposal number and image list.  
3. The system should transfer the image file from the server to the client machine, displaying a progress message during the transfer.  
4. If the image file is in PDF format, it should be opened using the default PDF viewer.  
5. If the image file is in another format, it should be opened using the default image viewer.  
6. The system should handle any exceptions that occur during the process without crashing.  
  
Definition of Done:  
- The "View Image" button functionality is implemented and tested.  
- The system successfully retrieves and constructs the client path.  
- The image file is transferred and opened correctly based on its format.  
- Exception handling is in place to manage errors gracefully.  
- The feature is documented and reviewed.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Not applicable as the provided logic does not include direct CRUD operations on the database.

# Dynamic Visibility and Enablement of Commission Field Based on IC's Own Policy Checkbox

Type: AGENTS

Title: Dynamic Visibility and Enablement of Commission Field Based on IC's Own Policy Checkbox  
  
Acceptance Criteria:  
1. When the "IC's Own Policy" checkbox is checked:  
 - The commission field should become visible.  
 - The commission field should be enabled for user input.  
2. When the "IC's Own Policy" checkbox is unchecked:  
 - The commission field should be hidden.  
 - The commission field should be disabled to prevent user input.  
  
Definition of Done:  
- The commission field visibility and enablement should correctly toggle based on the state of the "IC's Own Policy" checkbox.  
- The changes should be tested to ensure they work as expected in various scenarios.  
- 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 XML content does not include any direct database queries.

# Verify and Display Specified Person/IC Signature

Type: AGENTS

Detailed description: As a user, I want to verify the signature of a specified person or IC (Insurance Consultant) and display it within the system, so that I can ensure the authenticity of the signature for compliance and verification purposes.  
  
Acceptance criteria:  
1. When the "Match Specified Person / IC Signature" button is pressed, the system should:  
 - Set the `v\_match\_ic\_sig` variable to 'Y'.  
 - Set the `v\_sign\_matched` control variable to 'Y'.  
 - Clear the `SIGN\_DECISION` field if `v\_sign\_matched` is 'Y'.  
 - Check if a signature exists for the specified person and IC code in the `azbj\_signature` table.  
 - Check if the agent's signature is mandatory based on the `azbj\_system\_constants` table.  
 - If the signature is mandatory and not bypassed, retrieve the signature image from the database and display it in the `SIGNATURE\_IN\_SYSTEM` field.  
 - If the agent code starts with '1' or '23', check for the signature in the `balic.ddt\_704` table and display it using a URL if available.  
 - If no signature is found, display a warning message indicating that the signature is not available.  
  
Definition of Done:  
- The button functionality is implemented and tested.  
- The system correctly retrieves and displays the signature image from the database.  
- Appropriate messages are displayed when the signature is not available.  
- The functionality is verified to work for different agent codes and conditions as specified.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Retrieve count of signatures for specified person and IC code:  
 ```sql  
 SELECT COUNT(1)  
 INTO v\_count  
 FROM azbj\_signature  
 WHERE sub\_ic\_code = :agents.specified\_person\_sub\_ic\_code  
 AND ic\_code = :agents.ag\_agent\_code;  
 ```  
  
- Check if agent's signature is mandatory:  
 ```sql  
 SELECT COUNT(1)  
 INTO v\_ic\_count  
 FROM azbj\_system\_constants  
 WHERE sys\_type = 'AGENT\_SIGN'  
 AND sys\_code = 'AGENT\_SIGN\_MANDATORY'  
 AND :agents.ag\_agent\_code LIKE char\_value || '%';  
 ```  
  
- Check if agent's signature is bypassed:  
 ```sql  
 SELECT COUNT(1)  
 INTO v\_agnet\_sign\_bypass  
 FROM azbj\_sign\_bypass\_agents  
 WHERE agent\_code = :agents.ag\_agent\_code  
 AND by\_pass\_flag = 'Y';  
 ```  
  
- Retrieve application number for the agent:  
 ```sql  
 SELECT APPLICATION\_NO  
 INTO v\_agent\_appln\_no  
 FROM azbj\_ic\_recruitement\_dtls  
 WHERE IC\_CODE = :agents.ag\_agent\_code  
 AND modified\_date = (SELECT MAX(modified\_date)  
 FROM azbj\_ic\_recruitement\_dtls  
 WHERE ic\_code = :agents.ag\_agent\_code);  
 ```  
  
- Check for signature in `balic.ddt\_704` table:  
 ```sql  
 SELECT COUNT()  
 INTO v\_omni\_count  
 FROM balic.ddt\_704  
 WHERE UPPER(field\_1209) = 'SIGNATURE'  
 AND field\_1206 = :agents.ag\_agent\_code;  
 ```

# Customer Selection List in New Business Module

Type: AGENTS

Title: Customer Selection List in New Business Module  
  
Acceptance Criteria:  
1. The customer selection list should contain at least three predefined customer options.  
2. The customer selection list should be displayed on the "AML" tab of the "New Business" module.  
3. The customer selection list should be hidden by default and only become visible when required for data entry.  
4. The selected customer information should be stored and associated with the business transaction.  
  
Definition of Done:  
1. The customer selection list is implemented and contains the predefined customer options.  
2. The list is correctly displayed on the "AML" tab and is hidden by default.  
3. The selected customer information is correctly stored and associated with the business transaction.  
4. All acceptance criteria are met and verified through testing.

# View Signature Confidence Details

Type: AGENTS

Detailed description: As a user, I want to view the signature confidence details for a specific application, so that I can assess the confidence level of the signatures associated with the application.  
  
Acceptance criteria:  
1. When the "Signature Confidence Details" button is pressed, the system should navigate to the "Signature Confidence" section.  
2. The system should clear any existing data in the "Signature Confidence" section.  
3. The system should retrieve the signature names and their corresponding confidence levels from the database for the given application number.  
4. The application number should be determined based on the insured person's verification number or signature card number.  
5. The retrieved data should be displayed in the "Signature Confidence" section, with each record showing the document name and its confidence level.  
6. The system should navigate to the first record after displaying the data.  
  
Definition of Done:  
- The "Signature Confidence Details" button should be functional and trigger the described behavior.  
- The "Signature Confidence" section should be populated with the correct data from the database.  
- The data retrieval should be based on the insured person's verification number or signature card number.  
- The user should be able to see the document names and their confidence levels in the "Signature Confidence" section.  
- The system should navigate to the first record after displaying the data.  
  
DB queries for Table reference CRUD operations only(With Usage):  
```sql  
SELECT SIGNATURENAME, TotalConf  
FROM balic.ng\_asv\_results  
WHERE APPLICATIONNO = TO\_CHAR(NVL(:insured\_person.ip\_verf\_no, :insured\_person.ip\_sign\_card\_no));  
```

# PAN Card Number Validation and Personal Details Retrieval

Type: AGENTS

Title: PAN Card Number Validation and Personal Details Retrieval  
  
Acceptance Criteria:  
1. When a PAN card number is entered, the system should check if the PAN card number is unique by querying the database.  
2. If the PAN card number is found in previous policies, the system should display a warning message: "Please check PAN No found in previous policies".  
3. If the PAN card number is not found in previous policies, the system should enable the "Previous Policy Details" field.  
4. The system should retrieve and display the middle name, first name, surname, and date of birth associated with the PAN card number from the database.  
5. If the PAN card number is associated with an insured person (IP), the system should update the insured person's PAN card number field.  
  
Definition of Done:  
- The PAN card number validation logic is implemented and tested.  
- The system displays appropriate messages based on the validation results.  
- The system retrieves and displays personal details associated with the PAN card number.  
- The "Previous Policy Details" field is enabled or disabled based on the validation results.  
- The insured person's PAN card number field is updated if applicable.  
  
DB queries for Table reference CRUD operations only (With Usage):  
```sql  
-- Query to check if PAN card number is unique  
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 = :agents.pan\_card  
AND a.top\_indicator = 'Y'  
AND c.part\_id <> :policy\_holder.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;  
  
-- Query to retrieve personal details associated with PAN card number  
SELECT middle\_name, first\_name, surname, DATE\_OF\_BIRTH  
INTO :agents.pc\_middle\_name, :agents.pc\_first\_name, :agents.pc\_last\_name, :agents.pc\_dob  
FROM cp\_partners  
WHERE part\_id = CASE  
 WHEN :agents.ip\_ph = 'PH' THEN :policy\_holder.ph\_part\_id  
 WHEN :agents.ip\_ph = 'IP' THEN :insured\_person.ip\_part\_id  
 ELSE :policy\_holder.ph\_part\_id  
END;  
```

# Implement Dropdown for Broker Model Selection in Agent Details

Type: AGENTS

User Story:  
As a user, I want to select the type of broker model from a predefined list when entering agent details, so that I can accurately categorize the broker model associated with each agent.  
  
Acceptance Criteria:  
1. The user should be able to see a dropdown list labeled "Type of Broker model" when entering agent details.  
2. The dropdown list should contain predefined options for broker models.  
3. The selected broker model should be saved and associated with the agent's record.  
  
Definition of Done:  
1. The dropdown list for selecting the broker model is implemented and visible in the agent details section.  
2. The predefined options for broker models are available in the dropdown list.  
3. The selected broker model is correctly saved and associated with the agent's record in the database.  
4. The feature is tested and verified to ensure it works as expected.  
  
SQL Query for Reference:  
- Not applicable as the provided XML content does not include specific SQL queries or table references.  
  
Block Name: AGENTS

# Rule Confidence Window Functionality

Type: AGENTS

Detailed description: As a user, I want to be able to open a Rule Confidence Window from the application interface, so that I can view and manage rule confidence settings for insured persons.  
  
Acceptance criteria:  
1. When the "Rule Confidence Window" button is pressed, the system should retrieve the application number based on the insured person's verification number or sign card number.  
2. The system should check if a parameter list named 'input\_params' exists and destroy it if it does.  
3. A new parameter list named 'input\_params' should be created, and the application number should be added to this list.  
4. The system should open a new form named 'AZBJ\_NEW\_RULE\_CONFIGURATOR' with the created parameter list.  
5. The system should navigate to the 'Further\_Req' section and display the last record.  
6. The system should set the message level to 25 to suppress certain messages.  
7. The system should query the 'bbu\_req\_rule\_confidence' table for specific conditions and iterate through the results.  
8. For each result, if the rule confidence function returns 'N', the system should:  
 - Navigate to the last record and then to the next record.  
 - Populate the 'Further\_Req' section with the retrieved data.  
 - Set the background color of the test code field based on specific conditions.  
 - Set additional fields such as 'fr\_ip\_type', 'fr\_date\_called', 'COMMENTS', 'RAISED\_BY', and 'FR\_DT\_CALLED'.  
9. The system should set a global flag 'rule\_confidence\_window' to 'Y'.  
10. The system should navigate to the first record and reset the message level to 0.  
  
Definition of Done:  
- The Rule Confidence Window button should function as described in the acceptance criteria.  
- The system should correctly handle the parameter list and open the new form with the appropriate parameters.  
- The 'Further\_Req' section should be populated with the correct data based on the query results.  
- The background color of the test code field should change based on the specified conditions.  
- All specified fields should be correctly populated.  
- The global flag and message levels should be set as described.  
  
DB queries for Table reference CRUD operations only(With Usage):  
```sql  
SELECT sugg\_req, req\_desc  
FROM bbu\_req\_rule\_confidence  
WHERE appln\_no = TO\_CHAR(NVL(:insured\_person.ip\_verf\_no, :insured\_person.ip\_sign\_card\_no))  
 AND sugg\_req IS NOT NULL  
 AND req\_desc IS NOT NULL  
 AND top\_indicator = 'Y'  
 AND rule\_accept IN ('YA', 'Y');  
```

# View Commission Details for Agent

Type: AGENTS

Title: View Commission Details for Agent  
  
Acceptance Criteria:  
1. When the user clicks the "IC Commission Det" button, the system should gather the agent code from the current agent record.  
2. The system should create a parameter list and add the agent code to this list.  
3. The system should then open a new form or screen displaying the commission details for the specified agent.  
  
Definition of Done:  
- The button labeled "IC Commission Det" is present and clickable.  
- Clicking the button successfully retrieves the agent code from the current record.  
- A parameter list is created and the agent code is added to this list.  
- The commission details form or screen is opened and displays the relevant information for the specified agent.  
  
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 Previous Policy Pan Details

Type: AGENTS

Detailed description: As a user, I want to view the details of the previous policy associated with a policyholder, so that I can verify and cross-check the information.  
  
Acceptance criteria:  
1. When the "Previous Policy Pan Details" button is pressed, the system should:  
 - Check if a parameter list named 'Param1' exists and destroy it if it does.  
 - Create a new parameter list named 'Param1'.  
 - Add the following parameters to the list:  
 - `PAR\_PH\_PART\_ID` with the value of the policy holder's part ID.  
 - `PAR\_PAN\_CARD\_NO` with the value of the agent's PAN card.  
 - `PAR\_MODULE` with the value 'BBU'.  
 - `PAR\_PAN\_PH\_NAME` with the value of the policy holder's name.  
 - `PAR\_PAN\_PH\_DOB` with the value of the policy holder's date of birth.  
 - Call the form 'AZBJ\_OLD\_POLICY\_DTLS' with the created parameter list.  
 - Set a global variable `v\_pan\_validated` to 'Y'.  
  
Definition of Done:  
- The button should be disabled by default.  
- The button should be labeled "Previous Policy Pan Details".  
- The button should be positioned at coordinates (634, 449) with a width of 186 and a height of 19.  
- The button should have a white background and black foreground.  
- The button should use the "MS Sans Serif" font with a size of 8, plain style, and bold weight.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- No direct database queries are mentioned in the provided XML content that can be executed independently of Oracle Forms constructs.

# View and Update Aadhaar Details

Type: AGENTS

User Story: View and Update Aadhaar Details  
  
Detailed description:   
As a user, I want to view and update Aadhaar details for the insured person and policyholder, so that I can ensure the correct Aadhaar information is associated with the policy.  
  
Acceptance criteria:  
1. When the "Aadhaar details with multiple CP" button is pressed, the system should:  
 - Check if the insured person or policyholder has provided Aadhaar as age proof.  
 - Retrieve and trim the Aadhaar numbers for both the insured person and policyholder.  
 - If the policyholder is not the same as the insured person, retrieve and trim the policyholder's Aadhaar number.  
 - If an AML (Anti-Money Laundering) record exists, use the AML Aadhaar ID.  
 - Create a parameter list with the following parameters:  
 - Insured Person's Aadhaar number  
 - Policyholder's Aadhaar number  
 - Insured Person's ID  
 - Policyholder's ID  
 - Module name  
 - Call the form to display Aadhaar details if any Aadhaar proof is provided or an AML record exists.  
 - Display a warning message if no Aadhaar proof is selected.  
  
Definition of Done:  
- The system correctly retrieves and displays Aadhaar details for the insured person and policyholder.  
- The system handles cases where no Aadhaar proof is provided by displaying an appropriate warning message.  
- The system updates the parameter list and calls the form to display Aadhaar details when applicable.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Not applicable as the provided logic involves form-specific operations and parameter handling.

# Display Suspected Cases List

Type: AGENTS

Title: Display Suspected Cases List  
  
Acceptance Criteria:  
1. The list of suspected cases should be displayed on the "New Business" tab.  
2. The list should contain 5 elements.  
3. The list should be visually distinct with a gray background and black text.  
4. The list should be positioned at the top left of the "New Business" tab.  
5. The list should be labeled "Suspected Case" and the label should be aligned to the start of the list.  
  
Definition of Done:  
1. The list of suspected cases is implemented and displayed correctly on the "New Business" tab.  
2. The list contains exactly 5 elements.  
3. The list has a gray background and black text.  
4. The list is positioned correctly as per the design specifications.  
5. The label "Suspected Case" is correctly aligned and displayed.  
  
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 BFL TA Segment Field

Type: AGENTS

Title: Implement BFL TA Segment Field  
  
Acceptance Criteria:  
- The BFL TA Segment field should display a list of 5 predefined segments.  
- The field should be located within the New Business module.  
- The field should be initially hidden and only become visible under specific conditions.  
- The field should have a maximum length of 50 characters.  
- The field should be styled with a specific font and color scheme.  
  
Definition of Done:  
- The BFL TA Segment field is implemented and displays the correct list of segments.  
- The field is correctly positioned and styled as per the requirements.  
- The field is hidden by default and becomes visible under the specified conditions.  
- The functionality is tested and verified to ensure it meets the acceptance criteria.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- No specific DB queries provided in the XML content.

# Validate Partner Type Field

Type: AGENTS

Title: Validate Partner Type Field  
  
Acceptance Criteria:  
1. When the validation flag is set to 'Y', 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 prompting the user to enter a partner type.  
  
Definition of Done:  
1. The validation logic is implemented and tested.  
2. The system displays the appropriate warning message when the "Partner Type" field is empty and the validation flag is 'Y'.  
3. 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 database-specific queries.

# Manage Aadhar Information for Agents

Type: AGENTS

Title: Manage Aadhar Information for Agents  
  
Acceptance Criteria:  
1. The Aadhar information input should be available as a radio group option within the "AML" tab.  
2. The input field for Aadhar information should have a width of 100 units and a height of 21 units.  
3. The Aadhar information should be displayed on a tabbed canvas named "NBTABS" with a gray background and black foreground.  
4. The tabbed canvas should have a width of 1076 units and a height of 980 units.  
5. The window displaying the Aadhar information should be titled "New Business" and should have a width of 1023 units and a height of 550 units.  
6. The window should not show a horizontal scrollbar but should display a vertical scrollbar.  
  
Definition of Done:  
1. The Aadhar information input field is implemented as a radio group within the "AML" tab.  
2. The input field dimensions and display properties match the specified criteria.  
3. The tabbed canvas and window properties are correctly configured to display the Aadhar information.  
4. The user interface is tested to ensure that the Aadhar information is captured and displayed correctly without any visual or functional issues.

# Dynamic Adjustment of PAN Card Fields Based on Checkbox Status

Type: AGENTS

Title: Dynamic Adjustment of PAN Card Fields Based on Checkbox Status  
  
Acceptance Criteria:  
1. When the "Pan Card not Available" checkbox is checked and the `phonepe\_flag` is 'N':  
 - The "PH\_NO\_PAN\_LOV" field should become visible and enabled.  
 - The "PAN\_CARD" field should become disabled and its value should be cleared.  
 - The "ST\_CUSTOMER" field should become visible and enabled if the mailing state is one of the specified states (Arunachal Pradesh, Manipur, Mizoram, Assam, Nagaland, Meghalaya).  
2. When the "Pan Card not Available" checkbox is unchecked:  
 - The "PH\_NO\_PAN\_LOV" field should become invisible and its value should be cleared.  
 - The "PAN\_CARD" field should become enabled.  
 - The "ST\_CUSTOMER" field should become invisible.  
  
Definition of Done:  
- The form dynamically adjusts the visibility and enablement of the fields based on the checkbox status and other specified conditions.  
- The changes are tested and verified to ensure they meet the acceptance criteria.  
- The implementation is reviewed and approved by the stakeholders.

# Validate Channel Field in Agents Section

Type: AGENTS

Detailed description: As a user, I want the system to validate the "Channel" field in the "Agents" section to ensure that the channel information is accurate and visually indicate the validation status.  
  
Acceptance criteria:  
1. When the "Channel" field is not empty, the system should:  
 - Query the `azbj\_batch\_items` table to retrieve the channel based on the `application\_no` derived from either `ip\_verf\_no` or `ip\_sign\_card\_no`, where `transaction\_type` is 'FRP' and `PRINT` is not 'C'.  
 - If no result is found, query the `azbj\_phub\_scrutiny\_prop\_extn` table to retrieve the channel based on the same `application\_no`.  
 - Compare the retrieved channel with the value in the "Channel" field.  
 - If the retrieved channel does not match the value in the "Channel" field, visually indicate this by setting the field's visual attribute to red.  
 - If the retrieved channel matches the value in the "Channel" field, visually indicate this by setting the field's visual attribute to green.  
  
Definition of Done:  
- The "Channel" field validation logic is implemented as described.  
- The system correctly queries the relevant tables and handles exceptions.  
- The visual indication (red or green) is correctly applied based on the validation result.  
- The feature is tested and verified to work as expected.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Query to retrieve channel from `azbj\_batch\_items`:  
 ```sql  
 SELECT channel  
 INTO v\_channel  
 FROM azbj\_batch\_items  
 WHERE application\_no = TO\_CHAR(NVL(:insured\_person.ip\_verf\_no, :insured\_person.ip\_sign\_card\_no))  
 AND transaction\_type = 'FRP'  
 AND NVL(PRINT, 'X') <> 'C';  
 ```  
  
- Query to retrieve channel from `azbj\_phub\_scrutiny\_prop\_extn`:  
 ```sql  
 SELECT channel  
 INTO v\_channel  
 FROM azbj\_phub\_scrutiny\_prop\_extn  
 WHERE application\_no = TO\_CHAR(NVL(:insured\_person.ip\_verf\_no, :insured\_person.ip\_sign\_card\_no));  
 ```

# Validate Aadhaar Number

Type: AGENTS

Title: Validate Aadhaar Number  
  
Acceptance Criteria:  
1. The Aadhaar number must be exactly 12 digits long.  
2. If the Aadhaar number is not 12 digits long, an error message should be displayed: "Kindly Enter Valid Aadhar Card No!".  
3. The system should not proceed with the form submission if the Aadhaar number is invalid.  
  
Definition of Done:  
1. The validation logic for the Aadhaar number is implemented.  
2. The error message is displayed correctly when the Aadhaar number is invalid.  
3. The form submission is halted if the Aadhaar number is not 12 digits long.  
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 CRUD operations.

# Validate Age, Location, and PAN Card Information

Type: AGENTS

Title: Validate Age, Location, and PAN Card Information  
  
Acceptance Criteria:  
1. The system should calculate the age of the insured person and policyholder based on their date of birth and the current date.  
2. If the policyholder's mailing or residential district is 'LEH' and the state is 'JAMMU AND KASHMIR', a flag should be set.  
3. If the total premium or the insured person's annual income exceeds specified thresholds, the system should check if the policyholder has a PAN card.  
4. If the PAN card is not available, the system should display an error message prompting the user to select an appropriate option from the list of values (LOV).  
5. The system should handle various conditions based on the age and nationality of the insured person and policyholder, and display appropriate messages or set specific customer status values.  
6. The system should ensure that the selected LOV for the PAN card not received flag is not null and display an error message if it is.  
  
Definition of Done:  
- The age calculation logic for the insured person and policyholder is implemented and tested.  
- The flag for the policyholder's location is correctly set based on the specified conditions.  
- The system checks the total premium and annual income thresholds and validates the presence of a PAN card.  
- Appropriate error messages are displayed when the PAN card is not available, and the user is prompted to select an option from the LOV.  
- The system handles all specified conditions related to age, nationality, and customer status, and displays the correct messages.  
- The LOV selection for the PAN card not received flag is validated to ensure it is not null.  
- All acceptance criteria are met, and the functionality is tested and verified.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Not applicable as the provided XML content does not include specific CRUD operations or direct database queries.

# Validate Aadhaar Number

Type: AGENTS

Title: Validate Aadhaar Number  
  
Acceptance Criteria:  
1. The Aadhaar number must be exactly 12 digits long.  
2. If the Aadhaar number is not 12 digits long, an error message should be displayed: "Kindly Enter Valid Aadhar Card No!!".  
3. The system should validate the Aadhaar number only if a specific validation flag is set to 'Y'.  
  
Definition of Done:  
1. The system checks the length of the Aadhaar number upon entry.  
2. An error message is displayed if the Aadhaar number is not 12 digits long.  
3. The validation logic is executed only when the validation flag is set to 'Y'.  
4. The error message is displayed in a user-friendly manner.  
  
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.

# Manage Aadhar Information for Agents

Type: AGENTS

Title: Manage Aadhar Information for Agents  
  
Acceptance Criteria:  
1. The Aadhar information should be entered through a radio group interface.  
2. The interface should be displayed on a tab within the New Business module.  
3. The tab should have a gray background and black foreground, with a width of 1076 pixels and a height of 980 pixels.  
4. The window displaying this information should have a title "New Business" and should be 1023 pixels wide and 550 pixels high.  
5. The window should have a vertical scrollbar but no horizontal scrollbar.  
  
Definition of Done:  
1. The Aadhar information input interface is implemented and accessible within the New Business module.  
2. The interface meets the specified design and layout criteria.  
3. The functionality is tested and verified to ensure that Aadhar information can be accurately entered and managed.  
4. The user interface is intuitive and user-friendly.  
5. All acceptance criteria are met without any Oracle Forms-specific terminology or dependencies.  
  
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.

# Input and Validate Aadhaar Number

Type: AGENTS

Title: Input and Validate Aadhaar Number  
  
Acceptance Criteria:  
1. The Aadhaar number field should accept a maximum of 12 characters.  
2. The field should only accept uppercase characters.  
3. The field should be editable and allow new entries.  
4. The field should be validated upon entry to ensure it meets the required format.  
  
Definition of Done:  
1. The Aadhaar number field is implemented and visible on the user interface.  
2. The field accepts a maximum of 12 characters and converts all input to uppercase.  
3. The field allows users to input new Aadhaar numbers and update existing ones.  
4. Validation logic is in place to ensure the Aadhaar number meets the required format.  
5. The field is tested and confirmed 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 database queries.

# Validate and Update Sub Id Code

Type: AGENTS

Detailed description: As a user, I want to validate the "Sub Id Code" input to ensure it is correct and update the associated fields and visual attributes accordingly.  
  
Acceptance criteria:  
1. When the "Sub Id Code" is not null, the system should:  
 - Check if the "Sub Id Code" exists in the `azbj\_v\_agents` table and the agent status is not 'T' or 'TE'.  
 - If the "Sub Id Code" is invalid, display an error message "Invalid Sub Id Code!".  
 - If valid, fetch and display the full name associated with the "Sub Id Code" from the `azbj\_v\_agents` table.  
 - Attempt to fetch the `sub\_id` from the `azbj\_batch\_items` table using the `application\_no` derived from either `ip\_verf\_no` or `ip\_sign\_card\_no` fields.  
 - If not found, attempt to fetch the `sub\_id\_code` from the `azbj\_phub\_scrutiny\_prop\_extn` table using the same `application\_no`.  
 - Compare the fetched `sub\_id` or `sub\_id\_code` with the input "Sub Id Code":  
 - If they do not match, set the visual attribute to red.  
 - If they match, set the visual attribute to green.  
2. Fetch the `channel\_code` associated with the `ag\_agent\_code` from the `azbj\_v\_agents` table.  
3. Clear and repopulate the "CHANNEL" list based on the `channel\_code` and the substring of the "Sub Id Code".  
  
Definition of Done:  
- The "Sub Id Code" validation logic is implemented and tested.  
- Error messages and visual attributes are correctly displayed based on the validation results.  
- The "CHANNEL" list is dynamically updated based on the `channel\_code` and "Sub Id Code".  
- All functionalities are tested and verified to work as expected.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Check if "Sub Id Code" exists and fetch full name:  
 ```sql  
 SELECT COUNT()  
 INTO v\_valid\_cnt  
 FROM azbj\_v\_agents  
 WHERE reference\_code = UPPER(:agents.sub\_id\_code)  
 AND agent\_status NOT IN ('T', 'TE');  
 ```  
  
 ```sql  
 SELECT full\_name  
 INTO :agents.sub\_id\_name  
 FROM azbj\_v\_agents  
 WHERE reference\_code = UPPER(:agents.sub\_id\_code)  
 AND agent\_status NOT IN ('T', 'TE');  
 ```  
  
- Fetch `sub\_id` from `azbj\_batch\_items`:  
 ```sql  
 SELECT sub\_id  
 INTO v\_sub\_id  
 FROM azbj\_batch\_items  
 WHERE application\_no = TO\_CHAR(NVL(:insured\_person.ip\_verf\_no, :insured\_person.ip\_sign\_card\_no))  
 AND transaction\_type = 'FRP'  
 AND NVL(PRINT, 'X') <> 'C';  
 ```  
  
- Fetch `sub\_id\_code` from `azbj\_phub\_scrutiny\_prop\_extn`:  
 ```sql  
 SELECT sub\_id\_code  
 INTO v\_sub\_id  
 FROM azbj\_phub\_scrutiny\_prop\_extn  
 WHERE application\_no = TO\_CHAR(NVL(:insured\_person.ip\_verf\_no, :insured\_person.ip\_sign\_card\_no));  
 ```  
  
- Fetch `channel\_code` from `azbj\_v\_agents`:  
 ```sql  
 SELECT channel\_code  
 INTO v\_channel  
 FROM azbj\_v\_agents  
 WHERE reference\_code = UPPER(:agents.ag\_agent\_code);  
 ```  
  
- Populate "CHANNEL" list:  
 ```sql  
 SELECT ROWNUM AS rn, UPPER(vertical\_id) display\_val, UPPER(vertical\_id) internal\_val  
 FROM azbj\_sub\_id\_mapping  
 WHERE channel\_code = v\_channel  
 AND sub\_id\_code = SUBSTR(UPPER(:agents.sub\_id\_code), 3, 2);  
 ```

# Input and Validate Aadhaar Number

Type: AGENTS

Detailed description: As a user, I want to input and validate a new Aadhaar number in the system, ensuring that the data is stored correctly and adheres to the specified format and length constraints.  
  
Acceptance criteria:  
1. The Aadhaar number field should accept a maximum of 12 characters.  
2. The field should only accept uppercase characters.  
3. The field should be editable and allow new entries.  
4. The field should be validated upon entry to ensure it meets the required format.  
  
Definition of Done:  
1. The Aadhaar number field is implemented and visible on the user interface.  
2. The field accepts a maximum of 12 characters and converts all input to uppercase.  
3. The field allows users to insert and update the Aadhaar number.  
4. Validation logic is in place to ensure the Aadhaar number meets the required format.  
5. The field is tested and confirmed to work as expected in various scenarios.  
  
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: AGENTS

Title: Exit Button Functionality  
  
Acceptance Criteria:  
1. When the exit button is pressed, the current view should be hidden.  
2. The focus should then move to the "Add SU User" control.  
  
Definition of Done:  
- The exit button should be functional and perform the specified actions.  
- The current view should be hidden upon pressing the exit button.  
- The focus should shift to the "Add SU User" control seamlessly.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable for this user story as there are no direct database operations mentioned in the provided XML content.

# Aadhar Card Verification

Type: AGENTS

Title: Aadhar Card Verification  
  
Acceptance Criteria:  
1. When the "Aadhar Card Verification" button is pressed, the system should:  
 - Hide any previously visible radio buttons related to Aadhar details matching, not matching, or no data for both the insured person and policyholder.  
 - Prompt the user to select the partner type if not already selected.  
 - Calculate the entry age of the insured person.  
 - Check if the age proof provided is valid and complete for both the insured person and policyholder.  
 - Display radio buttons for Aadhar details matching, not matching, or no data based on the validation results.  
 - If the Aadhar number is not found in the database, display a message indicating "Data not available" for the respective person.  
 - If the Aadhar number is found, create a parameter list with the Aadhar details and call the Aadhar KYC form.  
 - Update the Aadhar details status (matched, not matched, or none) based on the global variables set by the KYC form.  
  
2. The system should handle exceptions and display an error message if there is an issue during the verification process.  
  
Definition of Done:  
- The "Aadhar Card Verification" button functionality is implemented and tested.  
- The system correctly hides and shows the appropriate radio buttons based on the validation results.  
- The system prompts the user to select the partner type if not already selected.  
- The system calculates the entry age and validates the age proof for both the insured person and policyholder.  
- The system displays the correct Aadhar details status based on the database records and KYC form results.  
- The system handles exceptions and displays error messages as needed.  
- The functionality is reviewed and approved by stakeholders.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- SELECT COUNT (1) INTO v\_cnt1 FROM balic\_aadhaar\_dtls WHERE aadhaar\_aadhaarnumber = :insured\_person.ip\_age\_prf\_id;  
- SELECT COUNT (1) INTO v\_cnt2 FROM balic\_aadhaar\_dtls WHERE aadhaar\_aadhaarnumber = :policy\_holder.ph\_age\_prf\_id;  
  
These queries are used to check if the Aadhar numbers provided by the insured person and policyholder exist in the database.

# Delete User from Senior UnderWriter List

Type: AGENTS

Title: Delete User from Senior UnderWriter List  
  
Acceptance Criteria:  
1. When the delete button is pressed, the system should check if the user ID is not null.  
2. The system should verify if the user exists in the Senior UnderWriter List by checking the `AZBJ\_SYSTEM\_CONSTANTS` table where `SYS\_TYPE` is 'SU' and `CHAR\_VALUE` matches the user ID.  
3. If the user exists, the system should prompt the user with a confirmation message asking if they want to delete the user from the Senior UnderWriter List.  
4. If the user confirms the deletion, the system should delete the user from the `AZBJ\_SYSTEM\_CONSTANTS` table where `SYS\_TYPE` is 'SU', `SYS\_CODE` is 'SENIOR\_UW', and `CHAR\_VALUE` matches the user ID.  
5. The system should commit the transaction and hide the current view.  
6. If the user does not exist in the Senior UnderWriter List, the system should display a warning message indicating that the user is not available in the list.  
  
Definition of Done:  
- The delete button functionality is implemented and tested.  
- The system correctly checks for the existence of the user in the Senior UnderWriter List.  
- Appropriate confirmation and warning messages are displayed to the user.  
- The user is successfully deleted from the list if confirmed.  
- The transaction is committed, and the view is updated accordingly.  
  
DB queries for Table reference CRUD operations only(With Usage):  
```sql  
-- Check if the user exists in the Senior UnderWriter List  
SELECT COUNT()  
INTO V\_EXIST\_CNT  
FROM AZBJ\_SYSTEM\_CONSTANTS  
WHERE SYS\_TYPE = 'SU' AND CHAR\_VALUE = :CONTROL.SU\_USER\_ID;  
  
-- Delete the user from the Senior UnderWriter List  
DELETE FROM AZBJ\_SYSTEM\_CONSTANTS  
WHERE SYS\_TYPE = 'SU' AND SYS\_CODE = 'SENIOR\_UW' AND CHAR\_VALUE = :CONTROL.SU\_USER\_ID;  
```

# Implement EXIT button functionality

Type: AGENTS

Title: Implement EXIT button functionality  
  
Acceptance Criteria:  
- When the "EXIT" button is pressed, the current window should be hidden.  
- The view associated with the current window should also be hidden.  
- The focus should then be moved to the main screen, specifically to the "PIVC\_GEN\_LINK" item within the "AGENTS" section.  
  
Definition of Done:  
- The "EXIT" button successfully hides the current window and its associated view.  
- The focus is correctly shifted to the "PIVC\_GEN\_LINK" item on the main screen.  
- The functionality 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 any database CRUD operations.

# Display PIVC Video Link for Application

Type: AGENTS

Detailed description: As a user, I want to view a specific video link related to the PIVC process for a given application number, so that I can ensure the correct video is displayed based on the application details and eligibility criteria.  
  
Acceptance criteria:  
1. The system should retrieve the application number from either the verification number or the sign card number of the insured person.  
2. If the application number is not null, the system should check if the product ID is 321, the PB exist count is 0, and the agent code does not belong to the 'AXIS\_BANK\_AGENT' under the 'BANK\_PARTNER' system.  
3. If the above conditions are met and a specific PIVC video link is available, the system should construct the final URL by appending the application number and '\_VIDEOPIVC\_JL' to the base video link.  
4. The system should then display the video using the constructed URL.  
5. If any error occurs during the URL construction or video display, the system should show an appropriate error message.  
  
Definition of Done:  
- The video link is correctly retrieved and displayed based on the application number and eligibility criteria.  
- Appropriate error messages are shown if any issues occur during the process.  
- 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 include direct CRUD operations on the database tables.

# Download KYC Details

Type: AGENTS

Detailed description: As a user, I want to download the KYC (Know Your Customer) details for a specific policyholder, so that I can validate the customer's identity and other relevant information.  
  
Acceptance criteria:  
1. When the "DOWNLOAD\_KYC" button is pressed, the system should retrieve the application number from the insured person's verification or sign card number.  
2. The system should format the policyholder's date of birth in 'DD-MM-YYYY' format.  
3. The system should fetch the contract ID based on the policy reference.  
4. The system should retrieve the policyholder's PAN number, date of birth, sex, first name, middle name, and surname from the database.  
5. If any of the above details are not found, they should be set to null.  
6. The system should construct a JSON string with the retrieved details.  
7. The system should send an HTTP POST request to a predefined URL with the JSON string.  
8. The system should log the response from the HTTP request.  
9. The system should display a message with the KYC response.  
  
Definition of Done:  
- The "DOWNLOAD\_KYC" button should trigger the retrieval and validation process as described.  
- The system should handle exceptions gracefully and set missing details to null.  
- The constructed JSON string should be correctly formatted and sent to the predefined URL.  
- The response from the HTTP request should be logged and displayed to the user.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Retrieve contract ID:  
 ```sql  
 SELECT azbj\_pk0\_acc.get\_contract\_id(:control.cn\_policy\_ref) INTO v\_contract\_id FROM dual;  
 ```  
  
- Retrieve policyholder details:  
 ```sql  
 SELECT TAX\_ID, DATE\_OF\_BIRTH, SEX, FIRST\_NAME, v\_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;  
 ```

# Aadhaar Number Validation for Policyholder

Type: AGENTS

Detailed description: As a user, I want to validate the Aadhaar number of the policyholder to ensure that the provided Aadhaar details are accurate and verified.  
  
Acceptance criteria:  
1. The system should enable the Aadhaar input field if it is currently disabled.  
2. The system should extract the first name, last name, and date of birth from the policyholder's details.  
3. If the e-KYC flag is set to 'E-KYC', the system should:  
 - Set the Aadhaar number from a predefined variable.  
 - Mark the e-KYC flag as 'Y'.  
 - Validate the Aadhaar number using an external validation service.  
 - Update the age proof ID if it does not match the Aadhaar number.  
 - Update the AML (Anti-Money Laundering) records with the Aadhaar number if the age proof type is 'AC' or 'ACS'.  
4. If the e-KYC flag is not set to 'E-KYC', the system should:  
 - Validate the Aadhaar number using an external validation service.  
 - Update the Aadhaar flag and new Aadhaar number based on the validation results.  
 - Log the validation process for debugging purposes.  
5. If the Aadhaar number is invalid or not verified, the system should:  
 - Navigate to the validation list block.  
 - Display an error message indicating that a valid Aadhaar number is required.  
6. The system should disable the Aadhaar input field after validation if it was enabled during the process.  
  
Definition of Done:  
- The Aadhaar number validation process is implemented and tested.  
- The system correctly enables and disables the Aadhaar input field as needed.  
- The system accurately extracts and processes the policyholder's details.  
- The e-KYC and non-e-KYC validation paths are correctly implemented.  
- The system updates the age proof ID and AML records as specified.  
- Error handling and logging are implemented for invalid Aadhaar numbers.  
- 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 that can be directly executed in the database without modification.

# View IIB Matched Details

Type: AGENTS

Detailed description: As a user, I want to view detailed information about IIB matched records when I press the "IIB Matched Details" button, so that I can review and analyze the matching details for a specific transaction.  
  
Acceptance criteria:  
1. When the "IIB Matched Details" button is pressed, the system should:  
 - Allow insert and update operations on the IIB details block.  
 - Check if the IIB number in the AGENTS block matches the pattern 'MATCH%FOUND'.  
 - If a match is found, navigate to the IIB details block, clear any existing data, and load the first record.  
 - Retrieve and display records from the `customer.azbj\_iib\_nb\_output` table where the `transaction\_id` matches the current transaction ID, ordered by the `inserted\_date`.  
 - Populate the IIB details block with the retrieved data, including fields such as `INPUT\_PROPOSAL\_POLICY\_NO`, `TRANSACTION\_ID`, `QUESTDBNO`, `INPUT\_MATCHING\_PARAMETER`, and others.  
 - For each record, fetch and display descriptions for `POLICY\_STATUS`, `COMPANY\_NUMBER`, and `CAUSE\_OF\_DEATH` from the `CUSTOMER.AZBJ\_SYSTEM\_CONSTANTS` table based on their respective codes.  
 - Navigate to the next record and repeat the process until all records are processed.  
 - Navigate back to the first record after processing all records.  
 - Disable insert and update operations on the IIB details block.  
 - Display the IIB details view.  
 - Navigate to the "Back" button in the control block.  
  
Definition of Done:  
- The "IIB Matched Details" button functionality is implemented and tested.  
- The system correctly retrieves and displays IIB matched records based on the specified criteria.  
- Descriptions for `POLICY\_STATUS`, `COMPANY\_NUMBER`, and `CAUSE\_OF\_DEATH` are correctly fetched and displayed.  
- The IIB details view is displayed, and the user can navigate back using the "Back" button.  
- All acceptance criteria are met, and the feature is reviewed and approved.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Retrieve IIB matched records:  
 ```sql  
 SELECT   
 FROM customer.azbj\_iib\_nb\_output   
 WHERE transaction\_id = :current\_transaction\_id   
 ORDER BY inserted\_date;  
 ```  
  
- Fetch descriptions for `POLICY\_STATUS`, `COMPANY\_NUMBER`, and `CAUSE\_OF\_DEATH`:  
 ```sql  
 SELECT sys\_desc   
 INTO :policy\_status\_desc   
 FROM CUSTOMER.AZBJ\_SYSTEM\_CONSTANTS   
 WHERE sys\_type = 'POLICY STATUS'   
 AND sys\_code = :quest\_policy\_status;  
  
 SELECT sys\_desc   
 INTO :company\_number\_desc   
 FROM CUSTOMER.AZBJ\_SYSTEM\_CONSTANTS   
 WHERE sys\_type = 'COMPANY\_CODE'   
 AND sys\_code = :quest\_company\_number;  
  
 SELECT sys\_desc   
 INTO :cause\_of\_death\_desc   
 FROM CUSTOMER.AZBJ\_SYSTEM\_CONSTANTS   
 WHERE sys\_type = 'CAUSE\_OF\_DEATH'   
 AND sys\_code = :quest\_cause\_of\_death;  
 ```

# Exit Button Functionality for Income Benefit Rate

Type: AGENTS

Title: Exit Button Functionality for Income Benefit Rate  
  
Acceptance Criteria:  
1. When the "Exit" button is pressed, the "Income Benefit Rate" view should be hidden.  
2. The "Income Benefit Rate" window should be hidden.  
3. The focus should be moved to the "Agents AWG Per Rates List" item.  
  
Definition of Done:  
- The "Exit" button successfully hides the "Income Benefit Rate" view and window.  
- The focus is correctly moved to the "Agents AWG Per Rates List" item.  
- 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 XML content does not include any database queries.

# Generate PIVC Link for Insured Person

Type: AGENTS

Detailed description: As a user, I want to generate a PIVC (Personal Identification Video Conference) link for an insured person so that I can view their video conference details.  
  
Acceptance criteria:  
1. When the "Generate PIVC Link" button is pressed, the system should:  
 - Retrieve the application number using either the verification number or the sign card number of the insured person.  
 - Append '\_VIDEOPIVC' to the application number to form a system type identifier.  
 - Generate a short URL for the PIVC video using the system type identifier and a predefined constant ('BALICVIDEO\_PIVC').  
 - Display the generated PIVC URL in a new window.  
  
2. The new window should be displayed with the PIVC URL.  
  
Definition of Done:  
- The "Generate PIVC Link" button is functional and performs the described actions.  
- The PIVC URL is correctly generated and displayed in a new window.  
- The new window is properly formatted and shows the PIVC URL without any errors.  
- Exception handling is in place to manage any errors during the URL generation process.  
  
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.

# Validate Aadhaar for Insured Person

Type: AGENTS

User Story: Validate Aadhaar for Insured Person  
  
Detailed description:   
As a user, I want to validate the Aadhaar number for an insured person to ensure that the provided Aadhaar details are correct and verified.  
  
Acceptance criteria:  
1. When the "Verify IP Aadhaar" button is pressed, the system should enable the Aadhaar input field if it is currently disabled.  
2. The system should extract the first and last names from the insured person's full name.  
3. If the e-KYC flag is set to 'E-KYC', the system should:  
 - Set the Aadhaar number from a predefined variable.  
 - Mark the e-KYC flag as 'Y'.  
 - Validate the Aadhaar number using an external validation service.  
 - Update the age proof ID if it matches specific conditions.  
 - If the policy holder is the same as the insured person, update the AML (Anti-Money Laundering) records with the Aadhaar number.  
4. If the e-KYC flag is not set to 'E-KYC', the system should:  
 - Validate the provided Aadhaar number and other details.  
 - Update the Aadhaar details in the system.  
 - Log the validation process for debugging purposes.  
5. If the Aadhaar number is invalid or not verified, the system should:  
 - Navigate to the validation list block.  
 - Display an error message indicating that a valid Aadhaar number is required.  
6. If the Aadhaar number is valid, the system should:  
 - Update the age proof ID if it matches specific conditions.  
 - If the policy holder is the same as the insured person, update the AML records with the Aadhaar number.  
7. The system should disable the Aadhaar input field after validation.  
  
Definition of Done:  
- The Aadhaar validation process is triggered by pressing the "Verify IP Aadhaar" button.  
- The system correctly enables and disables the Aadhaar input field as needed.  
- The system extracts and processes the insured person's name correctly.  
- The e-KYC validation process is executed correctly, and the Aadhaar number is validated using the external service.  
- The system updates the age proof ID and AML records as per the specified conditions.  
- Error handling is implemented to navigate to the validation list block and display appropriate error messages.  
- The validation process is logged for debugging purposes.  
- The system is tested and verified to ensure all acceptance criteria are met.

# Link PAN and Aadhar Documents

Type: AGENTS

Title: Link PAN and Aadhar Documents  
  
Acceptance Criteria:  
1. The system should prompt the user to select a document type if it is not already selected.  
2. The system should generate a URL for the document using the provided policy reference and document type.  
3. If the URL is successfully generated, the system should display the document in a web browser.  
4. If there is an error in generating the URL, the system should display a warning message indicating that there is an issue with the URL.  
  
Definition of Done:  
- The user is able to link PAN and Aadhar documents.  
- The system prompts the user to select a document type if it is not selected.  
- The system generates and displays the document URL correctly.  
- Appropriate error messages are shown if there are issues with the URL generation.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- The system should fetch document types from the database using the following query:  
 ```sql  
 SELECT DOC\_TYPE, DOC\_DESC  
 FROM azbj\_dms\_doc\_type\_master  
 ORDER BY SR\_NO;  
 ```

# Fetch and Display Rate Details for Application Number

Type: AGENTS

Title: Fetch and Display Rate Details for Application Number  
  
Acceptance Criteria:  
1. When the button is pressed, the system should fetch the following rate details from the `AZBJ\_ANNUITY\_PROD\_DET` table:  
 - `online\_rate`  
 - `staff\_rate`  
 - `hpmu\_rate`  
 - `income\_factor`  
 - `auto\_pay\_rate`  
 - `loyalty\_factor`  
 - `wmm\_rate`  
 - `awg\_total\_perc`  
2. The application number should be determined based on the `ip\_verf\_no` or `ip\_sign\_card\_no` of the insured person.  
3. The fetched rates should be displayed in the respective fields on the screen.  
4. If the rates cannot be fetched (e.g., due to an error), the fields should be set to `NULL`.  
5. An appropriate warning message should be displayed if an error occurs during the fetching process.  
  
Definition of Done:  
- The button functionality is implemented and tested.  
- The rate details are correctly fetched and displayed based on the application number.  
- Error handling is in place, and appropriate messages are displayed.  
- The functionality is verified to work without any Oracle Forms-specific terminology or dependencies.  
  
DB queries for Table reference CRUD operations only(With Usage):  
```sql  
SELECT online\_rate, staff\_rate, hpmu\_rate, income\_factor, auto\_pay\_rate, loyalty\_factor, wmm\_rate, awg\_total\_perc  
INTO V\_online\_rate, V\_staff\_rate, V\_hpmu\_rate, V\_income\_factor, V\_auto\_pay\_rate, V\_loyality\_factor, V\_wmm\_rate, V\_awg\_total\_perc  
FROM AZBJ\_ANNUITY\_PROD\_DET  
WHERE appln\_no = TO\_CHAR(NVL(:insured\_person.ip\_verf\_no, :insured\_person.ip\_sign\_card\_no)) AND top\_indicator = 'Y';  
```

# Populate Solution Button Functionality

Type: AGENTS

Detailed description: As a user, I want to populate the solution details based on various parameters when I press the "Populate Solution" button, so that the relevant data is fetched and displayed accurately.  
  
Acceptance criteria:  
1. When the "Populate Solution" button is pressed, the system should navigate to the 'LOADINGS' section and fetch the first record.  
2. The system should check if a parameter list named 'Param1' exists. If it does, it should be destroyed and a new parameter list should be created.  
3. The system should add the following parameters to the parameter list:  
 - `inception\_date` from `coverhead.ch\_inception\_date`  
 - `POLICY\_REF` from `control.cn\_policy\_ref`  
 - `APPLICATION\_NO` from `insured\_person.ip\_sign\_card\_no`  
 - `solution\_config\_id` with a value of 'NULL'  
 - `solution\_name` from `coverhead.SOLUTION\_NAME`  
 - `amount\_invested` from `coverhead.AMOUNT\_INVESTED`  
 - `contract\_id` from `control.cn\_contract\_id`  
 - `solution\_id` from `coverhead.SOLUTION\_ID`  
 - `med\_loading\_perc` from `loadings.cv\_ml\_perc`  
 - `oc\_loading\_perc` from `loadings.cv\_oc\_perc`  
 - `sr\_loading\_perc` from `loadings.cv\_sr\_perc`  
 - `occ\_class` from `loadings.cv\_oc\_class`  
 - `nri\_loading\_perc` from `loadings.cv\_nri\_perc`  
 - `occ\_class\_type` from `loadings.cv\_oc\_calc\_type`  
 - `med\_term` from `loadings.cv\_ml\_term`  
 - `discount\_type` from `COVERHEAD.ch\_disc`  
 - `agent\_code` from `AGENTS.AG\_AGENT\_CODE`  
 - `calling\_from` with a value of 'BBU'  
4. The system should call a form named 'AZBJ\_SOLN\_FORM\_CALL' with the created parameter list.  
5. After the form call, the system should navigate to the 'COVERS' section and fetch the first record.  
  
Definition of Done:  
- The "Populate Solution" button should trigger the described sequence of actions.  
- All parameters should be correctly added to the parameter list.  
- The form 'AZBJ\_SOLN\_FORM\_CALL' should be called with the correct parameters.  
- The system should navigate to the 'COVERS' section and fetch the first record without any errors.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Not applicable as the provided logic involves form navigation and parameter handling, which are specific to Oracle Forms and cannot be executed directly in the database without modification.

# Video Link Retrieval and Display

Type: AGENTS

Detailed description: As a user, I want to be able to play a video related to an application number, so that I can view the relevant video content based on specific conditions and eligibility criteria.  
  
Acceptance criteria:  
1. The system should retrieve the application number from either the verification number or the sign card number of the insured person.  
2. The system should calculate the policy holder's age using their date of birth and the current date.  
3. If the application number is not null:  
 - If the product ID is either 297 or 345, the system should attempt to retrieve the video URL from the `azbj\_video\_dtls` table where the module flag is 'BLACKCAT' and the application number matches. If no URL is found, it should set the video link to null.  
 - If the product ID is not 297 or 345, the system should check if the application is eligible for a video by querying the `azbj\_proposal\_appln\_det\_ext` table where the `med\_pivc\_flag` is 'VIDEO' and the application number matches. If eligible, it should proceed to further checks.  
 - The system should check additional conditions such as product ID being 321, the existence of a bank partner, and other product-specific criteria.  
 - If the conditions are met, the system should check if a video already exists in the `balic.pdbdocument` and `balic.ddt\_1` tables. If no video exists, it should retrieve the video link from the `azbj\_system\_constants` table where the system type is 'PIVC\_LINK' and the system code is 'PIVC\_VIDEO'.  
 - If a video link is found, it should append the application number to the link.  
 - If a video already exists, it should construct the final URL using the `v\_spg\_pivc\_link` and the application number.  
4. If the product ID is 321 and certain conditions are not met, the system should attempt to retrieve the URL path from the `azbj\_pivc\_dtls` table using the agent's business identifier number.  
5. If the URL path does not start with 'HTTP', it should construct the final URL using the `v\_spg\_pivc\_link` and the application number.  
6. If specific flags related to video eligibility are enabled, the system should retrieve the video calling URL from the `azbj\_system\_constants` table where the system type is 'PIVC\_CALLING' and the system code is 'PIVC\_VIDEO\_CALLING'.  
7. The system should display the final URL using a web document viewer.  
8. If any errors occur during the process, the system should display an appropriate error message.  
  
Definition of Done:  
- The video link retrieval and display functionality should be implemented and tested.  
- The system should handle all specified conditions and eligibility checks.  
- Appropriate error handling and messaging should be in place.  
- The functionality should be independent of any specific technology or Oracle Forms terminology.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Retrieve video URL from `azbj\_video\_dtls`:  
 ```sql  
 SELECT view\_video\_url  
 INTO v\_video\_link  
 FROM azbj\_video\_dtls a  
 WHERE a.module\_flag = 'BLACKCAT'  
 AND a.application\_no = v\_application\_no  
 AND a.insert\_time\_stamp = (SELECT MAX(b.insert\_time\_stamp)  
 FROM azbj\_video\_dtls b  
 WHERE b.module\_flag = 'BLACKCAT'  
 AND b.application\_no = a.application\_no);  
 ```  
  
- Check video eligibility from `azbj\_proposal\_appln\_det\_ext`:  
 ```sql  
 SELECT CASE WHEN COUNT(1) > 0 THEN 'Y' ELSE 'N' END  
 INTO v\_pivc\_vid\_eligible  
 FROM azbj\_proposal\_appln\_det\_ext  
 WHERE UPPER(med\_pivc\_flag) = 'VIDEO'  
 AND appln\_no = v\_application\_no;  
 ```  
  
- Check existing video count from `balic.pdbdocument` and `balic.ddt\_1`:  
 ```sql  
 SELECT COUNT(1)  
 INTO vvideoplvccnt  
 FROM balic.pdbdocument a, balic.ddt\_1  
 WHERE folddocindex = documentindex  
 AND UPPER(field\_7) = v\_application\_no  
 AND UPPER(name) LIKE v\_application\_no || '\_VIDEOPIVC';  
 ```  
  
- Retrieve video link from `azbj\_system\_constants`:  
 ```sql  
 SELECT sys\_desc  
 INTO v\_video\_link  
 FROM azbj\_system\_constants  
 WHERE sys\_type = 'PIVC\_LINK'  
 AND sys\_code = 'PIVC\_VIDEO';  
 ```  
  
- Retrieve URL path from `azbj\_pivc\_dtls`:  
 ```sql  
 SELECT NVL(url\_path, 'X')  
 INTO v\_final\_url  
 FROM azbj\_pivc\_dtls  
 WHERE bi\_no = :agents.bi\_no AND ROWNUM = 1;  
 ```  
  
- Retrieve video calling URL from `azbj\_system\_constants`:  
 ```sql  
 SELECT SYS\_DESC  
 INTO v\_final\_url  
 FROM azbj\_system\_constants  
 WHERE sys\_type = 'PIVC\_CALLING'  
 AND sys\_code = 'PIVC\_VIDEO\_CALLING'  
 AND end\_date IS NULL;  
 ```

# Validate PAN Card Details

Type: AGENTS

User Story: Validate PAN Card Details  
  
Detailed description: As a user, I want to validate the PAN card details entered in the system to ensure that the PAN card number, name, and other related details are accurate and match the required format and criteria.  
  
Acceptance criteria:  
1. The system should prompt an error message if the PAN card number is not entered.  
2. The system should validate that the PAN card number is exactly 10 characters long.  
3. The system should check if the PAN card number follows the correct format (e.g., 5 letters followed by 4 digits and 1 letter).  
4. The system should verify that the PAN card number matches the proposal type and display an appropriate message if there is a mismatch.  
5. The system should update the PAN card details in the relevant sections if the PAN card number is valid.  
6. The system should handle exceptions and display error messages for invalid PAN card numbers or other validation failures.  
7. The system should log the PAN card validation process and results for auditing purposes.  
  
Definition of Done:  
- The PAN card validation logic is implemented and integrated into the system.  
- The system displays appropriate error messages for invalid PAN card numbers.  
- The system updates the PAN card details in the relevant sections if the PAN card number is valid.  
- The system logs the PAN card validation process and results.  
- The functionality is tested and verified to ensure it meets the acceptance criteria.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- The system should execute the following query to validate the PAN card number format:  
 ```sql  
 SELECT 'Y'  
 INTO V\_IP\_AGE\_PROOF\_ID  
 FROM DUAL  
 WHERE REGEXP\_LIKE(:agents.pan\_card, '^[A-Z]{3}[C,P,H,F,A,T,B,L,J,G][A-Z][0-9]{4}[A-Z]');  
 ```  
- The system should execute the following query to check the proposal type:  
 ```sql  
 SELECT CASE WHEN COUNT(1) > 0 THEN 'Y' ELSE 'N' END  
 INTO v\_valid\_pancard  
 FROM AZBJ\_PROPOSAL\_TYPE  
 WHERE INTERNAL\_VALUE = :policy\_holder.PROP\_I\_VAL  
 AND 1 = CASE  
 WHEN :policy\_holder.PROP\_I\_VAL IN ('I', 'N', 'F', 'P', 'JL', 'Q', 'R', 'NPS', 'BDP', 'EIA', 'NGST', 'MWP') AND UPPER(v\_digit) IN ('P') THEN 1  
 WHEN :policy\_holder.PROP\_I\_VAL IN ('K', 'E', 'GS') AND UPPER(v\_digit) IN ('C', 'F', 'G', 'L', 'A', 'B', 'T') THEN 1  
 WHEN :policy\_holder.PROP\_I\_VAL IN ('H') AND UPPER(v\_digit) IN ('H') THEN 1  
 WHEN :policy\_holder.PROP\_I\_VAL IN ('PS') AND UPPER(v\_digit) IN ('F') THEN 1  
 WHEN :policy\_holder.PROP\_I\_VAL IN ('TF') AND UPPER(v\_digit) IN ('T', 'A') THEN 1  
 ELSE 0  
 END;  
 ```

# Add New Medical Requirements Form

Type: MED\_FUR\_REQ

Title: Add New Medical Requirements Form  
  
Acceptance Criteria:  
1. The form should have a button labeled "OK" to confirm the addition of the new requirement.  
2. The form should have a button labeled "Cancel" to discard the new requirement and close the form.  
3. The form should include a text field for entering the test number.  
4. The form should include a text field for displaying the test description, which should be disabled for editing.  
5. The form should include a field for selecting the request type.  
  
Definition of Done:  
1. The form is accessible and displays correctly within the application.  
2. The "OK" button successfully saves the new requirement and closes the form.  
3. The "Cancel" button discards any input and closes the form without saving.  
4. The test number field allows input and validates against a predefined list.  
5. The test description field displays relevant information based on the test number entered but is not editable.  
6. The request type field is available for selection and correctly records the chosen type.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include specific database queries.

# Cancel Operation and Return to Main List

Type: MED\_FUR\_REQ

Title: Cancel Operation and Return to Main List  
  
Acceptance Criteria:  
1. When the "Cancel" button is pressed, the current item should be disabled.  
2. The system should navigate to the main block of records.  
3. The system should move to the last record in the main block.  
4. The focus should be set to a specific item in the main block.  
  
Definition of Done:  
- The "Cancel" button is functional and performs the specified actions.  
- The user is able to cancel the current operation and return to the main list of records.  
- The system correctly disables the current item, navigates to the main block, moves to the last record, and sets the focus to the specified item.

# Test Number Input and Validation

Type: MED\_FUR\_REQ

Title: Test Number Input and Validation  
  
Acceptance Criteria:  
1. The input field for the test number should be positioned at the start of the form.  
2. The test number should be validated against a predefined list when the user interacts with the field.  
3. The field should support double-click functionality to trigger the list validation.  
4. The form should display the test number in a specific font and color for better readability.  
5. The form should be presented in a dialog window that cannot be resized, minimized, or maximized.  
  
Definition of Done:  
1. The test number input field is correctly positioned and styled as per the requirements.  
2. The validation logic for the test number is implemented and triggers on user interaction.  
3. The form is displayed in a non-resizable, non-minimizable, and non-maximizable dialog window.  
4. The form is tested and verified to ensure all acceptance criteria are met.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include any specific database queries.

# Process Medical and Further Requests on Button Press

Type: MED\_FUR\_REQ

Title: Process Medical and Further Requests on Button Press  
  
Acceptance Criteria:  
1. When the button is pressed, if the request type is 'MED':  
 - The system should navigate to the medical underwriting section.  
 - If a test number is provided, the system should:  
 - Create a new record if no test number exists in the medical underwriting section.  
 - Populate the test number, description, and date called fields.  
 - Set the validation control to 'N'.  
 - Determine the 'Raised By' field based on the existence of the contract ID in the policy base extension table.  
 - Set the request called and date called fields.  
 - If the policy holder is equal to the insured, set the insured person type to 1.  
 - Disable the test number field and navigate to the results received field.  
  
2. If the request type is not 'MED':  
 - The system should navigate to the further requests section.  
 - If a test number is provided, the system should:  
 - Create a new record if no test number exists in the further requests section.  
 - Populate the test number, description, and date called fields.  
 - Set the results received field to 'N'.  
 - Determine the 'Raised By' field based on the existence of the contract ID in the policy base extension table.  
 - Set the request called and date called fields.  
 - If the policy holder is equal to the insured, set the insured person type to 1.  
 - Disable the test number field and navigate to the results received field.  
  
Definition of Done:  
- The button functionality should be implemented and tested.  
- The system should correctly handle the navigation and data population based on the request type.  
- All fields should be populated and validated as per the acceptance criteria.  
- The test number fields should be disabled after processing.  
- The system should navigate to the appropriate fields after processing.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Query to determine 'Raised By' field:  
 ```sql  
 SELECT CASE WHEN COUNT(1) > 0 THEN 'SYS-USER CALLED' ELSE 'USER CALLED' END  
 INTO :med\_uw.RAISED\_BY  
 FROM wip\_azbj\_med\_uw  
 WHERE contract\_id IN (  
 SELECT contract\_id  
 FROM wip\_azbj\_policy\_bases\_ext  
 WHERE sign\_Card\_no = TO\_CHAR(NVL(:insured\_person.ip\_verf\_no, :insured\_person.ip\_sign\_card\_no))  
 )  
 AND testno = :Med\_FUR\_REQ.test\_no;  
 ```  
  
- Query to determine 'Raised By' field for further requests:  
 ```sql  
 SELECT CASE WHEN COUNT(1) > 0 THEN 'SYS-USER CALLED' ELSE 'USER CALLED' END  
 INTO :FURTHER\_REQ.RAISED\_BY  
 FROM wip\_azbj\_med\_uw  
 WHERE contract\_id IN (  
 SELECT contract\_id  
 FROM wip\_azbj\_policy\_bases\_ext  
 WHERE sign\_Card\_no = TO\_CHAR(NVL(:insured\_person.ip\_verf\_no, :insured\_person.ip\_sign\_card\_no))  
 )  
 AND testno = :Med\_FUR\_REQ.test\_no;  
 ```

# Manage Fund Information

Type: SSO\_FUND

Title: Manage Fund Information  
  
Acceptance Criteria:  
1. The system should display a list of funds with the following details:  
 - Fund Number  
 - Fund Name  
 - Percentage Allocation  
 - Total Fund Value (calculated as the sum of percentage allocations)  
2. The system should validate the fund name from a predefined list.  
3. The system should restrict the selection to a maximum of four funds.  
4. The system should display a message if more than four funds are selected.  
5. The system should fetch fund details based on specific criteria:  
 - The fund must be active within a given date range.  
 - The fund must not have a specific ID.  
 - The fund must match the product and cover code criteria.  
  
Definition of Done:  
- The user can view a list of funds with the specified details.  
- The user can add and update fund details.  
- The system validates the fund name from a predefined list.  
- The system restricts the selection to a maximum of four funds and displays an appropriate message if this limit is exceeded.  
- The system fetches and displays fund details based on the specified criteria.  
  
DB queries for Table reference CRUD operations only(With Usage):  
```sql  
SELECT fund\_short\_name, fund\_full\_name   
FROM AZBJ\_TFV\_FUND\_DEFINITION a, azbj\_cover\_funds b   
WHERE a.fund\_short\_name = b.fund\_name  
AND b.product\_id = :control.cn\_product\_id  
AND :susac.sa\_daterecd > NVL(start\_date, '01-apr-2005')  
AND :susac.sa\_daterecd < NVL(end\_date, '01-jan-3000')  
AND a.fund\_id <> 'FVFD000025'  
AND cover\_code = :covers.cv\_cover\_code;  
```

# Validate Apportionment for Cash Plus Pension Fund

Type: SSO\_FUND

Detailed description: As a user, I want to ensure that the apportionment for the Cash Plus Pension Fund does not exceed 20% when certain product IDs are selected, so that the system enforces the correct business rules for fund apportionment.  
  
Acceptance criteria:  
1. When the product ID is one of the following: 31, 32, 33, 34, 49, or 50, and the fund ID is 'NCPPF', the system should validate that the apportionment percentage does not exceed 20%.  
2. If the apportionment percentage exceeds 20% under these conditions, an error message should be displayed: "Apportionment for Cash Plus Pension Fund cannot be more than 20".  
3. The form status should be updated to 'Y' after the validation check.  
  
Definition of Done:  
- The system correctly identifies the specified product IDs and fund ID.  
- The validation logic is implemented to check the apportionment percentage.  
- An appropriate error message is displayed when the apportionment percentage exceeds the allowed limit.  
- The form status is updated as required after the validation.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include any direct database queries.

# User can view and select a fund from the list on double-clicking the 'Fund Name' field

Type: SSO\_FUND

Title: User can view and select a fund from the list on double-clicking the 'Fund Name' field  
  
Acceptance Criteria:  
1. When the user double-clicks on the 'Fund Name' field, the system should check if the global loading status is 'F'.  
2. If the global loading status is 'F', the system should set the form status to 'Y'.  
3. The system should then navigate to the 'covers' block and display the first record.  
4. The system should return to the 'SSO\_FUND' block.  
5. The system should execute a query to fetch the fund short name and fund full name from the database based on the product ID, date range, and cover code.  
6. The system should populate the list of values (LOV) with the fetched data.  
7. The system should display the LOV to the user.  
8. The user should be able to select a fund from the LOV.  
9. The system should navigate to the 'Fundapor' field in the 'SSO\_FUND' block after the user selects a fund.  
  
Definition of Done:  
- The user can double-click on the 'Fund Name' field to open a list of available funds.  
- The list of available funds is populated based on the specified criteria.  
- The user can select a fund from the list.  
- The system navigates to the 'Fundapor' field after the user selects a fund.  
  
DB queries for Table reference CRUD operations only (With Usage):  
```sql  
SELECT fund\_short\_name, fund\_full\_name  
FROM AZBJ\_TFV\_FUND\_DEFINITION a, azbj\_cover\_funds b  
WHERE a.fund\_short\_name = b.fund\_name  
AND b.product\_id = :control.cn\_product\_id  
AND :susac.sa\_daterecd > NVL(start\_date, '01-apr-2005')  
AND :susac.sa\_daterecd < NVL(end\_date, '01-jan-3000')  
AND a.fund\_id <> 'FVFD000025'  
AND cover\_code = :covers.cv\_cover\_code  
ORDER BY fund\_short\_name;  
```

# View and Manage Probable Client Policies

Type: BLK\_PROBABLE\_CP\_PH

Title: View and Manage Probable Client Policies  
  
Acceptance Criteria:  
1. The interface should display a list of probable client policies with the following attributes:  
 - Client ID  
 - Policy Number  
 - Name  
 - Annual Premium  
 - Policy Status  
 - SUC  
 - TASA  
 - Sum Assured  
 - Product ID  
 - Package Code  
 - Rated Up status  
2. Each policy should have a checkbox for selection.  
3. There should be a button to view detailed information about the selected policy.  
4. There should be an exit button to close the interface.  
5. The interface should be visually organized and easy to navigate.  
  
Definition of Done:  
- The interface displays all the required attributes for each probable client policy.  
- Users can select policies using checkboxes.  
- Users can view detailed information about a selected policy by clicking the "View CP" button.  
- Users can exit the interface by clicking the "Exit" button.  
- The interface is tested for usability and functionality, ensuring it meets the acceptance criteria.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- No specific database queries are provided in the XML content.

# Implement Checkbox for IP/PH Selection

Type: BLK\_PROBABLE\_CP\_PH

Title: Implement Checkbox for IP/PH Selection  
  
Acceptance Criteria:  
1. When the checkbox is checked, the system should set a variable `v\_prob\_ph\_sel` to 'Y'.  
2. The checkbox should have a checked value of '1' and an unchecked value of '0'.  
3. The checkbox should be visible and properly aligned with the label "IP/PH Selection".  
4. The checkbox should be located at the specified position on the screen.  
  
Definition of Done:  
1. The checkbox is implemented and visible on the user interface.  
2. The checkbox correctly sets the variable `v\_prob\_ph\_sel` to 'Y' when checked.  
3. The checkbox label "IP/PH Selection" is properly aligned and readable.  
4. The checkbox is functional and its state (checked/unchecked) is correctly captured and stored.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Not applicable as the provided XML content does not include any direct database operations.

# View Policy Details on Button Press

Type: BLK\_PROBABLE\_CP\_PH

Title: View Policy Details on Button Press  
  
Acceptance Criteria:  
1. When the "View CP" button is pressed, the system should navigate to the policy details section.  
2. The system should clear any existing data in the policy details section.  
3. The system should retrieve and display the first record of the policy details.  
4. The system should fetch policy details from the database based on the insured person's ID.  
5. The system should populate the policy details section with the following information:  
 - Partner ID  
 - Policy Reference  
 - Partner Name  
 - Annual Premium  
 - Policy Status  
 - Cover Code  
 - Sum Insured for Whole Cover  
 - Entry Age  
 - Premium Term  
 - Benefit Term  
 - Interest Rate  
 - Frequency Standard Premium  
 - Extra Amount  
 - ML Percentage  
 - OC Percentage  
 - NRI Percentage  
 - SR Percentage  
 - Premium Discount Amount  
6. The system should calculate the premium amount and inception date for the policy.  
7. The system should call a function to calculate the reinsurance details and update the policy details section with the calculated values.  
8. The system should handle any errors during the process and display an appropriate error message.  
  
Definition of Done:  
- The "View CP" button functionality is implemented and tested.  
- The policy details section is correctly populated with the retrieved data.  
- The premium amount and inception date are accurately calculated and displayed.  
- The reinsurance details are correctly calculated and displayed.  
- Error handling is implemented, and appropriate error messages are shown when necessary.  
- The functionality is reviewed and approved by stakeholders.  
  
DB queries for Table reference CRUD operations only(With Usage):  
```sql  
-- Fetch premium amount from WIP\_AZBJ\_POLICY\_BASES\_EXT  
SELECT premium\_amount INTO v\_premium\_amount   
FROM WIP\_AZBJ\_POLICY\_BASES\_EXT   
WHERE contract\_id = cur\_partner.contract\_id;  
  
-- Fetch premium amount from AZBJ\_POLICY\_BASES\_EXT  
SELECT premium\_amount INTO v\_premium\_amount   
FROM AZBJ\_POLICY\_BASES\_EXT   
WHERE contract\_id = cur\_partner.contract\_id  
 AND top\_indicator = 'Y'   
 AND action\_code <> 'D';  
  
-- Fetch inception date from wip\_policy\_contracts  
SELECT inception\_date INTO v\_inception\_date   
FROM wip\_policy\_contracts   
WHERE contract\_id = cur\_partner.contract\_id;  
  
-- Fetch inception date from ocp\_policy\_contracts  
SELECT inception\_date INTO v\_inception\_date   
FROM ocp\_policy\_contracts   
WHERE contract\_id = cur\_partner.contract\_id;  
```

# View and Manage Probable Client Policies (CPs)

Type: BLK\_PROBABLE\_CP

Detailed description: As a user, I want to view and manage probable client policies (CPs) in a structured and user-friendly interface, so that I can easily access and review important policy details.  
  
Acceptance criteria:  
1. The interface should display a list of probable CPs with the following details:  
 - IP CP Selection (checkbox)  
 - Client ID  
 - Policy Number  
 - Name  
 - Annual Premium  
 - Policy Status  
 - SUC  
 - TASA  
 - Sum Assured  
 - IP TASA  
 - IP SUC  
 - RATED UP  
 - Prod ID  
 - Package code  
2. The interface should allow the user to select or deselect CPs using a checkbox.  
3. The interface should include an "Exit" button to close the window.  
4. The interface should be visually organized with labels and fields aligned properly for readability.  
  
Definition of Done:  
- The interface displays all the specified CP details.  
- The checkbox for IP CP Selection is functional.  
- The "Exit" button closes the window.  
- The interface is visually organized and user-friendly.  
- All fields are visible and properly aligned.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- No specific DB queries provided in the XML content.

# IP CP Selection Checkbox Functionality

Type: BLK\_PROBABLE\_CP

Title: IP CP Selection Checkbox Functionality  
  
Acceptance Criteria:  
- When the checkbox for IP CP Selection is checked, the system should set the variable `v\_prob\_ip\_sel` to 'Y'.  
- When the checkbox for IP CP Selection is unchecked, the system should set the variable `v\_prob\_ip\_sel` to 'N'.  
  
Definition of Done:  
- The checkbox for IP CP Selection is visible and functional within the Probable CPs section.  
- The system correctly updates the variable `v\_prob\_ip\_sel` based on the checkbox state.  
- The changes are tested and verified to ensure the correct behavior of the checkbox.  
  
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.

# Exit Button Functionality in Probable CPs Section

Type: BLK\_PROBABLE\_CP

Detailed description: As a user, I want to be able to exit the "Probable CPs" section and return to the "Insured Person" section seamlessly, so that I can navigate the application efficiently.  
  
Acceptance criteria:  
1. When the "Exit" button is pressed, the "Probable CPs" section should be hidden.  
2. The window titled "Probable CPs" should be closed.  
3. The focus should be moved to the "Insured Person" section.  
  
Definition of Done:  
- The "Exit" button is functional and performs the actions as described in the acceptance criteria.  
- The "Probable CPs" section and window are hidden upon pressing the "Exit" button.  
- The focus is successfully moved to the "Insured Person" section.  
- The functionality is tested and verified to ensure smooth navigation.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include any database queries.

# Manage Insured Person Details

Type: INSURED\_PERSON

Detailed description: As a user, I want to manage the details of an insured person, including personal information, contact details, and other relevant data, so that I can ensure all necessary information is captured and maintained accurately.  
  
Acceptance criteria:  
1. The system should validate that the insured person's ID is not null when certain conditions are met.  
2. If the insured person's ID is null, an error message should be displayed, and the user should not be able to proceed.  
3. The system should copy the insured person's details to the policy holder's details when a specific condition is met.  
4. The system should disable certain fields in the policy holder's section when the insured person's details are copied.  
5. The system should enable certain fields in the policy holder's section when the insured person's details are not copied.  
  
Definition of Done:  
1. The insured person's details can be entered and saved.  
2. The system performs the necessary validations and displays appropriate error messages.  
3. The insured person's details are correctly copied to the policy holder's section when required.  
4. The relevant fields in the policy holder's section are enabled or disabled based on the conditions.  
5. The user interface is intuitive and user-friendly, allowing users to easily manage the insured person's details.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include any direct database queries.

# Display 'Not Available' for Empty Contact Telephone Number

Type: INSURED\_PERSON

Title: Display 'Not Available' for Empty Contact Telephone Number  
  
Acceptance Criteria:  
1. When the user double-clicks on the contact telephone number field for an insured person, the system should check if the field is empty.  
2. If the contact telephone number field is empty, the system should automatically populate it with the text 'Not Available'.  
3. If the contact telephone number field is not empty, no changes should be made.  
  
Definition of Done:  
1. The functionality to check and update the contact telephone number field is implemented.  
2. The feature is tested to ensure that it correctly updates the field to 'Not Available' when it is empty.  
3. The feature is reviewed and approved by the stakeholders.  
4. 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 any database-specific operations.

# Validate Changed Weight Field in Insured Person Section

Type: INSURED\_PERSON

Title: Validate Changed Weight Field in Insured Person Section  
  
Acceptance Criteria:  
1. If the "Changed Weight" field is left empty, it should default to 0.  
2. If the form is not in loading mode, the form status should be set to 'Y'.  
3. The "Changed Weight" value should not be greater than or equal to the "Weight" value.  
4. The validation should only apply if the policy holder's property value is not 'H' or 'E' and the product ID is not 169.  
5. If the validation fails, an error message should be displayed: "Changed Weight Cannot be greater or equal to Weight".  
  
Definition of Done:  
- The "Changed Weight" field defaults to 0 when left empty.  
- The form status is updated correctly when the form is not in loading mode.  
- The validation logic ensures that the "Changed Weight" is less than the "Weight" value under specified conditions.  
- An appropriate error message is displayed when the validation fails.  
- All changes are tested and verified to work as expected.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include any direct database queries.

# Occupation Status Field for Insured Person

Type: INSURED\_PERSON

Title: Occupation Status Field for Insured Person  
  
Acceptance Criteria:  
1. The occupation status field should be displayed on the "Insured Person" tab.  
2. The field should be a dropdown list containing predefined occupation statuses.  
3. The field should be read-only, meaning users cannot add or modify the list items.  
4. The field should be positioned correctly on the form as per the design specifications.  
5. The field should be labeled "Occupation" and the label should be bold and aligned properly.  
6. The field should not allow any input or updates directly by the user.  
  
Definition of Done:  
1. The occupation status field is visible on the "Insured Person" tab.  
2. The dropdown list contains the predefined occupation statuses.  
3. The field is read-only and cannot be modified by the user.  
4. The field and its label are correctly positioned and styled as per the design specifications.  
5. The functionality has been tested and verified to meet the acceptance criteria.  
  
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.

# Determine and Display Age Proof Type

Type: INSURED\_PERSON

Title: Determine and Display Age Proof Type  
  
Acceptance Criteria:  
1. When the user interacts with the age proof field, the system should:  
 - Check the type of age proof provided.  
 - Display "Non standard Age Proof" if the age proof type is not standard.  
 - Display "Standard Age Proof" if the age proof type is standard.  
 - Handle any exceptions by displaying an appropriate error message.  
  
2. When the user double-clicks on the age proof field, the system should:  
 - Retrieve and display a unique identifier for the age proof type.  
 - Handle cases where no data is found by displaying "NA".  
 - Handle any other exceptions by displaying "NA".  
  
3. When the age proof field is validated, the system should:  
 - Set the age proof field to null if the age proof type is not provided.  
 - Enable or disable the PAN issuance date fields based on the age proof type.  
 - Update the form status to indicate changes.  
  
4. When the age proof field gains focus, the system should:  
 - Check the type of age proof provided.  
 - Display "Non standard Age Proof" if the age proof type is not standard.  
 - Display "Standard Age Proof" if the age proof type is standard.  
 - Handle any exceptions by displaying an appropriate error message.  
  
Definition of Done:  
- The system correctly determines and displays the type of age proof.  
- The system handles specific conditions related to the age proof type and PAN card details.  
- All exceptions are handled gracefully with appropriate error messages.  
- The form status is updated correctly based on user interactions.  
- The functionality is tested and verified to work as expected.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Retrieve unique identifier for age proof type:  
 ```sql  
 SELECT AGE\_UID  
 INTO :insured\_person.IP\_AGE\_PRF\_MSG  
 FROM azbj\_Age\_proof\_UID  
 WHERE age\_proof = :insured\_person.ip\_age\_proof;  
 ```  
  
- Handle no data found and other exceptions:  
 ```sql  
 EXCEPTION  
 WHEN no\_data\_found THEN  
 :insured\_person.IP\_AGE\_PRF\_MSG := 'NA';  
 WHEN others THEN  
 :insured\_person.IP\_AGE\_PRF\_MSG := 'NA';  
 ```

# Automatic Navigation from MICR Code to Account Number

Type: INSURED\_PERSON

Title: Automatic Navigation from MICR Code to Account Number  
  
Acceptance Criteria:  
1. When the user enters the MICR code and presses the designated key to move to the next item, the system should automatically navigate to the account number field of the insured person.  
2. The MICR code field should accept a maximum of 9 numeric characters.  
3. The MICR code field should be visually distinct with a specific background color and font style.  
  
Definition of Done:  
1. The MICR code field is implemented and accepts up to 9 numeric characters.  
2. The system automatically navigates to the account number field after the MICR code is entered.  
3. The MICR code field is styled with the specified background color and font attributes.  
4. The functionality is tested and verified to ensure smooth navigation and data entry.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include any specific database queries.

# Navigation based on Product ID for Sign Card No.

Type: INSURED\_PERSON

Title: Navigation based on Product ID for Sign Card No.  
  
Acceptance Criteria:  
- If the product ID is 11 or the product definition is 'CHILDGAIN', the system should navigate to the "Minor Question Flag" field for the insured person.  
- If the product ID is not 11 and the product definition is not 'CHILDGAIN', the system should navigate to the "Method of Payment" field for the policy holder.  
  
Definition of Done:  
- The navigation logic is implemented and tested.  
- The system correctly navigates to the specified fields based on the product ID and product definition.  
- 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 any direct database queries.

# Validate Age Proof ID for Insured Person

Type: INSURED\_PERSON

Detailed description: As a user, I want the system to validate the age proof ID for the insured person based on specific conditions, so that the correct age proof details are captured and verified.  
  
Acceptance criteria:  
1. If the age proof type is 'AC' or 'ACS' and the length of the age proof ID is 12 characters, the system should:  
 - Extract the first name, middle name, and last name from the insured person's name.  
 - Convert the insured person's date of birth to the format 'DD/MON/RRRR'.  
 - Create a record with the insured person's details, including the age proof ID, and validate the Aadhaar data.  
 - Update the age proof ID with the validated Aadhaar data.  
 - Log the validation process and any errors encountered.  
  
2. If the age proof type is 'Pan Card' and the age proof ID is not null, the system should:  
 - Assign the age proof ID to the agent's PAN card.  
 - Set the agent's PAN card verification status to 'Y'.  
 - Navigate to the agent's validation item and execute the validation trigger.  
 - Navigate to the insured person's PAN issuance date item.  
  
Definition of Done:  
- The system correctly validates and updates the age proof ID based on the specified conditions.  
- The validation process is logged, and any errors are captured and displayed to the user.  
- The user can navigate through the form without encountering any issues related to the age proof ID validation.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Not applicable as the provided XML content does not include direct database queries for CRUD operations.

# Checkbox for Health Conditions and Auto-Navigation

Type: INSURED\_PERSON

Title: Checkbox for Health Conditions and Auto-Navigation  
  
Acceptance Criteria:  
1. The checkbox should have two states: checked (Y) and unchecked (N).  
2. The label for the checkbox should read "Hearing, Vision & Speech, Handicap".  
3. When the checkbox is checked, the system should automatically move the cursor to the next field, which is the "Appointee" field for the insured person.  
  
Definition of Done:  
1. The checkbox is displayed on the form with the correct label.  
2. The checkbox can be checked and unchecked.  
3. Upon checking or unchecking the checkbox, the cursor automatically moves to the "Appointee" field for the insured person.  
4. The form is visually consistent with the rest of the application, maintaining the same color scheme and font styles.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include any specific database queries.

# Input Field for Panel Doctor ID

Type: INSURED\_PERSON

Title: Input Field for Panel Doctor ID  
  
Acceptance Criteria:  
- The input field for the panel doctor ID should be positioned at coordinates (343, 281) on the user interface.  
- The input field should have a width of 118 units and a height of 20 units.  
- The background color of the input field should be white, and the text color should be black.  
- The font used for the input field should be "MS Sans Serif" with a font size of 8 and a font weight of "Demilight".  
- The input field should accept only numerical values.  
- The input field should be left-justified.  
- The input field should be navigable using the keyboard, allowing the user to move to the next item using the "Tab" key.  
  
Definition of Done:  
- The input field for the panel doctor ID is implemented and meets all the specified acceptance criteria.  
- The input field is tested to ensure it accepts only numerical values and is navigable using the keyboard.  
- The input field is visually verified to ensure it matches the specified coordinates, dimensions, and styling.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include any specific database queries.

# Validate MIB Code for Insured Person

Type: INSURED\_PERSON

Title: Validate MIB Code for Insured Person  
  
Acceptance Criteria:  
1. When a user enters an MIB code, the system should check the length of the code.  
2. If the length of the MIB code is 4 or more characters, the 4th character must be one of the following: G, H, J, K, M, T, W, Q, S (case insensitive). If not, an error message "Invalid MIB Code Entered, Please check!!!" should be displayed.  
3. If the length of the MIB code is 5 or more characters, the 5th character must be one of the following: X, R, Z, Y, E (case insensitive). If not, an error message "Invalid MIB Code Entered, Please check!!!" should be displayed.  
4. If the length of the MIB code is 6 or more characters, the 6th character must be one of the following: N, B, C, D (case insensitive). If not, an error message "Invalid MIB Code Entered, Please check!!!" should be displayed.  
5. After entering a valid MIB code, the system should automatically navigate to the next item in the form.  
  
Definition of Done:  
- The MIB code validation logic is implemented and tested.  
- Error messages are displayed correctly for invalid MIB codes.  
- The system navigates to the next item after a valid MIB code is entered.  
- All acceptance criteria are met and verified through testing.

# Capture and Validate Annual Income for Insured Person

Type: INSURED\_PERSON

User Story: Capture and Validate Annual Income for Insured Person  
  
Detailed Description:  
As a user, I want to capture and validate the annual income of the insured person to ensure that the data is accurate and meets the necessary criteria for further processing.  
  
Acceptance Criteria:  
1. The system should allow the user to input the annual income of the insured person.  
2. The system should validate the input to ensure it is a numeric value and within the acceptable range.  
3. If the annual income is changed, the system should re-evaluate the form status and set it to 'Y' if the form is not in loading mode.  
4. The system should check if the annual income change requires re-evaluation by the Business Unit (BBU) and disable the save option if necessary.  
5. The system should ensure that all required questions for the insured person and policy members are populated before proceeding.  
6. The system should handle specific conditions such as product definitions, annuity flags, and previous policy checks to ensure comprehensive validation.  
  
Definition of Done:  
- The user can successfully input and save the annual income of the insured person.  
- The system performs all necessary validations and updates the form status accordingly.  
- All required questions for the insured person and policy members are populated and validated.  
- The system handles specific conditions and exceptions as per the business rules.  
  
DB Queries for Table Reference CRUD Operations:  
- The system should execute the following query to check for previous policy details:  
 ```sql  
 SELECT COUNT () INTO v\_cnt2  
 FROM balic\_history\_details  
 WHERE appl\_no = TO\_CHAR (NVL (:insured\_person.ip\_sign\_card\_no, :insured\_person.ip\_verf\_no))  
 AND UPPER(tele\_ver\_status) = 'COMPLETE'  
 AND updated\_flag = 'Y';  
 ```  
- The system should execute the following query to check for incomplete inwarded applications:  
 ```sql  
 SELECT COUNT()  
 INTO v\_incomplete\_flg  
 FROM azbj\_phub\_add\_req\_tracker a, azbj\_phub\_req\_tracker b  
 WHERE a.APPLICATION\_NO = b.APPLICATION\_NO  
 AND a.REQ\_TYPE = b.REQ\_TYPE  
 AND a.application\_no = TO\_CHAR(NVL(:INSURED\_PERSON.IP\_VERF\_NO, :INSURED\_PERSON.IP\_SIGN\_CARD\_NO))  
 AND b.RECD\_IN\_OPUS = 'N';  
 ```  
  
This user story ensures that the annual income data for the insured person is accurately captured and validated, adhering to the business rules and conditions.

# Input Appointee's Name in Insured Person Section

Type: INSURED\_PERSON

Title: Input Appointee's Name in Insured Person Section  
  
Acceptance Criteria:  
1. The input field for the appointee's name should be labeled "App Name".  
2. The field should be positioned appropriately within the "Insured Person" section.  
3. The field should accept a maximum of 100 characters.  
4. The input should be restricted to uppercase letters.  
5. Upon entering the appointee's name and pressing the next item key, the focus should move to the "Method of Payment" field in the "Policy Holder" section.  
  
Definition of Done:  
1. The appointee's name field is visible and correctly labeled.  
2. The field accepts input as per the specified constraints.  
3. The navigation to the "Method of Payment" field works as expected upon pressing the next item key.  
4. All changes are tested and verified for accuracy and functionality.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include any specific database queries.

# Manage Probable CP Details for Insured Persons

Type: INSURED\_PERSON

Title: Manage Probable CP Details for Insured Persons  
  
Acceptance Criteria:  
1. When the "Probable CP" button is pressed, the system should display the "Probable CPs" window.  
2. The system should navigate to the "Probable CP" block.  
3. If the variable `v\_prob\_ip\_sel` is 'N', the system should clear the block.  
4. The system should fetch and display unique partner IDs from the `azbj\_auto\_cp\_merge\_details` table where the application number matches the insured person's verification or sign card number, and the partner type is 'IP' with a top indicator of 'Y'.  
5. For each partner ID, the system should:  
 - Fetch and display details such as partner ID, policy reference, name, annual premium, policy status, and sum insured from various tables.  
 - Attempt to fetch and display the `tasa` value and `suc` value from the `azbj\_ip\_ext` table, and if not found, from the `wip\_azbj\_ip\_ext` table.  
 - Fetch and display percentage values (`ml\_perc`, `oc\_perc`, `nri\_perc`, `sr\_perc`) from the `azbj\_policy\_covers\_ext` table, and if not found, from the `wip\_azbj\_policy\_covers\_ext` table.  
 - Determine if the policy is rated up based on the percentage values and set the `RATED\_UP` field accordingly.  
 - Fetch and display the package code from the `wip\_azbj\_policy\_bases\_ext` table, and if not found, from the `azbj\_policy\_bases\_ext` table.  
 - Fetch and display the product ID using a function call.  
6. The system should handle exceptions gracefully and log any errors encountered during the process.  
  
Definition of Done:  
- The "Probable CPs" window is displayed correctly when the button is pressed.  
- The "Probable CP" block is navigated to and populated with the correct data.  
- All specified data fields are fetched and displayed accurately.  
- The system handles exceptions and logs errors without crashing.  
- The user can navigate through the records and see the updated information.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Fetch unique partner IDs:  
 ```sql  
 SELECT unique part\_id  
 FROM azbj\_auto\_cp\_merge\_details  
 WHERE appln\_no = TO\_CHAR (NVL (:insured\_person.ip\_verf\_no, :insured\_person.ip\_sign\_card\_no))  
 AND partner\_type = 'IP'  
 AND top\_indicator = 'Y';  
 ```  
  
- Fetch partner details:  
 ```sql  
 SELECT partner\_id, a.contract\_id, f.booking\_frequency,  
 azbj\_pk0\_acc.get\_policy\_ref (a.contract\_id) policy\_ref,  
 e.sum\_insured\_whole\_cover, e.cover\_code,  
 b.before\_title ||' '||b.first\_name ||' '||b.surname name, g.ann\_prem,  
 c.change\_description status, c.product\_id, e.FTPREMIUM\_OR\_WHOLE\_COVER,  
 g.entry\_age, g.premium\_term, g.benefit\_term, g.col6 interest\_rate,  
 g.freq\_std\_prem, g.extra\_amt, g.ml\_perc, g.oc\_perc, g.nri\_perc, g.sr\_perc, g.prem\_disc\_amt  
 FROM ocp\_interested\_parties a, cp\_partners b, ocp\_policy\_versions c,  
 ocp\_policy\_covers e, ocp\_policy\_bases f,  
 azbj\_policy\_covers\_ext g, azbj\_policy\_contract\_ext h  
 WHERE a.partner\_id = cur.part\_id  
 AND a.partner\_id = b.part\_id AND a.contract\_id = c.contract\_id  
 AND a.contract\_id = e.contract\_id AND e.cover\_code LIKE 'L%'  
 AND f.contract\_id = c.contract\_id AND a.contract\_id = f.contract\_id  
 AND g.contract\_id = c.contract\_id AND a.contract\_id = g.contract\_id  
 AND g.contract\_id = f.contract\_id AND g.cover\_code LIKE 'L%'  
 AND h.contract\_id = c.contract\_id AND a.contract\_id = h.contract\_id  
 AND h.contract\_id = f.contract\_id;  
 ```  
  
- Fetch `tasa` and `suc` values:  
 ```sql  
 SELECT tasa\_value, bank\_ac\_no, col7  
 INTO :BLK\_PROBABLE\_CP.tasa, :BLK\_PROBABLE\_CP.suc, v\_cover\_type  
 FROM azbj\_ip\_ext  
 WHERE contract\_id = cur\_partner.contract\_id  
 AND ip\_no = 1;  
 ```  
  
- Fetch percentage values:  
 ```sql  
 SELECT nvl(ml\_perc,0), nvl(oc\_perc,0), nvl(nri\_perc,0), nvl(sr\_perc,0)  
 INTO v\_ml\_perc, v\_oc\_perc, v\_nri\_perc, v\_sr\_perc  
 FROM azbj\_policy\_covers\_ext  
 WHERE contract\_id = cur\_partner.contract\_id  
 AND (ml\_perc > 0 OR oc\_perc > 0 OR nri\_perc > 0 OR sr\_perc > 0)  
 AND cover\_code LIKE 'L%'  
 AND top\_indicator = 'Y'  
 AND action\_code <> 'D';  
 ```  
  
- Fetch package code:  
 ```sql  
 SELECT PACKAGE\_CODE  
 INTO v\_pack\_codee  
 FROM wip\_azbj\_policy\_bases\_ext  
 WHERE contract\_id = cur\_partner.contract\_id  
 AND top\_indicator = 'Y'  
 AND action\_code <> 'D';  
 ```

# Validate Insured Person's Weight

Type: INSURED\_PERSON

User Story: Validate Insured Person's Weight  
  
Detailed description:   
As a user, I want to ensure that the weight of the insured person is validated correctly based on specific product definitions and conditions, so that only valid weights are accepted and appropriate actions are taken based on the product and package codes.  
  
Acceptance criteria:  
1. If the global loading flag is set to 'F', the form status should be updated to 'Y'.  
2. The weight validation should only proceed if the product definition does not match specific patterns or values and the package code does not match specific patterns.  
3. If the weight is not null and the product definition and package code conditions are met, the weight must be greater than 0. If not, an error message should be displayed.  
4. If the premium frequency is not null, certain form properties should be disabled.  
  
Definition of Done:  
- The form status is updated correctly based on the global loading flag.  
- The weight validation logic is executed only when the product definition and package code conditions are met.  
- An error message is displayed if the weight is invalid.  
- Form properties are disabled when the premium frequency is not null.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Not applicable as the provided XML content does not include direct CRUD operations on the database.  
  
Explanation of Oracle Form Logics:  
- The logic checks the global loading flag and updates the form status accordingly.  
- It validates the weight based on specific product definitions and package codes.  
- If the weight is invalid, an error message is displayed.  
- Certain form properties are disabled based on the premium frequency.

# Manage Bank BM Certificate Checkbox

Type: INSURED\_PERSON

Title: Manage Bank BM Certificate Checkbox  
  
Acceptance Criteria:  
1. When the "Bank BM Certificate" checkbox is checked, the system should disable the "horizontal\_toolbar.commit\_form" button.  
2. The checkbox should have a checked value of "Y" and an unchecked value of "N".  
3. The checkbox should be displayed with a label "Bank BM Certificate" and should be positioned correctly within the form.  
  
Definition of Done:  
1. The "Bank BM Certificate" checkbox is implemented and visible in the user interface.  
2. The checkbox correctly toggles between checked and unchecked states.  
3. The "horizontal\_toolbar.commit\_form" button is disabled when the checkbox is checked.  
4. The checkbox is labeled "Bank BM Certificate" and is positioned as specified.  
5. The feature is tested and verified to work as expected.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- No specific database queries are provided in the XML content.

# Load Data for Insured Person on Button Press

Type: INSURED\_PERSON

Title: Load Data for Insured Person on Button Press  
  
Acceptance Criteria:  
1. If the verification number for the insured person is not null:  
 - If the product ID is less than 59, the system should load the new business data using the verification number.  
 - If the employee ID of the insured person is not null and its length is greater than 1, the system should navigate to the employee list.  
 - If the employee ID is null, the system should navigate to the fund list and calculate the policy year based on the effective date and contract ID.  
 - The system should then retrieve the default allocation percentage from the `azbj\_cover\_event\_values` table based on the policy year, product ID, event code, term, and the permanent record date.  
2. If the verification number is null, the system should display a message prompting the user to enter the verification number first.  
3. After processing, the system should navigate to the insured person's name field.  
  
Definition of Done:  
- The button should trigger the data loading process as described.  
- The system should handle navigation and data retrieval based on the specified conditions.  
- Appropriate messages should be displayed if required conditions are not met.  
- The system should navigate to the insured person's name field after processing.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Retrieve default allocation percentage:  
 ```sql  
 SELECT DEFAULT\_VALUE  
 INTO :Allocation.alloct\_perc  
 FROM azbj\_cover\_event\_values  
 WHERE Mn\_policy\_yr BETWEEN pol\_year\_from AND pol\_year\_to  
 AND product\_id = :control.cn\_product\_id  
 AND Event\_code = 'ALL'  
 AND pk\_vars.v\_perm\_rec\_date >= start\_date  
 AND pk\_vars.v\_perm\_rec\_date <= NVL(end\_date, TO\_DATE('01/01/3000', 'dd/mm/yyyy'));  
 ```

# Generate New Proposal Number for Insured Person

Type: INSURED\_PERSON

Title: Generate New Proposal Number for Insured Person  
  
Acceptance Criteria:  
- When the user navigates to the "New Proposal Number" field, the system should automatically trigger the logic to generate a new proposal number.  
- The new proposal number should be displayed in the "New Proposal Number" field.  
- The field should be located at the specified position on the form and should have a maximum length of 50 characters.  
- The field should be visually styled with the specified font, color, and size attributes.  
  
Definition of Done:  
- The "New Proposal Number" field is implemented and visible on the form.  
- The field correctly triggers the logic to generate a new proposal number when the user navigates to it.  
- The field adheres to the specified visual and positional attributes.  
- The functionality is tested and verified to ensure it meets the acceptance criteria.

# Validate Age Proof for Insured Person

Type: INSURED\_PERSON

User Story: Validate Age Proof for Insured Person  
  
Detailed Description:  
As a system, I need to validate the age proof provided by the insured person to ensure it meets the required criteria and to display appropriate messages based on the validation results.  
  
Acceptance Criteria:  
1. If the premium frequency is not null, disable the form's commit and exit buttons.  
2. If the age proof provided by the insured person is 'SYBM' and the agent code does not start with '522', display an error message stating that the Syndicate Bank BM Certificate is allowed only for Syndicate Bank.  
3. On double-clicking the age proof field, display a list of valid age proof types for selection.  
4. The list of valid age proof types should be fetched from the database where the document type is 'AGE\_PROOF' and the enabled flag is 'Y'.  
  
Definition of Done:  
- The system should correctly validate the age proof based on the provided criteria.  
- Appropriate error messages should be displayed when the validation fails.  
- The list of valid age proof types should be displayed on double-clicking the age proof field.  
- The system should fetch the list of valid age proof types from the database as specified.  
  
DB Queries for Table Reference CRUD Operations:  
```sql  
SELECT PROOF\_TYPE, PROOF\_DESC  
FROM azbj\_aml\_requirements  
WHERE DOCUMENT\_TYPE = 'AGE\_PROOF'  
AND ENABLED\_FLAG = 'Y'  
ORDER BY PROOF\_TYPE;  
```

# Manage Lateral Shift Status Using Checkbox

Type: INSURED\_PERSON

Title: Manage Lateral Shift Status Using Checkbox  
  
Acceptance Criteria:  
1. When the checkbox is checked, the system should verify if the insured person's ID matches any record in the lateral shift data.  
2. If a match is found and the checkbox is checked, the system should set a flag to 'Y'.  
3. If a match is found and the checkbox is unchecked, the system should delete the corresponding lateral shift record.  
4. If no match is found, the system should create a new lateral shift record with the insured person's ID, set the shift number to 1, and update the partner name.  
  
Definition of Done:  
- The checkbox functionality should be implemented and tested.  
- The system should correctly update, delete, or create lateral shift records based on the checkbox status.  
- All acceptance criteria should be met and verified through testing.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- No specific DB queries provided in the XML content.

# Search for Customer Partner Information

Type: INSURED\_PERSON

User Story: Search for Customer Partner Information  
  
Detailed Description:  
As a user, I want to search for customer partner information when I press the "Search CP" button, so that I can retrieve and display relevant details about the customer partner.  
  
Acceptance Criteria:  
1. When the "Search CP" button is pressed, the system should:  
 - Set a flag indicating that the partner ID is being searched.  
 - Check if a control flag is set to 'Y'. If it is, another control flag should be set to 'Y'; otherwise, it should be set to 'N'.  
 - Retrieve the partner ID from the insured person’s details.  
 - Navigate to the insured person’s block and call a procedure to fetch customer partner details.  
 - Display a warning message if the agent code matches specific patterns and the partner ID is the same as the LG partner ID.  
 - If the partner ID has changed, clear certain fields related to the insured person and delete related records from another block.  
 - Populate a list of partners and update certain fields based on the product definition.  
 - Adjust visibility and enablement of various fields based on conditions related to the policy holder and product ID.  
 - Calculate the entry age of the insured person and update related fields in the covers and loadings blocks.  
 - Update occupation class and percentage based on the insured person’s occupation code and industry.  
 - Check for NRI loading and update related fields.  
 - If the insured person is a minor, enable or disable specific fields based on their age.  
 - Populate income proof lists for both the insured person and the policy holder based on their occupation status.  
  
Definition of Done:  
- The "Search CP" button functionality is implemented as described.  
- All conditions and field updates are correctly handled.  
- The system displays appropriate messages and updates fields based on the logic.  
- The functionality is tested and verified to work as expected.  
  
DB Queries for Table Reference CRUD Operations:  
- The provided logic includes several database operations such as retrieving partner IDs, updating fields, and deleting records. These operations should be implemented using appropriate SQL queries within the application logic, ensuring they are independent of Oracle Forms-specific constructs.

# Auto-populate Telephone2 field on double-click

Type: INSURED\_PERSON

Title: Auto-populate Telephone2 field on double-click  
  
Acceptance Criteria:  
1. When the user double-clicks on the Telephone2 field, the system should check if the field is empty.  
2. If the Telephone2 field is empty, it should be automatically populated with the value 'Not Available'.  
3. If the Telephone2 field already contains a value, no changes should be made.  
  
Definition of Done:  
- The functionality to check and populate the Telephone2 field is implemented.  
- The feature is tested and verified to ensure that it works as expected.  
- The user interface reflects the changes without any errors or disruptions.  
- Documentation is updated to include this new behavior.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided logic does not involve direct database CRUD operations.

# Validate Insured Person's Height

Type: INSURED\_PERSON

User Story: Validate Insured Person's Height  
  
Detailed description:   
As a system, I need to validate the height of the insured person to ensure it meets the required criteria based on the product definition and other conditions.  
  
Acceptance criteria:  
1. The system should check if the product definition of the insured person's product ID does not match specific product names or codes.  
2. The system should ensure that the package code does not match certain patterns.  
3. The system should verify that the pension flag is set to 0.  
4. If the height of the insured person is not null, the system should further check that the product definition is not in a specified list and certain flags are not set.  
5. If the height is less than or equal to 0, the system should display an error message: "Please Select a Valid Height for the Person".  
6. If the global loading flag is set to 'F', the form status should be updated to 'Y'.  
7. If the frequency premium is not null, certain form properties should be disabled.  
  
Definition of Done:  
- The height validation logic is implemented and integrated into the system.  
- The system correctly identifies invalid heights and displays the appropriate error message.  
- The form status and properties are updated based on the specified conditions.  
- All acceptance criteria are met and tested.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Not applicable as the provided logic does not include direct CRUD operations on the database.  
  
Explanation of Oracle Form Logic:  
- The logic involves checking multiple conditions related to the product definition, package code, and pension flag.  
- It ensures that the height of the insured person is valid and within acceptable limits.  
- The form status and certain properties are updated based on the conditions specified in the logic.

# Input Fresh Proposal Sign Date

Type: INSURED\_PERSON

Title: Input Fresh Proposal Sign Date  
  
Acceptance Criteria:  
1. The date input field should accept dates in the format "dd/mm/yyyy".  
2. The date input field should be labeled as "Fresh Prop Date".  
3. When the user inputs a date and presses the key to move to the next item, the system should trigger the logic to generate a new proposal number.  
  
Definition of Done:  
1. The date input field is visible and correctly labeled.  
2. The date input field accepts and validates dates in the specified format.  
3. The logic to generate a new proposal number is triggered upon moving to the next item.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include specific CRUD operations.

# Verification Number Validation for Unit Linked Policies

Type: INSURED\_PERSON

Title: Verification Number Validation for Unit Linked Policies  
  
Acceptance Criteria:  
1. If the verification number is not provided, an error message "Verification Number Is mandatory For Unit Linked" should be displayed.  
2. If the verification number and receipt number do not match, an error message "Verification Number And Receipt No. Is Not Being Matched !!!" should be displayed.  
3. If the verification number and receipt number match:  
 - If the product ID is less than 59, the "Load UL" button should be enabled and the focus should move to this button.  
 - If the product ID is 59 or greater, the "Benefit Term" field should be enabled.  
4. The system should check the count of records in the `wip\_policy\_bases` table for the given contract ID. If the count is greater than 0:  
 - The system should navigate to the "Top-Up" block and display the first record.  
 - The system should loop through the records in the `AZBJ\_BATCH\_ITEMS` and `AZBJ\_CASHIER\_COLL` tables where the verification number matches and the transaction type is 'TPR', and populate the "Top-Up" block with the receipt number, receipt date, amount, and transaction type.  
  
Definition of Done:  
- The user is able to input a verification number and the system validates it against the receipt number.  
- Appropriate error messages are displayed if the verification number is missing or does not match the receipt number.  
- The system enables the correct fields/buttons based on the product ID.  
- The system checks the contract ID in the `wip\_policy\_bases` table and navigates to the "Top-Up" block if records are found, populating it with the relevant data.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Select 'Y' Into v\_flag From `AZBJ\_BATCH\_ITEMS` Where `Verification\_no` = :Insured\_person.ip\_verf\_no And `perm\_receipt\_no` = :SOFA\_INFO\_DATA.perm\_rcpt\_no;  
- Select count(0) into x\_count from `wip\_policy\_bases` where `contract\_id` = :control.cn\_contract\_id;  
- For I in (Select `PERM\_RECEIPT\_NO`, `PERM\_RECEIPT\_DATE`, `AMOUNT`, `TRANSACTION\_TYPE` From `AZBJ\_BATCH\_ITEMS` A, `AZBJ\_CASHIER\_COLL` B Where A.`CASHIER\_BATCH\_ID` = B.`CASHIER\_BATCH\_ID` And A.`COLLECTION\_NO` = B.`COLLECTION\_NO` And A.`VERIFICATION\_NO` = :INSURED\_PERSON.IP\_VERF\_NO And A.`TRANSACTION\_TYPE` = 'TPR' And B.`CDA\_ACC\_BATCH` Is Null And NVL(`PRINT`, 'C') <> 'C' And `AC\_PASS` = 'Y') Loop

# Search Mailing Address for Insured Person

Type: INSURED\_PERSON

Title: Search Mailing Address for Insured Person  
  
Acceptance Criteria:  
1. If no partner is selected, an error message should be displayed indicating that a partner must be selected before searching for a mailing address.  
2. Upon pressing the search button, the system should call the address search form.  
3. The selected mailing address should be retrieved and displayed in the appropriate fields.  
4. If no mailing address is selected, a message should prompt the user to select a mailing address.  
5. The mailing address details should be populated in the insured person's record.  
6. If the policy holder's mailing address is set to be the same as the insured person's, it should be updated accordingly.  
  
Definition of Done:  
- The error message is displayed when no partner is selected.  
- The address search form is successfully called.  
- The mailing address details are correctly retrieved and displayed.  
- The user is prompted to select a mailing address if none is selected.  
- The insured person's mailing address fields are populated with the correct data.  
- The policy holder's mailing address is updated if it is set to be the same as the insured person's.  
  
DB queries for Table reference CRUD operations only(With Usage):  
```sql  
SELECT azbj\_pk\_policy\_schedule1.get\_son\_of\_wife\_of(:INSURED\_PERSON.IP\_PART\_ID)||CHR(10) ||  
 ADDRESS\_LINE1 || CHR(10) || ADDRESS\_LINE2 ||  
 tae.AREA\_FREE\_TEXT\_M ||  
 CASE WHEN tae.AREA\_FREE\_TEXT\_M IS NOT NULL THEN CHR(10) ELSE NULL END ||  
 CHR(10) || ADDRESS\_LINE3 || CHR(10) ||  
 ADDRESS\_LINE4 || CHR(10) || ADDRESS\_LINE5 || CHR(10) ||  
 ADDRESS\_LINE6 || CHR(10) || ADDRESS\_LINE7 || 'COUNTRY- ' || UPPER(DESCRIPTION) || ' ' || POSTCODE, POSTCODE, CPA.COUNTRY\_CODE,  
 TELEPHONE2, TELEPHONE, MOBILE\_NO  
INTO :insured\_person.IP\_MAIL\_ADDR, :INSURED\_PERSON.IP\_MAILING\_POSTCODE, :INSURED\_PERSON.IP\_MAILING\_COUNTRY\_CODE,  
 :INSURED\_PERSON.IP\_CONTACT\_TELNO2, :INSURED\_PERSON.IP\_CONTACT\_TELNO, :INSURED\_PERSON.IP\_CONTACT\_MOBILE  
FROM CP\_ADDRESSES CPA, AZBJ\_ADDRESS\_EXTN TAE, CP\_COUNTRIES CP  
WHERE CPA.ADD\_ID = :IP\_MAIL\_ID  
 AND TAE.ADD\_ID = CPA.ADD\_ID  
 AND CPA.COUNTRY\_CODE = CP.COUNTRY\_CODE;  
```

# Validate PAN Issuance Date for Insured Person

Type: INSURED\_PERSON

Detailed description: As a user, I want to ensure that the PAN Issuance Date for an insured person is not set to a future date, so that the data integrity is maintained and only valid dates are entered.  
  
Acceptance criteria:  
1. The system should validate the PAN Issuance Date when it is entered.  
2. If the PAN Issuance Date is greater than the current system date, an error message should be displayed: "PAN Issuance Date should not be less than system date."  
  
Definition of Done:  
1. The validation logic for the PAN Issuance Date is implemented.  
2. The error message is displayed correctly when the PAN Issuance Date is set to a future date.  
3. 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 direct database operations.

# Display and Navigation for Father Name Field

Type: INSURED\_PERSON

Title: Display and Navigation for Father Name Field  
  
Acceptance Criteria:  
1. The "Father Name" field should be displayed within the "Insured Person" section.  
2. The field should have a maximum length of 100 characters.  
3. The field should not allow insertion or updates.  
4. The field should be positioned at coordinates (85, 52) on the screen.  
5. The field should be labeled "Father Name" with a bold font style.  
6. The field should not be keyboard navigable.  
7. When the user attempts to navigate to the next item, the system should automatically move to the "Method of Payment" field in the "Policy Holder" section.  
  
Definition of Done:  
- The "Father Name" field is correctly displayed and positioned within the "Insured Person" section.  
- The field adheres to the specified constraints (maximum length, no insertion or updates).  
- The field label is correctly styled and positioned.  
- The navigation logic to the "Method of Payment" field is implemented and functional.  
  
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 Navigate 'Spouse Name' Field

Type: INSURED\_PERSON

Title: View and Navigate 'Spouse Name' Field  
  
Acceptance Criteria:  
1. The 'Spouse Name' field should be displayed within the 'Insured Person' section.  
2. The field should be read-only and not allow any insert or update operations.  
3. The field should have a maximum length of 100 characters.  
4. When navigating through the form, pressing the key to move to the next item should automatically navigate to the 'Method of Payment' field in the 'Policy Holder' section.  
  
Definition of Done:  
1. The 'Spouse Name' field is visible and correctly positioned within the 'Insured Person' section.  
2. The field is read-only and does not allow any modifications.  
3. The field adheres to the maximum length constraint of 100 characters.  
4. Navigation logic is implemented such that moving to the next item from this field correctly navigates to the 'Method of Payment' field in the 'Policy Holder' section.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the field is not a database item and does not involve any CRUD operations.

# Handle PAN Issuance Date Not Available Checkbox Functionality

Type: INSURED\_PERSON

Title: Handle PAN Issuance Date Not Available Checkbox Functionality  
  
Acceptance Criteria:  
1. When the "PAN Issuance Date Not Available" checkbox is checked:  
 - The "PAN Issuance Date" field should be disabled.  
 - The "PAN Issuance Date" field should be cleared.  
 - The system should save the state indicating that the PAN issuance date is not available.  
2. When the "PAN Issuance Date Not Available" checkbox is unchecked:  
 - The "PAN Issuance Date" field should be enabled.  
 - If the "PAN Issuance Date" field is not empty, the system should save the state indicating that the PAN issuance date is available.  
  
Definition of Done:  
- The functionality should be implemented and tested.  
- The "PAN Issuance Date Not Available" checkbox should correctly enable/disable and clear the "PAN Issuance Date" field based on its state.  
- The system should save the appropriate state based on the checkbox status.  
- All acceptance criteria should be met and verified through testing.

# Surrogate Income Proof Selection

Type: INSURED\_PERSON

Title: Surrogate Income Proof Selection  
  
Acceptance Criteria:  
1. When the user selects an item from the surrogate income proof list, the system should display the detailed view for surrogate income proof.  
2. The system should automatically navigate to the surrogate income proof type field within the detailed view.  
3. If any error occurs during this process, the system should handle it gracefully without crashing.  
  
Definition of Done:  
- The surrogate income proof list is displayed and selectable.  
- Upon selection, the detailed view for surrogate income proof is shown.  
- The system navigates to the surrogate income proof type field.  
- Error handling is implemented to manage any exceptions that may occur during the process.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- No specific database queries are mentioned in the provided XML content.

# Input and Selection of Income Proof for Insured Person

Type: INSURED\_PERSON

Title: Input and Selection of Income Proof for Insured Person  
  
Acceptance Criteria:  
1. The input field for income proof should be clearly labeled as "IP income proof".  
2. The field should be a list item, allowing users to select from predefined options.  
3. The field should be positioned appropriately within the form for easy access.  
4. The field should have a maximum length of 100 characters.  
5. The field should be visually distinct with a white background and black text.  
6. The field should be part of the "INSURED\_PERSON" section within the New Business module.  
  
Definition of Done:  
1. The "IP income proof" field is implemented and visible in the form.  
2. Users can select from predefined options in the "IP income proof" field.  
3. The field adheres to the specified visual and positional requirements.  
4. The field is tested and verified to ensure it meets the acceptance criteria.  
5. 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 database queries.

# Dropdown List for Insured Person's Mother's Title

Type: INSURED\_PERSON

Title: Dropdown List for Insured Person's Mother's Title  
  
Acceptance Criteria:  
1. The title field for the insured person's mother should be a dropdown list.  
2. The dropdown list should be initialized with the value "Mrs".  
3. The dropdown list should be disabled by default, meaning users cannot modify it.  
4. The dropdown list should have a maximum length of 5 characters.  
5. The dropdown list should be displayed on the "Insured Person" tab.  
6. The dropdown list should be positioned at coordinates (17, 88) on the tab.  
7. The dropdown list should have a width of 53 and a height of 18.  
8. The dropdown list should have a white background and black text.  
9. The dropdown list should use the "MS Sans Serif" font with a size of 8, plain style, medium weight, and normal spacing.  
  
Definition of Done:  
- The dropdown list for the insured person's mother's title is implemented and meets all acceptance criteria.  
- The dropdown list is tested to ensure it displays correctly and is initialized with the value "Mrs".  
- The dropdown list is verified to be disabled and unmodifiable by users.  
- The dropdown list is positioned correctly on the "Insured Person" tab and adheres to the specified dimensions and styling.  
- The feature is reviewed and approved by stakeholders.

# Account Type Field for Insured Person

Type: INSURED\_PERSON

Title: Account Type Field for Insured Person  
  
Acceptance Criteria:  
1. The account type field should be displayed on the "Insured Person" tab.  
2. The account type field should be a dropdown list containing predefined account types.  
3. The account type field should be disabled for editing directly and should only allow selection from the dropdown list.  
4. The account type field should be positioned at the specified location on the form.  
5. The account type field should have a prompt labeled "Account Type" aligned to the right of the field.  
6. The account type field should be restricted to uppercase input.  
7. The account type field should have a maximum length of 50 characters.  
  
Definition of Done:  
1. The account type field is visible on the "Insured Person" tab.  
2. The account type field is a dropdown list with predefined values.  
3. The account type field is disabled for direct text input.  
4. The account type field is correctly positioned and labeled as per the design specifications.  
5. The account type field only accepts uppercase input and has a maximum length of 50 characters.  
6. The changes are tested and verified to meet the acceptance criteria.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Not applicable as the provided XML content does not include any specific database queries.

# Manage Reinsurance Cover Details

Type: AZBJ\_REINS\_C\_OFFER

Detailed description: As a user, I want to view and manage reinsurance cover details on a dedicated screen, so that I can efficiently handle reinsurance offers.  
  
Acceptance criteria:  
1. The screen should display the following fields:  
 - Cover Code (read-only)  
 - Cover Description (read-only)  
 - OC Percentage (default value: 0)  
 - ML Percentage (default value: 0)  
 - SP Percentage (default value: 0)  
 - NRI Percentage (default value: 0)  
 - SA (default value: 0)  
2. There should be a button labeled "Delete" to remove selected reinsurance cover details.  
  
Definition of Done:  
1. The screen is accessible and displays all the specified fields.  
2. The fields "Cover Code" and "Cover Description" are read-only.  
3. The fields "OC Percentage," "ML Percentage," "SP Percentage," "NRI Percentage," and "SA" are editable and initialized to 0.  
4. The "Delete" button is functional and removes the selected reinsurance cover details.  
5. The screen layout is user-friendly and matches the specified design.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include specific database queries.

# Delete Record Functionality

Type: AZBJ\_REINS\_C\_OFFER

Title: Delete Record Functionality  
  
Acceptance Criteria:  
- When the "Delete" button is pressed, the system should delete the currently selected record from the database.  
- The deletion should be confirmed by the system, ensuring that the record is no longer present in the database.  
  
Definition of Done:  
- The "Delete" button is visible and labeled correctly.  
- Pressing the "Delete" button successfully removes the selected record from the database.  
- The user receives a confirmation that the record has been deleted.  
  
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.

# Display Reinsurer Selection Screen

Type: RI\_REQ

Title: Display Reinsurer Selection Screen  
  
Acceptance Criteria:  
1. The reinsurer selection screen should display the following fields:  
 - Test Code (SYS\_CODE)  
 - Test Description (SYS\_DESC)  
 - Life Insurance Type (IP\_TYPE)  
 - Called Date (CALLED\_DATE)  
2. All fields should be read-only and not allow insertion or updates.  
3. The fields should be displayed in a user-friendly manner with appropriate labels and alignment.  
  
Definition of Done:  
1. The reinsurer selection screen is implemented and displays the required fields.  
2. The fields are read-only and cannot be modified by the user.  
3. The screen layout is user-friendly and matches the specified design.  
4. The functionality is tested and verified to meet the acceptance criteria.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- No specific DB queries provided in the XML content.

# Implement Insurance Policy Type Dropdown

Type: RI\_REQ

Title: Implement Insurance Policy Type Dropdown  
  
Acceptance Criteria:  
1. The insurance policy type should be presented as a dropdown list with predefined options.  
2. The dropdown list should be non-editable and should not allow new entries.  
3. The dropdown list should be displayed on the "Reinsurer Selection Screen".  
4. The dropdown list should be positioned at the specified coordinates on the screen.  
5. The dropdown list should have a white background and should be styled with a bold, plain Tahoma font for the prompt "Life".  
6. The dropdown list should contain exactly 10 elements.  
  
Definition of Done:  
- The dropdown list for selecting the insurance policy type is implemented and visible on the "Reinsurer Selection Screen".  
- The dropdown list is non-editable and contains 10 predefined options.  
- The dropdown list is styled and positioned as specified.  
- The feature has been tested and verified to meet the acceptance criteria.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- No specific database queries are provided in the XML content.

# Display and Interact with Reasons List

Type: REASON

Title: Display and Interact with Reasons List  
  
Acceptance Criteria:  
1. The list of reasons should be displayed with a maximum of 6 records at a time.  
2. The reasons should be displayed in a text format with a maximum length of 200 characters.  
3. The reasons should be read-only and not allow any insert or update operations.  
4. The list should have a scrollbar to navigate through the records if there are more than 6 reasons.  
5. The scrollbar should be positioned appropriately within the section for easy access.  
  
Definition of Done:  
- The list of reasons is displayed correctly with a maximum of 6 records at a time.  
- The reasons are displayed in a read-only text format with a maximum length of 200 characters.  
- The scrollbar is functional and allows users to navigate through the list of reasons.  
- The user interface is tested and verified to ensure it meets the acceptance criteria.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the operations are read-only and no direct CRUD operations are mentioned in the provided XML content.

# View Previous Underwriting and Reinsurance Comments

Type: RI\_PREV\_COMMENTS

Title: View Previous Underwriting and Reinsurance Comments  
  
Acceptance Criteria:  
1. The system should display the following fields in a read-only format:  
 - Underwriting Previous Recommendation  
 - Reinsurance Previous Recommendation  
 - Underwriting Final Comment  
 - Reinsurance Final Comment  
 - Underwriting Medical Comment  
 - Reinsurance Medical Comment  
 - Underwriting Financial Comment  
 - Reinsurance Financial Comment  
 - Serial Number (SR No)  
2. Each field should have a maximum length of 500 characters, except for the Serial Number which should be numeric.  
3. The fields should be displayed in a specific layout with defined positions and sizes to ensure readability and consistency.  
  
Definition of Done:  
- The user can view all the specified fields in a read-only format.  
- The fields are displayed with the correct labels and in the specified layout.  
- The data displayed in these fields is accurate and up-to-date.  
- The user interface is intuitive and easy to navigate.  
  
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 table references.

# Display Hierarchical Tree for Agent Data Analysis

Type: DATA

Title: Display Hierarchical Tree for Agent Data Analysis  
  
Acceptance Criteria:  
1. The hierarchical tree should display agent data analysis based on the following query logic:  
 - Select data from `azbj\_bbu\_prameter\_master` and `azbj\_agents\_data\_analysis` tables.  
 - The data should be filtered by `branch\_code` and `agent\_code`.  
 - The hierarchical structure should be created using the `CONNECT BY PRIOR` clause, starting with `master\_id` being NULL.  
 - The data should be ordered by level in ascending order and by `display\_name` or `display\_field\_name` in descending order.  
  
Definition of Done:  
1. The hierarchical tree structure is implemented and displays the correct data.  
2. The data is correctly filtered based on the user's branch code and agent code.  
3. The hierarchical tree is visually clear and easy to navigate.  
4. The feature is tested and verified to ensure it meets the acceptance criteria.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- The following query is used to fetch the data for the hierarchical tree:  
 ```sql  
 SELECT -1, LEVEL, NVL(display\_name, display\_field\_name), display\_field\_name, color\_code  
 FROM (  
 SELECT display\_field\_name, master\_id, ID, flag\_value, color\_code, display\_name  
 FROM (  
 SELECT display\_field\_name, NULL, master\_id, ID, item  
 FROM azbj\_bbu\_prameter\_master  
 ) a,  
 (  
 SELECT   
 FROM azbj\_agents\_data\_analysis  
 WHERE branch\_code = :Control.cn\_agent\_grid\_branch\_code  
 AND NVL(agent\_code, :Agents.ag\_agent\_code) = :Agents.ag\_agent\_code  
 ) b  
 WHERE a.item = b.flag(+)  
 )  
 CONNECT BY PRIOR ID = master\_id  
 START WITH master\_id IS NULL  
 ORDER BY level ASC, NVL(display\_name, display\_field\_name) DESC;  
 ```

# Manage PAN Approval Process

Type: BLK\_PAN\_APPROVAL

Title: Manage PAN Approval Process  
  
Acceptance Criteria:  
1. The user should be able to input the age proof and age proof ID.  
2. The user should be able to input the PAN issuance date.  
3. The user should be able to provide underwriting reasons.  
4. The user should be able to input supervisor comments.  
5. The user should be able to approve the application by clicking the "Supervisor Approval" button.  
6. The user should be able to exit the approval process by clicking the "EXIT" button.  
  
Definition of Done:  
1. The form should allow the user to input and save age proof and age proof ID.  
2. The form should allow the user to input and save the PAN issuance date.  
3. The form should allow the user to input and save underwriting reasons.  
4. The form should allow the user to input and save supervisor comments.  
5. The "Supervisor Approval" button should trigger the approval process.  
6. The "EXIT" button should close the form and exit the approval process.  
  
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 table references.

# Exit PAN Approval and Navigate to Client Environment Validation

Type: BLK\_PAN\_APPROVAL

Title: Exit PAN Approval and Navigate to Client Environment Validation  
  
Acceptance Criteria:  
1. When the exit button is pressed, the PAN approval screen should be hidden.  
2. The system should navigate to the client environment validation screen.  
3. The validation process should be triggered automatically upon navigation to the client environment validation screen.  
  
Definition of Done:  
- The exit button hides the PAN approval screen.  
- The system successfully navigates to the client environment validation screen.  
- The validation process is triggered without any errors.  
  
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.

# Validate and Process PAN Card Approval

Type: BLK\_PAN\_APPROVAL

Title: Validate and Process PAN Card Approval  
  
Acceptance Criteria:  
1. When the "Supervisor Approval" button is pressed, the system should:  
 - Set the global approval status to 'N'.  
 - Check if both the "UW Reason" and "Supervisor Comment" fields are not null.  
 - Validate that the "UW Reason" and "Supervisor Comment" fields do not contain any special characters.  
 - Display an error message if special characters are found in either field.  
 - If validation passes, call the form 'AZBJ\_RP\_ERR\_APP' and update the global approval status.  
 - Display an error message if either "UW Reason" or "Supervisor Comment" is missing.  
2. Increment the approval count.  
3. Save the approval status using a standard save procedure.  
4. If the approval status does not start with 'Y-' and the approval count is 1:  
 - Navigate to the 'result' block and move to the last record.  
 - If there are records, move to the next record; otherwise, move to the first record.  
 - Update the 'result' block with specific messages indicating the PAN card is considered non-standard age proof.  
 - If the approval status is not 'Y-', set the global non-standard age proof status to 'N'.  
  
Definition of Done:  
- The approval process should validate and handle the input fields as described.  
- Appropriate error messages should be displayed for invalid inputs.  
- The form should be called and the global approval status updated correctly.  
- The result block should be updated with the correct messages based on the approval status.  
- The process should be thoroughly tested to ensure all conditions are met and the system behaves as expected.

# Display Approval Details

Type: BLK\_PAN\_APPROVE\_DET

Title: Display Approval Details  
  
Acceptance Criteria:  
1. The system should display the following fields:  
 - Approval ID  
 - Approval Name  
 - Approval Date  
 - Approval Time  
 - Approval Decision  
2. Each field should be displayed with appropriate labels and aligned properly for readability.  
3. The approval date should be displayed in a date format.  
4. The approval time should be displayed in a time format.  
5. There should be an "Exit" button that allows the user to close the approval details view.  
  
Definition of Done:  
- The approval details view is implemented and displays all required fields.  
- The fields are properly labeled and aligned.  
- The date and time fields are displayed in the correct formats.  
- The "Exit" button is functional and closes the view.  
- The feature is tested and verified to meet the acceptance criteria.

# Exit PAN Approved Details View

Type: BLK\_PAN\_APPROVE\_DET

Title: Exit PAN Approved Details View  
  
Acceptance Criteria:  
1. When the "EXIT" button is pressed, the "PAN Approved Details" view should be hidden.  
2. The main "New Business" tab view should be displayed.  
3. The focus should be set to the "PAN Approve Details" item in the main view.  
  
Definition of Done:  
- The "EXIT" button successfully hides the "PAN Approved Details" view.  
- The main "New Business" tab view is displayed.  
- The focus is correctly set to the "PAN Approve Details" item in the main view.  
- The functionality is tested and verified to work as expected.

# Manage Beneficiary and Appointee Details

Type: BENEFICIARIES

Detailed description: As a user, I want to manage the details of beneficiaries and appointees for a policy, so that I can ensure all necessary information is captured accurately.  
  
Acceptance criteria:  
1. The system should allow the user to enter the following details for each beneficiary:  
 - Name  
 - Surname  
 - Date of Birth  
 - Place of Birth  
 - Relationship  
 - Share percentage  
 - Gender  
  
2. The system should allow the user to enter the following details for each appointee:  
 - Name  
 - Relationship  
 - Date of Birth  
 - Gender  
  
3. The system should validate the following conditions:  
 - If no beneficiary is selected, appointee details should not be entered. If attempted, an error message should be displayed, and the appointee details should be cleared.  
 - If the beneficiary is a major (age > 18 years), appointee details should not be entered. If attempted, an error message should be displayed, and the appointee details should be cleared.  
 - If the appointee is a minor (age < 18 years), an error message should be displayed, and the form submission should be halted.  
  
4. The system should provide a list of relationships for selection when entering the relationship of a beneficiary or appointee.  
  
5. The system should allow the user to delete a nominee and add multiple nominees.  
  
Definition of Done:  
- The user can successfully enter and save beneficiary and appointee details.  
- All validation rules are enforced, and appropriate error messages are displayed when conditions are not met.  
- The user can select relationships from a predefined list.  
- The user can delete a nominee and add multiple nominees as needed.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- The system should fetch the list of relationships from the database using the query: `select CHAR\_VALUE from azbj\_system\_constants where sys\_type='NOM\_DET'`.

# Validate Share Percentage for Beneficiaries

Type: BENEFICIARIES

Title: Validate Share Percentage for Beneficiaries  
  
Acceptance Criteria:  
- If the share percentage entered for a beneficiary is greater than 100%, an error message should be displayed stating, "Share Percentage cannot be more than 100%."  
  
Definition of Done:  
- The system should validate the share percentage input.  
- An error message should be triggered if the share percentage exceeds 100%.  
- The error message should be clear and instructive to the user.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include any database-specific queries.

# Dynamic Adjustment of Beneficiary Fields Based on 'Nomination not opted' Checkbox

Type: BENEFICIARIES

Title: Dynamic Adjustment of Beneficiary Fields Based on 'Nomination not opted' Checkbox  
  
Acceptance Criteria:  
1. When the 'Nomination not opted' checkbox is checked:  
 - The fields for beneficiary name, date of birth, place of birth, relation, applicant name, applicant relation, applicant date of birth, multiple nominees, and delete beneficiary should be hidden and disabled.  
 - The beneficiary name field should display the text 'Nomination not opted'.  
 - All other fields should be cleared of any existing values.  
  
2. When the 'Nomination not opted' checkbox is unchecked:  
 - The fields for beneficiary name, date of birth, place of birth, relation, applicant name, applicant relation, applicant date of birth, multiple nominees, and delete beneficiary should be visible and enabled.  
  
Definition of Done:  
- The form dynamically adjusts the visibility and enablement of the specified fields based on the state of the 'Nomination not opted' checkbox.  
- The beneficiary name field displays 'Nomination not opted' when the checkbox is checked.  
- All other fields are cleared when the checkbox is checked.  
- The changes are tested and verified to ensure they meet the acceptance criteria.  
- The functionality is documented for future reference.

# Gender Selection for Nominee

Type: BENEFICIARIES

Title: Gender Selection for Nominee  
  
Acceptance Criteria:  
1. The gender selection should be available as a dropdown list with predefined options.  
2. The options in the dropdown list should be displayed in uppercase.  
3. The gender field should be positioned correctly on the form and should be clearly labeled as "Gender".  
4. The label for the gender field should be right-aligned and bold.  
5. The gender field should be part of the "POLICY\_HOLDER" tab within the form.  
  
Definition of Done:  
1. The gender dropdown list is implemented and displays the predefined options.  
2. The options in the dropdown list are in uppercase.  
3. The gender field is correctly positioned and labeled on the form.  
4. The label for the gender field is right-aligned and bold.  
5. The gender field is accessible within the "POLICY\_HOLDER" tab.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the gender field is not directly linked to a database item.

# Gender Selection for Appointee

Type: BENEFICIARIES

Title: Gender Selection for Appointee  
  
Acceptance Criteria:  
1. The gender selection should be available as a dropdown list with three options.  
2. The dropdown list should be positioned appropriately on the form.  
3. The selected gender should be saved and displayed correctly in the system.  
  
Definition of Done:  
1. The gender dropdown list is implemented and displays three options.  
2. The dropdown list is correctly positioned on the form.  
3. The selected gender is saved and can be retrieved accurately from the system.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include any specific database queries.

# Delete Nominee Functionality

Type: BENEFICIARIES

Title: Delete Nominee Functionality  
  
Acceptance Criteria:  
1. When the delete nominee button is pressed, the system should:  
 - Check if there are multiple nominees for the given application number.  
 - If multiple nominees are found, display an error message instructing the user to remove the nominee from the multiple nominee window.  
 - If no multiple nominees are found, proceed with the deletion process.  
2. If the policy holder is the same as the insured person, the system should display an error message if multiple nominees are found.  
3. If the deletion process is allowed to continue:  
 - If the control flag `cn\_wip\_continue` is set to 'T', the system should attempt to find the nominee in the extension table and delete the corresponding row.  
 - If the nominee is not found in the extension table, display a fatal error message.  
 - If the nominee is found, delete the row from the extension table and the current record.  
 - If the control flag `cn\_wip\_continue` is not set to 'T', simply delete the current record.  
  
Definition of Done:  
- The delete nominee functionality is implemented and tested.  
- The system correctly handles the presence of multiple nominees and displays appropriate error messages.  
- The system successfully deletes the nominee from both the main and extension tables when applicable.  
- All acceptance criteria are met and verified through testing.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Query to count multiple nominees:  
 ```sql  
 SELECT COUNT (1)  
 INTO v\_count  
 FROM azbj\_multiple\_nominee\_dtls  
 WHERE application\_no = TO\_CHAR(NVL(:insured\_person.ip\_sign\_card\_no, :insured\_person.ip\_verf\_no))  
 AND top\_indicator = 'Y';  
 ```  
  
- Query to find nominee in the extension table:  
 ```sql  
 dmi.find ('WIP\_AZBJ\_BENEFICIARY\_REP.BEN\_NO', :beneficiaries.bn\_rownum);  
 ```  
  
- Query to delete nominee from the extension table:  
 ```sql  
 dmi.delete\_row ('WIP\_AZBJ\_BENEFICIARY\_REP', azbj\_mn\_ext\_row);  
 ```  
  
- Query to delete the current record:  
 ```sql  
 DELETE\_RECORD;  
 ```

# Validate Surname Field in Beneficiaries Section

Type: BENEFICIARIES

Title: Validate Surname Field in Beneficiaries Section  
  
Acceptance Criteria:  
1. When the "Surname" field is validated, if the premium frequency is not null, the system should disable the "Commit Form" and "Exit Form" buttons.  
2. The system should ensure that the "Surname" field is always in uppercase.  
  
Definition of Done:  
1. The "Surname" field validation logic is implemented and tested.  
2. The "Commit Form" and "Exit Form" buttons are disabled when the premium frequency is not null.  
3. The "Surname" field is automatically converted to uppercase.  
4. All acceptance criteria are met and verified through testing.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Not applicable as the provided XML content does not include any direct database queries for CRUD operations.

# Validate Beneficiary Relationship and Navigate Based on Age and Product Type

Type: BENEFICIARIES

Title: Validate Beneficiary Relationship and Navigate Based on Age and Product Type  
  
Acceptance Criteria:  
1. When the user navigates to the next item, the system should calculate the entry age of the beneficiary using their date of birth and the policy's inception or effective date.  
2. If the beneficiary's age is less than 18, the system should navigate to the "Applicant Name" field.  
3. If the beneficiary's age is 18 or older and the product ID is one of the specified values (14, 31, 32, 33, 34, 49, 50) or matches certain product definitions, the system should navigate to the "Spouse Details" field.  
4. If none of the above conditions are met, the system should navigate to the "Sum Assured" field.  
5. When the item is validated, if the premium frequency is not null, the system should disable the "Commit Form" and "Exit Form" buttons.  
  
Definition of Done:  
- The system correctly calculates the entry age and navigates to the appropriate fields based on the beneficiary's age and product type.  
- The system disables the "Commit Form" and "Exit Form" buttons when the premium frequency is not null.  
- All specified conditions and validations are met without any errors.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- The system should use the following query to fetch the list of nominee details:  
 ```sql  
 SELECT CHAR\_VALUE FROM azbj\_system\_constants WHERE sys\_type='NOM\_DET';  
 ```

# Validate Beneficiary Relationship Input

Type: BENEFICIARIES

User Story: Validate Beneficiary Relationship Input  
  
Detailed Description:  
As a user, I want to ensure that the relationship input for beneficiaries is validated correctly so that the form behaves as expected based on the input provided.  
  
Acceptance Criteria:  
1. When the user inputs a relationship for a beneficiary, the system should check if the premium frequency is not null.  
2. If the premium frequency is not null, the system should disable the "Commit Form" and "Exit Form" buttons to prevent the user from submitting or exiting the form without completing the necessary steps.  
3. The system should automatically navigate to the next input field for the beneficiary's date of birth after the relationship input is validated.  
  
Definition of Done:  
- The system correctly validates the relationship input for beneficiaries.  
- The "Commit Form" and "Exit Form" buttons are disabled when the premium frequency is not null.  
- The form automatically navigates to the next input field for the beneficiary's date of birth after validation.  
- All acceptance criteria are met, and the functionality is tested and verified.  
  
DB Queries for Table Reference CRUD Operations:  
- Not applicable as the provided XML content does not include any direct database queries.  
  
Explanation of Oracle Form Logic:  
- The `WHEN-VALIDATE-ITEM` trigger checks if the premium frequency (`CV\_FREQ\_PREM`) is not null. If it is not null, it disables the "Commit Form" and "Exit Form" buttons to ensure the user completes the necessary steps before submitting or exiting the form.  
- The `KEY-NEXT-ITEM` trigger automatically navigates to the next input field for the beneficiary's date of birth (`BN\_APP\_DOB`) after the relationship input is validated.

# Validate Beneficiary's Date of Birth

Type: BENEFICIARIES

User Story: Validate Beneficiary's Date of Birth  
  
Detailed description: As a user, I want to ensure that the date of birth entered for a beneficiary is validated to confirm that the beneficiary is not a minor, and that certain form actions are disabled if specific conditions are met.  
  
Acceptance criteria:  
1. If the age of the beneficiary (calculated from the current date) is less than 18 years, an error message should be displayed stating, "The appointee cannot be a minor again."  
2. If a specific premium frequency field is not null, the following actions should be disabled:  
 - Committing the form  
 - Exiting the form  
  
Definition of Done:  
- The system correctly calculates the age of the beneficiary based on the date of birth entered.  
- An error message is displayed if the beneficiary is a minor.  
- The form actions (commit and exit) are disabled when the premium frequency field is not null.  
- The user is unable to proceed with the form submission if the beneficiary is a minor or if the premium frequency field is not null.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Not applicable as the provided XML content does not include direct database queries that can be executed independently of Oracle Forms constructs.

# Validate Beneficiary Name Input

Type: BENEFICIARIES

User Story: Validate Beneficiary Name Input  
  
Detailed Description:  
As a user, I want to ensure that the beneficiary name input field is validated correctly so that the data entered is accurate and meets the required conditions.  
  
Acceptance Criteria:  
1. When the beneficiary name is entered, it should automatically set the beneficiary share to 100.  
2. If the premium frequency is not null, the form should disable the commit and exit buttons to prevent form submission.  
3. The system should check the age of the insured person. If the insured person is a minor (under 18 years old), the system should:  
 - Display a message indicating that nomination is not allowed.  
 - Clear the date of birth, place of birth, and relationship fields for the beneficiary.  
 - Disable the input fields for the beneficiary's name, date of birth, place of birth, and relationship.  
4. If the insured person is not a minor, the system should enable the input fields for the beneficiary's name, date of birth, place of birth, and relationship.  
  
Definition of Done:  
- The beneficiary name input field is validated according to the specified conditions.  
- Appropriate messages and field states (enabled/disabled) are displayed based on the age of the insured person.  
- The commit and exit buttons are disabled when the premium frequency is not null.  
- The system correctly handles both minor and non-minor insured persons.  
  
DB Queries for Table Reference CRUD Operations:  
- Not applicable as the provided XML content does not include specific CRUD operations that can be executed directly in the database.  
  
Explanation of Oracle Form Logic:  
- The `WHEN-VALIDATE-ITEM` trigger sets the beneficiary share to 100 and disables the commit and exit buttons if the premium frequency is not null.  
- The `PRE-TEXT-ITEM` trigger checks the age of the insured person. If the insured person is a minor, it displays a message and disables the relevant input fields. If the insured person is not a minor, it enables the input fields.  
- The `KEY-NEXT-ITEM` trigger navigates to the next item and calls a procedure to enable or disable nominee fields based on certain conditions.

# Add Multiple Nominee Functionality

Type: BENEFICIARIES

Title: Add Multiple Nominee Functionality  
  
Acceptance Criteria:  
1. When the "Add Multiple Nominee" button is pressed, the system should:  
 - Check if there are existing nominees for the given application or proposal number.  
 - If nominees exist, fetch and display the details of the first nominee.  
 - Calculate the age of each nominee and identify if any nominee is a minor.  
 - If a minor nominee is found, prompt the user to enter appointee details.  
 - If no nominees exist, allow the user to add new nominee details.  
2. The system should ensure that the nominee details include name, birthplace, relation, date of birth, and gender.  
3. The system should handle any exceptions and display appropriate error messages.  
  
Definition of Done:  
- The "Add Multiple Nominee" button functionality is implemented and tested.  
- The system correctly fetches and displays existing nominee details.  
- The system prompts for appointee details if a minor nominee is found.  
- The system allows adding new nominee details if no existing nominees are found.  
- All exceptions are handled, and appropriate error messages are displayed.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Fetch existing nominees:  
 ```sql  
 SELECT   
 FROM azbj\_multiple\_nominee\_dtls   
 WHERE (application\_no = TO\_CHAR(NVL(:insured\_person.ip\_sign\_card\_no, :insured\_person.ip\_verf\_no))   
 OR proposal\_no = :control.cn\_policy\_ref)   
 AND top\_indicator = 'Y'   
 ORDER BY nominee\_number;  
 ```  
  
- Count existing nominees:  
 ```sql  
 SELECT COUNT(1)   
 INTO v\_nom\_count   
 FROM azbj\_multiple\_nominee\_dtls   
 WHERE (application\_no = TO\_CHAR(NVL(:insured\_person.ip\_sign\_card\_no, :insured\_person.ip\_verf\_no))   
 OR proposal\_no = :control.cn\_policy\_ref)   
 AND top\_indicator = 'Y';  
 ```  
  
- Fetch first nominee details:  
 ```sql  
 SELECT nominee\_name, nominee\_birthplace, nominee\_dob, nominee\_relation, nominee\_gender   
 INTO :beneficiaries.bn\_ben\_name, :beneficiaries.bn\_ben\_place\_of\_birth, :beneficiaries.bn\_ben\_dob, :beneficiaries.bn\_ben\_relation, :beneficiaries.nominee\_gender   
 FROM azbj\_multiple\_nominee\_dtls   
 WHERE (application\_no = TO\_CHAR(NVL(:insured\_person.ip\_sign\_card\_no, :insured\_person.ip\_verf\_no))   
 OR proposal\_no = :control.cn\_policy\_ref)   
 AND top\_indicator = 'Y'   
 AND nominee\_number = 1;  
 ```  
  
- Calculate nominee age:  
 ```sql  
 v\_nom\_age := azbj\_bk\_all\_validations.azbj\_calc\_entry\_age(i.nominee\_dob, pme\_api.opus\_date, 'P', 1);  
 ```  
  
- Check for minor nominee:  
 ```sql  
 IF v\_nom\_age < 18 THEN  
 v\_minor\_count := v\_minor\_count + 1;  
 :beneficiaries.bn\_ben\_name := i.nominee\_name;  
 :beneficiaries.bn\_ben\_place\_of\_birth := i.nominee\_birthplace;  
 :beneficiaries.bn\_ben\_dob := i.nominee\_dob;  
 :beneficiaries.bn\_ben\_relation := i.nominee\_relation;  
 :beneficiaries.nominee\_gender := i.nominee\_gender;  
 EXIT WHEN v\_minor\_count = 1;  
 END IF;  
 ```  
  
- Handle minor nominee:  
 ```sql  
 IF v\_minor\_count > 0 AND :beneficiaries.bn\_app\_name IS NULL THEN  
 azbj\_message('W', 'Minor Nominee(s) found.. Please enter the Appointee Details..');  
 go\_item('BENEFICIARIES.BN\_BEN\_DOB');  
 execute\_trigger('WHEN-VALIDATE-ITEM');  
 GO\_ITEM('beneficiaries.BN\_APP\_NAME');  
 END IF;  
 ```  
  
- Exception handling:  
 ```sql  
 EXCEPTION  
 WHEN OTHERS THEN  
 azbj\_message('W', 'exception in nominee' || SQLERRM);  
 ```

# Validate Appointee Name and Update Form Properties

Type: BENEFICIARIES

User Story: Validate Appointee Name and Update Form Properties  
  
Detailed Description:  
As a user, I want to ensure that the appointee name is validated and the form properties are updated accordingly when the appointee name is entered, so that the form behaves correctly based on the appointee's details.  
  
Acceptance Criteria:  
1. When the appointee name is entered, the system should check if the premium frequency is not null.  
2. If the premium frequency is not null, the system should disable the "Commit Form" and "Exit Form" buttons on the toolbar.  
3. The system should navigate to the next item, which is the appointee's relationship field.  
  
Definition of Done:  
- The appointee name field should be validated as per the specified conditions.  
- The "Commit Form" and "Exit Form" buttons should be disabled if the premium frequency is not null.  
- The form should navigate to the appointee's relationship field after the appointee name is entered.  
  
DB Queries for Table Reference CRUD Operations:  
- Not applicable as the provided XML content does not include any direct database queries.  
  
Explanation of Oracle Form Logic:  
- The `WHEN-VALIDATE-ITEM` trigger checks if the premium frequency is not null. If it is not null, it disables the "Commit Form" and "Exit Form" buttons on the toolbar.  
- The `KEY-NEXT-ITEM` trigger navigates to the next item, which is the appointee's relationship field.

# Validate Beneficiary Date of Birth and Appointee Information

Type: BENEFICIARIES

User Story: Validate Beneficiary Date of Birth and Appointee Information  
  
Detailed Description:  
As a user, I want to ensure that the date of birth entered for a beneficiary is validated against specific business rules so that the system can determine if an appointee is required and enable or disable related fields accordingly.  
  
Acceptance Criteria:  
1. When a date of birth is entered for a beneficiary, the system should calculate the entry age.  
2. If the entry age is 18 or older, the system should:  
 - Clear any existing appointee information (name, relation, date of birth).  
 - Display a warning message indicating that an appointee cannot be entered for a major.  
 - Disable the fields for appointee name, relation, date of birth, and gender.  
3. If the entry age is less than 18, the system should:  
 - Enable the fields for appointee name, relation, date of birth, and gender.  
4. If a premium frequency is specified, the system should disable the form's commit and exit buttons to prevent further changes until the form is saved.  
  
Definition of Done:  
- The system correctly calculates the entry age based on the beneficiary's date of birth and the policy's inception or effective date.  
- The system appropriately enables or disables the appointee-related fields based on the beneficiary's age.  
- The system displays the correct warning messages when necessary.  
- The form's commit and exit buttons are disabled when a premium frequency is specified, ensuring no further changes can be made until the form is saved.  
  
DB Queries for Table Reference CRUD Operations:  
- Not applicable as the provided XML content does not include specific CRUD operations that can be executed directly in the database without modification.  
  
Explanation of Oracle Form Logic:  
- The logic involves calculating the entry age of the beneficiary using the date of birth and comparing it with the policy's inception or effective date.  
- Based on the calculated age, the system determines whether the beneficiary is a minor or an adult.  
- If the beneficiary is an adult (18 or older), the system clears any existing appointee information and disables the related fields.  
- If the beneficiary is a minor (under 18), the system enables the fields for entering appointee information.  
- Additionally, if a premium frequency is specified, the system disables the form's commit and exit buttons to ensure data integrity until the form is saved.

# Manage Policy Holder Information

Type: POLICY\_HOLDER

Detailed description: As a user, I want to manage policy holder information efficiently, so that I can ensure all relevant details are captured and maintained accurately.  
  
Acceptance criteria:  
1. The system should allow the user to input and display the following policy holder details:  
 - Full Name  
 - National ID  
 - Partner Reference  
 - Sex  
 - Date of Birth  
 - Nationality  
 - Occupation  
 - Official Address  
 - Mailing Address  
 - Contact Information (Telephone, Fax, Email, Mobile)  
 - Relationship to the insured  
 - Insurance Amount  
 - Annual Income  
 - Age Proof  
 - Height and Weight  
 - Employer Information  
 - PAN Issuance Date  
 - GSTN Number  
 - Mother's Name and Title  
 - Spouse Name  
 - Father's Name  
 - Maiden Name  
 - Social Sector Status  
 - Information Sharing Consent  
  
2. The system should provide dropdown lists for fields such as Proposal Type, Age Proof, and Bank Name, populated from predefined lists.  
  
3. The system should include buttons for actions such as searching for CP, address, and other document details.  
  
4. The system should ensure that certain fields are read-only and cannot be modified by the user, such as Full Name, National ID, and Partner Reference.  
  
5. The system should validate the input data where applicable, such as ensuring the date format for Date of Birth and PAN Issuance Date, and numeric values for fields like Height, Weight, and Annual Income.  
  
6. The system should provide visual cues for mandatory fields and ensure that all required information is provided before allowing the user to proceed.  
  
Definition of Done:  
- The user interface for managing policy holder information is implemented and accessible.  
- All specified fields are available for input and display.  
- Dropdown lists are populated with data from the database.  
- Buttons for search and document actions are functional.  
- Read-only fields are enforced.  
- Data validation is implemented for applicable fields.  
- Visual cues for mandatory fields are present.  
- The system is tested and verified to ensure all acceptance criteria are met.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- For Proposal Type:  
 ```sql  
 SELECT display\_name, internal\_value FROM azbj\_proposal\_type;  
 ```  
  
- For PA Code:  
 ```sql  
 SELECT a.pa\_code, a.subcode, b.institution\_name, b.part\_id  
 FROM azbj\_partner\_extn a, cp\_partners b  
 WHERE a.part\_id = b.part\_id  
 AND b.partner\_type = 'I' AND a.pa\_code IS NOT NULL;  
 ```  
  
- For Bank Name:  
 ```sql  
 SELECT bank\_name FROM azbj\_bank\_master ORDER BY 1;  
 ```

# Input and Process Annual Income for Policy Holder

Type: POLICY\_HOLDER

Title: Input and Process Annual Income for Policy Holder  
  
Acceptance Criteria:  
1. When the annual income is entered, if the global loading flag is set to 'F', the form status should be updated to 'Y'.  
2. Upon pressing the key to navigate to the next item:  
 - If the product ID is not 11 and the product definition is not 'CHILDGAIN', the focus should move to the beneficiary's name field.  
 - Otherwise, the focus should move to the height field of the policyholder.  
  
Definition of Done:  
- The annual income field is available for input.  
- The form status updates correctly based on the global loading flag.  
- Navigation between fields works as specified based on the product ID and product definition.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include any direct database queries.

# Validate and Populate Telephone Number Field

Type: POLICY\_HOLDER

Title: Validate and Populate Telephone Number Field  
  
Acceptance Criteria:  
1. When the telephone number field is validated, if the global loading flag is set to 'F', the form status should be updated to 'Y'.  
2. When the telephone number field is double-clicked and it is empty, it should be automatically populated with the text 'Not Available'.  
  
Definition of Done:  
- The telephone number field should be validated according to the specified conditions.  
- The telephone number field should be populated with 'Not Available' when double-clicked if it is empty.  
- The changes should be tested and verified to ensure they work as expected.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include any direct database operations.

# Email Field Input and Validation for Policy Holder

Type: POLICY\_HOLDER

Title: Email Field Input and Validation for Policy Holder  
  
Acceptance Criteria:  
- The email field should be enabled and allow for text input.  
- The email field should have a maximum length of 80 characters.  
- The email field should be positioned correctly on the form.  
- The email field should be validated when the user inputs data.  
- If the form is not in a loading state, the form status should be updated to 'Y' upon validation of the email field.  
  
Definition of Done:  
- The email field is implemented and visible on the form.  
- Users can input and update the email address in the email field.  
- The email field is validated correctly.  
- The form status is updated to 'Y' when the email field is validated and the form is not in a loading state.  
- All acceptance criteria are met and tested.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the email field is not directly tied to a database item in this context.

# Implement and Validate Weight Field for Policy Holder

Type: POLICY\_HOLDER

Title: Implement and Validate Weight Field for Policy Holder  
  
Acceptance Criteria:  
1. The weight field should accept numeric values only, with a maximum length of 6 digits.  
2. The field should be positioned correctly on the form as per the design specifications.  
3. When the weight is entered, the system should validate the input.  
4. If the global loading flag is set to 'F', the form status should be updated to 'Y'.  
5. Upon pressing the key to move to the next item, the focus should shift to the 'Policy\_holder.ph\_wt\_chg' field.  
  
Definition of Done:  
1. The weight field is implemented and accepts numeric input only.  
2. The field is correctly positioned on the form.  
3. Validation logic is in place and functional.  
4. The form status updates correctly based on the global loading flag.  
5. Navigation to the next item 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 database operations.

# Input Field for Amount of Insurance on Policyholder

Type: POLICY\_HOLDER

Title: Input Field for Amount of Insurance on Policyholder  
  
Acceptance Criteria:  
1. The input field for the amount of insurance should accept numerical values only.  
2. The field should have a maximum length of 14 characters.  
3. The field should be positioned correctly on the user interface as specified.  
4. When the value in the field changes, the system should check if the form is not in a loading state and then update the form status to indicate that changes have been made.  
  
Definition of Done:  
1. The input field for the amount of insurance is implemented and visible on the user interface.  
2. The field accepts only numerical values and has a maximum length of 14 characters.  
3. The field is positioned correctly as per the design specifications.  
4. The form status is updated correctly when the value in the field changes, provided the form is not in a loading state.  
5. All acceptance criteria are met and verified through testing.

# User can select proposal type from LOV

Type: POLICY\_HOLDER

Title: User can select proposal type from LOV  
  
Acceptance Criteria:  
1. When the user double-clicks on the proposal type field, a list of values (LOV) should be displayed.  
2. The LOV should fetch and display the `display\_name` and `internal\_value` from the `azbj\_proposal\_type` table.  
3. The LOV should be auto-positioned and have auto column width.  
4. The proposal type field should be validated against the list of values.  
5. The field should be justified to the start, have a width of 80, and be positioned at coordinates (477, 227) on the screen.  
6. The field should have a white background, black foreground, and use the "MS Sans Serif" font with a size of 8.  
7. The field should be visible and have a prompt "of IP Proposal Type" with bold styling.  
  
Definition of Done:  
- The proposal type field displays an LOV when double-clicked.  
- The LOV correctly fetches and displays data from the `azbj\_proposal\_type` table.  
- The field is validated against the LOV.  
- The field and its prompt are styled and positioned as specified.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- `SELECT display\_name, internal\_value FROM azbj\_proposal\_type;` (Used to populate the LOV for proposal type selection)

# Height Field Input and Validation

Type: POLICY\_HOLDER

Title: Height Field Input and Validation  
  
Acceptance Criteria:  
1. The height field should accept numeric input only, with a maximum length of 6 digits.  
2. The field should be positioned correctly on the form and should be labeled "Height".  
3. When the height is entered, the system should validate the input. If the form is not in a loading state, the form status should be updated.  
4. After entering the height, the cursor should automatically move to the next field for weight input.  
  
Definition of Done:  
- The height field is implemented and accepts numeric input only.  
- The field is correctly labeled and positioned on the form.  
- The system performs validation on the height input and updates the form status if necessary.  
- The cursor moves to the next field for weight input after the height is entered.

# Validate Age Proof and Enable/Disable Fields Based on Age Proof

Type: POLICY\_HOLDER

Detailed description: As a user, I want to ensure that the age proof provided by the policyholder is validated correctly and that the appropriate fields are enabled or disabled based on the type of age proof selected.  
  
Acceptance criteria:  
1. If the age proof provided by the insured person is 'SYBM' and the agent code does not start with '522', an error message should be displayed stating that the Syndicate Bank BM Certificate is allowed only for Syndicate Bank.  
2. When the age proof is changed, the system should:  
 - Retrieve the corresponding AGE\_UID from the `azbj\_Age\_proof\_UID` table and store it in the `PH\_AGE\_PRF\_MSG` field.  
 - If no data is found, set the `PH\_AGE\_PRF\_MSG` field to 'NA1'.  
 - If any other error occurs, set the `PH\_AGE\_PRF\_MSG` field to 'NA2'.  
3. If the age proof is 'PC', the fields `PH\_PAN\_ISSUANCE\_DATE` and `PH\_PAN\_ISSUANCE\_DATE\_NA` should be enabled.  
4. If the age proof is not 'PC', the fields `PH\_PAN\_ISSUANCE\_DATE` and `PH\_PAN\_ISSUANCE\_DATE\_NA` should be disabled.  
  
Definition of Done:  
- The system correctly validates the age proof and displays appropriate error messages.  
- The system retrieves and sets the `PH\_AGE\_PRF\_MSG` field based on the age proof selected.  
- The fields `PH\_PAN\_ISSUANCE\_DATE` and `PH\_PAN\_ISSUANCE\_DATE\_NA` are enabled or disabled based on the age proof selected.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Retrieve AGE\_UID based on age proof:  
 ```sql  
 SELECT AGE\_UID  
 INTO :POLICY\_HOLDER.PH\_AGE\_PRF\_MSG  
 FROM azbj\_Age\_proof\_UID  
 WHERE age\_proof = :POLICY\_HOLDER.PH\_AGE\_PROOF;  
 ```  
  
- Handle exceptions:  
 ```sql  
 EXCEPTION  
 WHEN no\_data\_found THEN  
 :POLICY\_HOLDER.PH\_AGE\_PRF\_MSG := 'NA1';  
 WHEN others THEN  
 :POLICY\_HOLDER.PH\_AGE\_PRF\_MSG := 'NA2';  
 ```

# Dynamic Field Enable/Disable Based on Policy Holder Relationship

Type: POLICY\_HOLDER

Title: Dynamic Field Enable/Disable Based on Policy Holder Relationship  
  
Acceptance Criteria:  
1. When the relationship status of the policy holder is set to 'Husband':  
 - The insurance field for the policy holder should be disabled.  
 - The annual income field for the policy holder should be disabled.  
2. When the relationship status of the policy holder is set to any value other than 'Husband':  
 - The insurance field for the policy holder should be enabled.  
 - The annual income field for the policy holder should be enabled.  
3. When navigating to the next item:  
 - Calculate the age of the insured person based on their date of birth and the inception date or effective date.  
 - If the calculated age is less than 18:  
 - Set the beneficiary relationship to match the policy holder's relationship.  
 - Disable the beneficiary relationship field.  
 - If the calculated age is 18 or older:  
 - Enable the beneficiary relationship field.  
  
Definition of Done:  
- The form dynamically enables or disables the insurance and annual income fields based on the policy holder's relationship status.  
- The form correctly calculates the age of the insured person and adjusts the beneficiary relationship field accordingly.  
- All changes are tested and verified to ensure they meet the acceptance criteria.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include any direct database queries.

# Occupation Status Field for Policy Holder

Type: POLICY\_HOLDER

Title: Occupation Status Field for Policy Holder  
  
Acceptance Criteria:  
1. The occupation status field should be displayed on the "Policy Holder" tab.  
2. The field should be a dropdown list containing predefined occupation statuses.  
3. The dropdown list should be non-editable and should not allow new entries.  
4. The field should be positioned at the specified location on the form.  
5. The field should be visually distinct with a gray background and bold prompt text.  
6. The field should not allow insertion or updates directly by the user.  
  
Definition of Done:  
1. The occupation status field is visible on the "Policy Holder" tab.  
2. The dropdown list contains the predefined occupation statuses.  
3. The field is non-editable and does not allow new entries.  
4. The field is correctly positioned and styled as specified.  
5. The field does not allow direct insertion or updates by the user.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the field is non-editable and does not directly interact with the database for CRUD operations.

# Validation for Changed Weight Field

Type: POLICY\_HOLDER

Title: Validation for Changed Weight Field  
  
Acceptance Criteria:  
1. If the "Changed Weight" field is left empty, it should default to 0.  
2. If the system is not in loading mode, the form status should be set to 'Y'.  
3. The system should display an error message if the following conditions are met:  
 - The "Changed Weight" is greater than or equal to the current weight.  
 - The partner type is not 'I'.  
 - The product ID is not 169.  
  
Definition of Done:  
- The "Changed Weight" field defaults to 0 when left empty.  
- The form status is updated to 'Y' when the system is not in loading mode.  
- An error message is displayed when the "Changed Weight" is greater than or equal to the current weight, the partner type is not 'I', and the product ID is not 169.  
- The user can navigate to the "Beneficiaries" section after entering the "Changed Weight".  
  
DB queries for Table reference CRUD operations only(With Usage):  
- No specific DB queries provided in the XML content.

# Auto-populate and clear policy holder details based on 'Same as Insured' checkbox

Type: POLICY\_HOLDER

Detailed description: As a user, I want the system to automatically populate the policy holder's details with the insured person's details when the 'Same as Insured' checkbox is checked, and clear the policy holder's details when the checkbox is unchecked. This will ensure that the policy holder's information is consistent with the insured person's information when applicable.  
  
Acceptance criteria:  
1. When the 'Same as Insured' checkbox is checked:  
 - If no insured person is selected, display an error message prompting the user to select an insured person.  
 - Populate the policy holder's details with the insured person's details.  
 - Disable the fields related to the policy holder's details to prevent manual editing.  
 - Set the policy holder's annual income to the insured person's annual income and disable the annual income field.  
 - Disable other related fields such as contact email, insurance, mailing address, age proof, and relation.  
 - Update the policy holder's name in the SOFA information data.  
 - Navigate to the beneficiaries' name field if it is enabled.  
  
2. When the 'Same as Insured' checkbox is unchecked:  
 - Clear the policy holder's details.  
 - Enable the fields related to the policy holder's details for manual editing.  
 - Clear the policy holder's annual income and enable the annual income field.  
 - Enable other related fields such as contact email, insurance, mailing address, age proof, and relation.  
 - Clear the policy holder's name in the SOFA information data.  
 - Navigate to the policy holder's contact search field.  
  
3. Additional conditions:  
 - If the checkbox is unchecked, ensure that the PAN issuance date fields are enabled.  
 - If the checkbox is checked, ensure that the PAN issuance date fields are cleared and disabled.  
 - If the checkbox is checked, ensure that the Aadhar field is visible and populated with the insured person's Aadhar number.  
 - If the checkbox is unchecked, store the beneficiaries' details in temporary variables.  
 - If the checkbox is checked, populate the beneficiaries' details from the temporary variables.  
  
Definition of Done:  
- The system correctly populates and clears the policy holder's details based on the state of the 'Same as Insured' checkbox.  
- All related fields are enabled or disabled as per the acceptance criteria.  
- Error messages are displayed appropriately when no insured person is selected.  
- The policy holder's name in the SOFA information data is updated correctly.  
- Navigation between fields works as expected based on the checkbox state.  
- All additional conditions are met, including handling of PAN issuance date fields and Aadhar field visibility.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Not applicable as the provided XML content does not include specific CRUD operations directly executable in the database.

# Account Number Validation for Credit Card Payment

Type: POLICY\_HOLDER

Title: Account Number Validation for Credit Card Payment  
  
Acceptance Criteria:  
1. When the method of payment for a policy holder is 'CCSI' (Credit Card), the system should validate the account number.  
2. The account number must be exactly 16 digits long.  
3. If the account number is not 16 digits long, an error message should be displayed stating, "Credit card number must be 16 digits."  
  
Definition of Done:  
1. The account number validation logic is implemented and tested.  
2. The error message is displayed correctly when the account number is not 16 digits long.  
3. The validation only triggers when the method of payment is 'CCSI'.  
4. All unit tests and integration tests pass.  
5. The feature is reviewed and approved by the stakeholders.

# Automatic Navigation Based on Payment Method

Type: POLICY\_HOLDER

Title: Automatic Navigation Based on Payment Method  
  
Acceptance Criteria:  
1. If the payment method is 'CCSI', the system should automatically navigate to the 'Expiration Date' field after entering the bank city.  
2. If the payment method is not 'CCSI', the system should automatically navigate to the 'Close' button after entering the bank city.  
  
Definition of Done:  
1. The system correctly identifies the payment method.  
2. The system navigates to the 'Expiration Date' field if the payment method is 'CCSI'.  
3. The system navigates to the 'Close' button if the payment method is not 'CCSI'.  
4. The functionality is tested and verified to work as expected in different scenarios.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include any specific database queries.

# PA Code Selection and Validation for Policy Holder

Type: POLICY\_HOLDER

Title: PA Code Selection and Validation for Policy Holder  
  
Acceptance Criteria:  
1. The PA Code field should allow the user to select a value from a predefined list.  
2. When a PA Code is entered, it should be validated against the database to ensure it corresponds to an existing partner with the type 'I'.  
3. If the PA Code is invalid, an error message should be displayed to the user indicating that the code is invalid and prompting them to select a valid code from the list.  
  
Definition of Done:  
1. The PA Code field is implemented and allows selection from a predefined list.  
2. The validation logic for the PA Code is implemented and correctly identifies valid and invalid codes.  
3. An error message is displayed for invalid PA Codes, guiding the user to select a valid code.  
4. The functionality is tested and verified to work as expected.  
  
DB queries for Table reference CRUD operations only (With Usage):  
```sql  
SELECT a.pa\_code, a.subcode, b.institution\_name, b.part\_id  
FROM azbj\_partner\_extn a, cp\_partners b  
WHERE a.part\_id = b.part\_id  
AND b.partner\_type = 'I'  
AND a.pa\_code IS NOT NULL;  
```  
- This query is used to populate the list of valid PA Codes for selection.

# Validate and Update Secondary Contact Telephone Number

Type: POLICY\_HOLDER

Detailed description: As a user, I want to ensure that the secondary contact telephone number (Telephone2) for a policy holder is validated and updated correctly, so that the information is accurate and complete.  
  
Acceptance criteria:  
1. When the Telephone2 field is validated, if the system is not in a loading state, the form status should be set to 'Y'.  
2. When the Telephone2 field is double-clicked, if the primary contact telephone number is empty, it should be automatically set to 'Not Available'.  
  
Definition of Done:  
- The Telephone2 field should be enabled and allow updates.  
- The form status should be updated correctly based on the loading state.  
- The primary contact telephone number should be set to 'Not Available' if it is empty when the Telephone2 field is double-clicked.  
- The user interface should reflect these changes without any errors.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include any direct database queries.

# Input Expiry Date for Policy Holder

Type: POLICY\_HOLDER

Title: Input Expiry Date for Policy Holder  
  
Acceptance Criteria:  
1. The expiry date field should accept dates in the format "mm/yyyy".  
2. Upon entering the expiry date, the system should automatically navigate to the next field, which is the card type.  
  
Definition of Done:  
1. The expiry date field is visible and accepts input in the specified format.  
2. The system successfully navigates to the card type field after the expiry date is entered.  
3. The expiry date field is properly aligned and styled as per the design specifications.  
4. The functionality is tested and verified to ensure it meets the acceptance criteria.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include any specific database queries.

# Card Type Selection for Policy Holder

Type: POLICY\_HOLDER

Title: Card Type Selection for Policy Holder  
  
Acceptance Criteria:  
- The card type field should display a list of three predefined card types.  
- Upon selecting a card type and pressing the key to move to the next item, the focus should automatically move to the card holder field.  
  
Definition of Done:  
- The card type field is implemented and displays the predefined list of card types.  
- The navigation from the card type field to the card holder field works as expected.  
- The functionality is tested and verified to ensure it meets the acceptance criteria.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include any specific database queries.

# Close Renewal Premium Details Window and Navigate to Method of Payment

Type: POLICY\_HOLDER

Detailed description: As a user, I want to be able to close the "Renewal Premium Details" window and navigate to the "Method of Payment" field within the "Policy Holder" section, so that I can efficiently manage policy holder details.  
  
Acceptance criteria:  
1. When the "Close" button is pressed, the "Renewal Premium Details" window should be hidden.  
2. After the window is hidden, the focus should automatically move to the "Method of Payment" field within the "Policy Holder" section.  
  
Definition of Done:  
1. The "Close" button successfully hides the "Renewal Premium Details" window.  
2. The focus is correctly set to the "Method of Payment" field in the "Policy Holder" section after the window is hidden.  
3. 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 CRUD operations mentioned in the provided XML content.

# Input Credit Card Holder's Name

Type: POLICY\_HOLDER

Title: Input Credit Card Holder's Name  
  
Acceptance Criteria:  
1. The input field for the credit card holder's name should be visible and enabled.  
2. The input field should accept a maximum of 250 characters.  
3. The input should be automatically converted to uppercase.  
4. Upon pressing the "Next" key, the focus should move to the next relevant field, which is the credit card relation field.  
  
Definition of Done:  
1. The input field for the credit card holder's name is implemented and visible in the policy holder section.  
2. The input field accepts up to 250 characters and converts all input to uppercase.  
3. The focus moves to the credit card relation field upon pressing the "Next" key.  
4. All acceptance criteria are met and tested 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.

# Implement Relationship Dropdown for Policy Holder

Type: POLICY\_HOLDER

Title: Implement Relationship Dropdown for Policy Holder  
  
Acceptance Criteria:  
1. The relationship field should be a dropdown list containing four predefined options.  
2. The dropdown list should be positioned at the specified coordinates on the screen.  
3. The field should have a maximum length of 30 characters.  
4. The field should be labeled "Reln with PH" and should be bold.  
5. The field should be initially hidden and only become visible under certain conditions.  
6. The field should be part of a canvas that is used for displaying multiple tabs with information.  
7. The canvas should be hidden initially and should have specific dimensions and properties.  
  
Definition of Done:  
1. The dropdown list for the relationship of the policy holder is implemented and contains the correct options.  
2. The field is correctly positioned and labeled on the screen.  
3. The field adheres to the specified maximum length and visibility conditions.  
4. The canvas containing the field is correctly configured and hidden initially.  
5. All acceptance criteria are met and verified through testing.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include any specific database queries.

# Edit Policy Holder Payment Details

Type: POLICY\_HOLDER

Title: Edit Policy Holder Payment Details  
  
Acceptance Criteria:  
1. If the policy holder's method of payment is 'DD' (Direct Debit), the following fields should be enabled and navigable:  
 - Account Number  
 - Bank Name  
 - Bank Branch  
 - Bank City  
2. The 'Account Number' field should display the prompt text 'Account No.' and be marked as required.  
3. The 'Expiration Date', 'Card Type', 'Credit Card Holder', and 'Credit Card Relation to Policy Holder' fields should be hidden.  
4. The 'Bank Name' field should be pre-filled with 'Standard Chartered Bank' and then disabled.  
5. The 'Expiration Date', 'Card Type', 'Credit Card Holder', and 'Credit Card Relation to Policy Holder' fields should be cleared.  
6. The focus should move to the 'Account Number' field if it is enabled; otherwise, it should move to the 'Close' button.  
  
Definition of Done:  
- The user can successfully edit the policy holder's payment details based on the specified conditions.  
- The specified fields are enabled, disabled, visible, or hidden as per the conditions.  
- The focus moves to the appropriate field based on the conditions.  
- The changes are saved and reflected in the system.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Not applicable as the provided XML content does not include any direct database queries.

# Spouse Name Field Functionality

Type: POLICY\_HOLDER

Title: Spouse Name Field Functionality  
  
Acceptance Criteria:  
1. The spouse's name field should be displayed on the policy holder section.  
2. The field should not allow any insertions or updates.  
3. The maximum length of the spouse's name should be 100 characters.  
4. The field should be positioned correctly on the form as per the design specifications.  
5. When the spouse's name is changed, if the global loading flag is set to 'F', the form status should be updated to 'Y'.  
6. Navigation from the spouse's name field should depend on the product ID:  
 - If the product ID is not 11 and the product definition is not 'CHILDGAIN', the focus should move to the beneficiary's name field.  
 - Otherwise, the focus should move to the height field.  
  
Definition of Done:  
1. The spouse's name field is visible and correctly positioned on the policy holder section.  
2. The field does not allow any insertions or updates.  
3. The maximum length of the field is set to 100 characters.  
4. The form status is updated correctly when the spouse's name is changed, based on the global loading flag.  
5. Navigation from the spouse's name field works as specified based on the product ID and product definition.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include any specific database queries.

# Capture and Validate Father's Name for Policy Holder

Type: POLICY\_HOLDER

Title: Capture and Validate Father's Name for Policy Holder  
  
Acceptance Criteria:  
1. The father's name field should be displayed on the "Policy Holder" tab.  
2. The field should have a maximum length of 100 characters.  
3. The field should not allow insertion or updates.  
4. Upon changing the value in the father's name field:  
 - If the form is not in a loading state, the form status should be set to 'Y'.  
5. Upon pressing the key to navigate to the next item:  
 - If the product ID is not 11 and the product definition is not 'CHILDGAIN', the focus should move to the beneficiary's name field.  
 - Otherwise, the focus should move to the height field.  
  
Definition of Done:  
1. The father's name field is visible and correctly positioned on the "Policy Holder" tab.  
2. The field adheres to the specified constraints (maximum length, no insertion, no updates).  
3. The form status logic is correctly implemented and tested.  
4. The navigation logic based on product ID and product definition is correctly implemented and tested.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include specific CRUD operations or table references.

# Validate PAN Issuance Date for Policy Holder

Type: POLICY\_HOLDER

Title: Validate PAN Issuance Date for Policy Holder  
  
Acceptance Criteria:  
- The system should validate the PAN Issuance Date when it is entered.  
- If the PAN Issuance Date is greater than the current system date, an error message should be displayed: "PAN Issuance Date should not be less than system date."  
  
Definition of Done:  
- The validation logic is implemented and tested.  
- The error message is displayed correctly when the PAN Issuance Date is a future date.  
- The user is prevented from saving the record if the PAN Issuance Date is invalid.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include any direct database operations.

# Search Mailing Address for Policy Holder

Type: POLICY\_HOLDER

Title: Search Mailing Address for Policy Holder  
  
Acceptance Criteria:  
1. If no policy holder is selected, an error message should be displayed indicating that a partner must be selected first.  
2. The system should set a global variable with the policy holder's partner ID.  
3. The system should call another form to search for the mailing address.  
4. If no mailing address is selected, an error message should be displayed prompting the user to select a mailing address.  
5. The system should retrieve and display the mailing address details, including address lines, country, postcode, and contact numbers, from the database.  
  
Definition of Done:  
- The user can initiate a search for a mailing address by pressing a button.  
- Appropriate error messages are displayed if prerequisites are not met.  
- The system successfully retrieves and displays the mailing address details if a valid address is selected.  
- The functionality is tested and verified to work as expected.  
  
DB queries for Table reference CRUD operations only(With Usage):  
```sql  
SELECT azbj\_pk\_policy\_schedule1.get\_son\_of\_wife\_of(:POLICY\_HOLDER.PH\_PART\_ID)||CHR(10) ||  
 ADDRESS\_LINE1 || CHR(10) || ADDRESS\_LINE2 || CHR(10) ||   
 tae.AREA\_FREE\_TEXT\_M || CASE WHEN tae.AREA\_FREE\_TEXT\_M IS NOT NULL THEN CHR(10) ELSE NULL END ||  
 ADDRESS\_LINE3 || CHR(10) || ADDRESS\_LINE4 || CHR(10) || ADDRESS\_LINE5 || CHR(10) || ADDRESS\_LINE6 || CHR(10) ||  
 ADDRESS\_LINE7 || 'COUNTRY- '||UPPER(DESCRIPTION)||' '||postcode, CPA.COUNTRY\_CODE, postcode,  
 TELEPHONE2, TELEPHONE, MOBILE\_NO  
INTO :POLICY\_HOLDER.PH\_MAIL\_ADDR, :policy\_holder.PH\_MAILING\_COUNTRY\_CODE, :policy\_holder.PH\_MAILING\_POSTCODE,  
 :policy\_holder.PH\_CONTACT\_TELNO2, :policy\_holder.PH\_CONTACT\_TELNO, :policy\_holder.PH\_CONTACT\_MOBILE  
FROM CP\_ADDRESSES CPA, AZBJ\_ADDRESS\_EXTN TAE, CP\_COUNTRIES CP  
WHERE CPA.ADD\_ID = :PH\_MAIL\_ID  
AND TAE.ADD\_ID = CPA.ADD\_ID  
AND CP.COUNTRY\_CODE = CPA.COUNTRY\_CODE;  
```

# Manage PAN Issuance Date for Policy Holders

Type: POLICY\_HOLDER

Title: Manage PAN Issuance Date for Policy Holders  
  
Acceptance Criteria:  
1. When the checkbox for "PAN Issuance Date Not Available" is checked:  
 - The PAN issuance date field should be disabled.  
 - The PAN issuance date field should be cleared.  
 - A function should be called to save the state indicating that the PAN issuance date is not available.  
2. When the checkbox for "PAN Issuance Date Not Available" is unchecked:  
 - The PAN issuance date field should be enabled.  
 - If the PAN issuance date field is not empty, a function should be called to save the state indicating that the PAN issuance date is available.  
  
Definition of Done:  
- The checkbox for "PAN Issuance Date Not Available" should correctly enable or disable the PAN issuance date field based on its state.  
- The PAN issuance date field should be cleared when the checkbox is checked.  
- The appropriate function should be called to save the state based on the checkbox's state.  
- The feature should be 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 direct database queries.

# Manage Probable Policy Holder Details

Type: POLICY\_HOLDER

Title: Manage Probable Policy Holder Details  
  
Acceptance Criteria:  
1. When the "Probable CP" button is pressed, the system should display the "Probable CP PH" view and navigate to the "Probable CP PH" block.  
2. If the variable `pk\_vars.v\_prob\_ip\_sel` is set to 'N', the system should clear the block.  
3. The system should iterate through records in the `azbj\_auto\_cp\_merge\_details` table where the application number matches the insured person's verification or sign card number, and the partner type is 'PH' and top indicator is 'Y'.  
4. For each partner record, the system should:  
 - Populate the fields in the "Probable CP PH" block with the partner's details, including partner ID, policy reference, name, annual premium, policy status, and sum insured.  
 - Retrieve and populate the `tasa` and `suc` values from the `azbj\_ip\_ext` or `wip\_azbj\_ip\_ext` tables based on the contract ID and IP number.  
 - Retrieve and populate the percentage values (`ml\_perc`, `oc\_perc`, `nri\_perc`, `sr\_perc`) from the `azbj\_policy\_covers\_ext` or `wip\_azbj\_policy\_covers\_ext` tables based on the contract ID and cover code.  
 - Determine if the policy is rated up based on the percentage values and set the `RATED\_UP` field accordingly.  
 - Retrieve and populate the package code from the `wip\_azbj\_policy\_bases\_ext` or `azbj\_policy\_bases\_ext` tables based on the contract ID.  
 - Retrieve and populate the product ID using the `azbj\_pk0\_acc.get\_product\_id` function based on the contract ID.  
5. The system should handle exceptions and display an error message if any issues occur during the data retrieval process.  
  
Definition of Done:  
- The "Probable CP PH" view is displayed correctly when the button is pressed.  
- The "Probable CP PH" block is populated with accurate data from the relevant tables.  
- The system handles exceptions gracefully and displays appropriate error messages.  
- The user can navigate through the records and view all relevant details for each probable policy holder.  
- The functionality is tested and verified to ensure data accuracy and system stability.  
  
DB queries for Table reference CRUD operations only(With Usage):  
```sql  
-- Retrieve partner details  
SELECT unique part\_id  
FROM azbj\_auto\_cp\_merge\_details   
WHERE appln\_no = TO\_CHAR (NVL (:insured\_person.ip\_verf\_no, :insured\_person.ip\_sign\_card\_no))  
AND partner\_type = 'PH'  
AND top\_indicator = 'Y';  
  
-- Retrieve partner contract details  
SELECT partner\_id, a.contract\_id, f.booking\_frequency, azbj\_pk0\_acc.get\_policy\_ref (a.contract\_id) policy\_ref,   
 e.sum\_insured\_whole\_cover, e.cover\_code, b.before\_title ||' '||b.first\_name ||' '||b.surname name,   
 g.ann\_prem, c.contract\_status status, c.product\_id, e.FTPREMIUM\_OR\_WHOLE\_COVER, g.entry\_age,   
 g.premium\_term, g.benefit\_term, g.col6 interest\_rate, g.freq\_std\_prem, g.extra\_amt, g.ml\_perc,   
 g.oc\_perc, g.nri\_perc, g.sr\_perc, g.prem\_disc\_amt  
FROM ocp\_interested\_parties a, cp\_partners b, ocp\_policy\_versions c, ocp\_policy\_covers e,   
 ocp\_policy\_bases f, azbj\_policy\_covers\_ext g, azbj\_policy\_contract\_ext h  
WHERE a.partner\_id = cur.part\_id  
AND a.partner\_id = b.part\_id   
AND a.contract\_id = c.contract\_id  
AND a.contract\_id = e.contract\_id   
AND e.cover\_code LIKE 'L%'  
AND f.contract\_id = c.contract\_id  
AND a.contract\_id = f.contract\_id  
AND g.contract\_id = c.contract\_id  
AND a.contract\_id = g.contract\_id  
AND h.contract\_id = c.contract\_id  
AND a.contract\_id = h.contract\_id;  
  
-- Retrieve tasa and suc values  
SELECT tasa\_value, bank\_ac\_no, col7  
INTO :BLK\_PROBABLE\_CP\_PH.tasa, :BLK\_PROBABLE\_CP\_PH.suc, v\_cover\_type  
FROM azbj\_ip\_ext  
WHERE contract\_id = cur\_partner.contract\_id  
AND ip\_no = 2;  
  
-- Retrieve percentage values  
SELECT nvl(ml\_perc,0), nvl(oc\_perc,0), nvl(nri\_perc,0), nvl(sr\_perc,0)  
INTO v\_ml\_perc, v\_oc\_perc, v\_nri\_perc, v\_sr\_perc  
FROM azbj\_policy\_covers\_ext  
WHERE contract\_id = cur\_partner.contract\_id  
AND (ml\_perc > 0 OR oc\_perc > 0 OR nri\_perc > 0 OR sr\_perc > 0)  
AND cover\_code LIKE 'L%'  
AND top\_indicator = 'Y'  
AND action\_code <> 'D';  
  
-- Retrieve package code  
SELECT PACKAGE\_CODE  
INTO v\_pack\_codee  
FROM wip\_azbj\_policy\_bases\_ext  
WHERE contract\_id = cur\_partner.contract\_id  
AND top\_indicator = 'Y'  
AND action\_code <> 'D';  
```

# Validate and Process Method of Payment for Policy Holders

Type: POLICY\_HOLDER

Title: Validate and Process Method of Payment for Policy Holders  
  
Acceptance Criteria:  
1. If the method of payment is 'Direct Debit' (DD) and the product definition is not in the specified list ('SUPER\_CASHGAIN', 'CASH\_RICH', 'YOUNG\_ASSURE', 'ELITE\_ASSURE'), and the agent code does not meet certain criteria, the system should:  
 - Set the method of payment to null.  
 - Display a message indicating that Direct Debit is available only for specific cases.  
2. If the method of payment is one of the specified types ('CCSI', 'ADI', 'DD', 'ECS', 'BJECS', 'NACH', 'EMAND'), the system should:  
 - Create a parameter list and add parameters related to application number, payment mode, and module.  
 - Call a form to handle the renewal payment mode.  
 - Retrieve and populate bank account details from a specific table based on the application number.  
3. If the method of payment is 'Direct Debit' (DD), the system should:  
 - Enable and make navigable the fields for account number, bank name, bank branch, and bank city.  
 - Set the bank name to 'Standard Chartered Bank' and disable the bank name field.  
 - Hide and disable fields related to credit card details.  
4. If the method of payment is not 'Direct Debit' (DD), the system should:  
 - Clear the fields for account number, bank name, bank branch, bank city, expiration date, card type, card holder, and card holder relation.  
 - Disable the fields for account number, bank name, bank branch, and bank city.  
 - Hide the fields for expiration date, card type, card holder, and card holder relation.  
 - Navigate to the appropriate item based on the value of a control flag.  
  
Definition of Done:  
- The system correctly validates and processes the method of payment based on the specified conditions.  
- Appropriate fields are enabled, disabled, or cleared as per the payment method selected.  
- Relevant messages are displayed to the user when conditions are not met.  
- The system navigates to the correct item based on the control flag value.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Retrieve bank account details from `azbj\_ren\_paymode\_reg\_qc\_bbu` table based on the application number and payment method.  
- Retrieve verification status from `azbj\_proposal\_appln\_det\_ext` table based on the application number.

# Consent Information Field for Policy Holder

Type: POLICY\_HOLDER

Title: Consent Information Field for Policy Holder  
  
Acceptance Criteria:  
1. The consent information field should be visible on the policy holder's tab.  
2. The field should allow a maximum of 50 characters.  
3. The field should have a dropdown list with three options for consent status.  
4. The field should be labeled "Information Sharing Consent".  
5. The field should be displayed with a white background and black text.  
6. The consent information should be stored and retrieved accurately.  
  
Definition of Done:  
1. The consent information field is implemented and visible on the policy holder's tab.  
2. The field allows input up to 50 characters.  
3. The dropdown list contains three predefined options.  
4. The field is labeled correctly as "Information Sharing Consent".  
5. The field's appearance matches the specified design (white background, black text).  
6. The consent information is correctly stored and retrieved from the database.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include specific database operations.

# Validate Age Proof ID and Trigger Appropriate Actions

Type: POLICY\_HOLDER

Title: Validate Age Proof ID and Trigger Appropriate Actions  
  
Acceptance Criteria:  
1. If the age proof type is 'AC' or 'ACS' and the length of the age proof ID is 12 characters, the system should:  
 - Extract the first name and last name from the policy holder's name.  
 - Format the date of birth of the policy holder.  
 - Validate the Aadhaar details using the provided information.  
 - Update the age proof ID with the validated Aadhaar details.  
 - Log the validation process for debugging purposes.  
2. If the age proof type is 'SYBM' and the agent code does not start with '522', the system should display an error message indicating that the Syndicate Bank BM Certificate is only allowed for Syndicate Bank.  
3. If the age proof type is 'Pan Card' and the age proof ID is not null, the system should:  
 - Update the agent's PAN card details with the provided age proof ID.  
 - Set the PAN card verification status to 'Y'.  
 - Trigger the validation process for the PAN card.  
 - Navigate to the PAN issuance date field.  
  
Definition of Done:  
- The system correctly validates the age proof ID based on the specified criteria.  
- Appropriate actions are triggered based on the type of age proof provided.  
- Error messages are displayed when validation criteria are not met.  
- The validation process is logged for debugging purposes.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include any direct database queries that can be executed independently of Oracle Forms constructs.

# Implement Income Proof Field for Policy Holder

Type: POLICY\_HOLDER

Title: Implement Income Proof Field for Policy Holder  
  
Acceptance Criteria:  
1. The income proof field should be a list item allowing users to select from predefined options.  
2. The field should be clearly labeled as "PH income proof".  
3. The field should be positioned appropriately within the policy holder section for easy access.  
4. The field should have a maximum length of 100 characters.  
5. The field should be visually distinct with appropriate font and color settings to ensure readability.  
  
Definition of Done:  
- The income proof field is implemented as a list item within the policy holder section.  
- The field is labeled "PH income proof" and is positioned correctly.  
- The field adheres to the specified visual and functional requirements.  
- The field is tested and verified to ensure it meets the acceptance criteria.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include specific database queries.

# Display and Select Title for Policy Holder's Mother

Type: POLICY\_HOLDER

Title: Display and Select Title for Policy Holder's Mother  
  
Acceptance Criteria:  
1. The title field for the policy holder's mother should be displayed as a dropdown list.  
2. The dropdown list should be initialized with the value "Mrs".  
3. The title field should be read-only and not allow any updates or insertions.  
4. The title field should be visible and positioned correctly within the form.  
5. The title field should have a maximum length of 5 characters.  
6. The title field should be queryable, allowing users to search for records based on this field.  
  
Definition of Done:  
1. The title field for the policy holder's mother is displayed as a dropdown list with the initial value "Mrs".  
2. The field is read-only and does not allow updates or insertions.  
3. The field is visible and correctly positioned within the form.  
4. The field has a maximum length of 5 characters.  
5. The field is queryable, allowing users to search for records based on this field.  
  
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.

# Capture Mother's First Name for Policyholder

Type: POLICY\_HOLDER

Title: Capture Mother's First Name for Policyholder  
  
Acceptance Criteria:  
1. The field for the mother's first name should be disabled for editing.  
2. The field should have a maximum length of 50 characters.  
3. The field should be positioned at coordinates (82, 51) on the form.  
4. The field should be labeled "Mother's First Name" with a font size of 8, font name "Tahoma", and bold style.  
5. The field should not allow insertion or updates.  
6. Upon changing the value in the field, if the global loading flag is set to 'F', the form status should be updated to 'Y'.  
7. When navigating to the next item, if the product ID is not 11 and the product definition is not 'CHILDGAIN', the focus should move to the beneficiary's name field. Otherwise, it should move to the height field.  
  
Definition of Done:  
- The field for the mother's first name is correctly displayed and labeled.  
- The field is disabled for editing and does not allow insertion or updates.  
- The field adheres to the specified position and size constraints.  
- The form status is updated correctly based on the global loading flag.  
- Navigation logic works as specified, moving to the correct next field based on the product ID and definition.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include any direct database queries.

# Display Passport Details for Insured Person and Policy Holder

Type: POLICY\_HOLDER

User Story: Display Passport Details for Insured Person and Policy Holder  
  
Detailed Description:  
As a user, I want to view the passport details of both the insured person and the policy holder when I press the "Passport Details" button. This feature should fetch and display relevant passport information, including passport number, date of birth, age, and gender, for both the insured person and the policy holder. If the passport details are not available, an appropriate message should be displayed.  
  
Acceptance Criteria:  
1. When the "Passport Details" button is pressed, the system should:  
 - Retrieve the application number based on the insured person's sign card number or verification number.  
 - Check if the insured person has provided a passport as age proof. If so, fetch the passport number, date of birth, age, and gender.  
 - Check if the policy holder has provided a passport as age proof. If so, fetch the passport number, date of birth, age, and gender.  
 - If the policy holder has not provided a passport, iterate through the AML records to find a passport proof type and fetch the passport number.  
 - Determine if the insured person and policy holder are the same and set a flag accordingly.  
 - Pass the retrieved details as parameters to another form for display.  
  
2. If neither the insured person nor the policy holder has provided passport details, display a warning message indicating that passport details are not selected for either.  
  
Definition of Done:  
- The "Passport Details" button successfully retrieves and displays the passport details for both the insured person and the policy holder.  
- If passport details are not available, a warning message is displayed.  
- The feature is tested and verified to ensure it works as expected without any errors.  
  
DB Queries for Table Reference CRUD Operations:  
- Not applicable as the provided logic involves form-specific constructs and cannot be executed directly in the database without modification.

# Mother's Last Name Field in Policy Holder Section

Type: POLICY\_HOLDER

Title: Mother's Last Name Field in Policy Holder Section  
  
Acceptance Criteria:  
1. The field for the mother's last name should be disabled for editing.  
2. The maximum length of the mother's last name should be 50 characters.  
3. The field should be positioned correctly within the policy holder section.  
4. Upon changing the value of the mother's last name, if the form is not in a loading state, the form status should be updated.  
5. When navigating away from the mother's last name field:  
 - If the product ID is not 11 and the product definition is not 'CHILDGAIN', the focus should move to the beneficiary's name field.  
 - Otherwise, the focus should move to the height field.  
  
Definition of Done:  
- The mother's last name field is correctly displayed and positioned in the policy holder section.  
- The field is disabled for editing and adheres to the maximum length constraint.  
- The form status is updated correctly upon changing the value of the mother's last name.  
- Navigation logic works as specified based on the product ID and product definition.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include specific CRUD operations.

# Capture and Display Mother's Middle Name

Type: POLICY\_HOLDER

Title: Capture and Display Mother's Middle Name  
  
Acceptance Criteria:  
1. The field for the mother's middle name should be displayed on the policyholder's tab.  
2. The field should be read-only and should not allow any insert or update operations.  
3. The field should automatically navigate to the next field based on the product ID:  
 - If the product ID is not 11 and the product definition is not 'CHILDGAIN', navigate to the beneficiary's name field.  
 - Otherwise, navigate to the height field.  
4. If the form is not in a loading state, the form status should be set to 'Y' upon any change in the mother's middle name field.  
  
Definition of Done:  
- The mother's middle name field is visible and read-only on the policyholder's tab.  
- The field navigation works as per the specified conditions.  
- The form status is updated correctly when changes are made to the field, provided the form is not in a loading state.  
- All acceptance criteria are met and tested successfully.

# Voter's ID Verification for Policyholder and Insured Person

Type: POLICY\_HOLDER

Detailed description: As a user, I want to verify the Voter's ID details of the policyholder and insured person to ensure the accuracy and validity of the provided information.  
  
Acceptance criteria:  
1. When the Voter's ID verification button is pressed, the system should:  
 - Set a flag indicating that the Voter's ID verification is required.  
 - Retrieve the application number based on the insured person's sign card number or verification number.  
 - Check if the age proof for the insured person or policyholder is 'VI' (Voter's ID). If so, set a flag indicating that the Voter's ID is required.  
 - If the age proof is not 'VI', navigate to the 'AML' block, iterate through the records, and check if any proof type is 'VI'. If found, set the Voter's ID flag.  
 - Determine if the policyholder and insured person are the same and retrieve the state of residence of the policyholder.  
 - Add parameters for calling a verification form, including the application number, policy number, state, and Voter's ID number.  
 - If the Voter's ID flag is set, call the verification form. Otherwise, display a message indicating that Voter's ID details are not selected.  
  
Definition of Done:  
- The Voter's ID verification process is triggered correctly when the button is pressed.  
- All necessary parameters are correctly passed to the verification form.  
- Appropriate messages are displayed if Voter's ID details are not selected.  
- The system handles exceptions and errors gracefully.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Not applicable as the provided logic involves form-specific operations and parameter handling.

# Validate and Display Driving License Details

Type: POLICY\_HOLDER

User Story: Validate and Display Driving License Details  
  
Detailed description:   
As a user, I want to validate and display the driving license details of the insured person and policyholder, so that I can ensure the correct information is captured and displayed for further processing.  
  
Acceptance criteria:  
1. When the "Driving License" button is pressed, the system should:  
 - Retrieve the application number based on the insured person's sign card number or verification number.  
 - Check if the insured person's age proof is a driving license and retrieve the driving license number and date of birth.  
 - Check if the policyholder's age proof is a driving license and retrieve the driving license number and date of birth.  
 - If the policyholder's age proof is not a driving license, iterate through the AML records to find a driving license proof type and retrieve the driving license number.  
 - Determine if the insured person and policyholder are the same.  
 - Add parameters for the driving license number and date of birth for both the insured person and policyholder, along with other necessary parameters.  
 - If either the insured person or policyholder has a driving license number, call the form to display driving license details.  
 - If neither has a driving license number, display a warning message indicating that driving license details are not selected for the insured person or policyholder.  
  
Definition of Done:  
- The system correctly retrieves and validates the driving license details for both the insured person and policyholder.  
- The form to display driving license details is called if valid driving license numbers are found.  
- A warning message is displayed if no driving license details are selected for the insured person or policyholder.  
- The functionality is tested and verified to ensure it meets the acceptance criteria.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Not applicable as the provided XML content does not include direct database queries that can be executed independently of Oracle Forms constructs.

# Document DE Button Functionality

Type: POLICY\_HOLDER

Title: Document DE Button Functionality  
  
Acceptance Criteria:  
1. When the "Document DE" button is pressed, the system should:  
 - Check if a parameter list named 'Param1' exists and destroy it if it does.  
 - Create a new parameter list named 'Param1'.  
 - Add parameters to the list including application number, proposal number, module name, and policy holder's date of birth.  
 - Navigate to the 'AML' block and iterate through records to find a specific proof type ('PC') and store its value.  
 - Add the found proof type value to the parameter list.  
 - Increment a document counter.  
 - Call another form named 'AZBJ\_FINANCIAL\_DOCUMENT' with the parameter list.  
  
Definition of Done:  
- The "Document DE" button should be functional and perform all the specified actions when pressed.  
- The system should correctly handle the parameter list creation, addition of parameters, and navigation to the 'AML' block.  
- The system should successfully call the 'AZBJ\_FINANCIAL\_DOCUMENT' form with the correct parameters.  
- All actions should be performed without any errors or failures.  
  
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 that can be executed independently of Oracle Forms constructs.

# View and Manage EIA Details

Type: POLICY\_HOLDER

User Story: View and Manage EIA Details  
  
Detailed Description:  
As a user, I want to view and manage the EIA (Electronic Insurance Account) details of a policyholder so that I can ensure all necessary information is accurately recorded and updated.  
  
Acceptance Criteria:  
1. When the "EIA Details" button is pressed, the system should:  
 - Set a global flag indicating the button was clicked.  
 - Create a parameter list with the following parameters:  
 - Application number (either the sign card number or verification number of the insured person).  
 - Module flag set to 'BBU'.  
 - Property type value from the policy holder.  
 - Scrutiny number from the control block.  
 - Policy reference from the control block.  
 - Partner ID from the policy holder.  
 - Agent code from the agents block.  
 - Call the form 'AZBJ\_INSURANCE\_REPOSITORY' with the created parameter list.  
  
2. The system should then:  
 - Query the EIA details from the database using the application number and top indicator.  
 - Check if an EIA account exists for a new applicant and enable or disable the "EI Account Opening" field based on the result.  
  
Definition of Done:  
- The "EIA Details" button functionality is implemented and tested.  
- The parameter list is correctly created and passed to the form.  
- The form 'AZBJ\_INSURANCE\_REPOSITORY' is called with the correct parameters.  
- The EIA details are queried and displayed correctly.  
- The "EI Account Opening" field is enabled or disabled based on the existence of an EIA account for a new applicant.  
  
DB Queries for Table Reference CRUD Operations:  
- Query to check EIA details:  
 ```sql  
 SELECT EIA\_ACCOUNT\_TYPE, EIA\_DOC\_TYPE  
 INTO pk\_vars.v\_EIA\_ACCOUNT\_TYPE, pk\_vars.v\_EIA\_DOC\_TYPE  
 FROM azbj\_eia\_details  
 WHERE application\_no = TO\_CHAR(NVL(:insured\_person.ip\_verf\_no, :insured\_person.ip\_sign\_card\_no))  
 AND top\_indicator = 'Y';  
 ```  
  
- Query to count EIA accounts:  
 ```sql  
 SELECT COUNT(1)  
 INTO pk\_vars.v\_eia\_exist  
 FROM azbj\_eia\_details  
 WHERE application\_no = TO\_CHAR(NVL(:insured\_person.ip\_verf\_no, :insured\_person.ip\_sign\_card\_no))  
 AND top\_indicator = 'Y';  
 ```  
  
- Query to check new applicant EIA accounts:  
 ```sql  
 SELECT COUNT(1)  
 INTO v\_eia\_exists  
 FROM azbj\_eia\_details  
 WHERE application\_no = TO\_CHAR(NVL(:insured\_person.ip\_verf\_no, :insured\_person.ip\_sign\_card\_no))  
 AND top\_indicator = 'Y'  
 AND eia\_account\_type = 'New\_Applicant';  
 ```  
  
Note: The above queries are provided for reference and should be adapted to the specific database schema and requirements.

# EI Account Opening for Policyholder

Type: POLICY\_HOLDER

Title: EI Account Opening for Policyholder  
  
Acceptance Criteria:  
1. When the button is pressed, the system should:  
 - Retrieve the policyholder's residential and mailing state, country, and bank name from the database.  
 - Check if an EI Account already exists for the policyholder.  
 - If an EI Account exists, retrieve the internal values for the state, country, and bank name.  
 - If no EI Account exists, insert a new record into the EI Account details table with the policyholder's information.  
 - Commit the transaction.  
 - Redirect the user to a webpage with the policyholder's application number, policy number, and other relevant details.  
 - Display an error message if the policyholder is not registered for an e-Insurance Account.  
  
Definition of Done:  
- The button should trigger the retrieval and validation of policyholder details.  
- The system should correctly check for existing EI Account records.  
- New records should be inserted into the EI Account details table if no existing records are found.  
- The transaction should be committed successfully.  
- The user should be redirected to the appropriate webpage with the necessary details.  
- Appropriate error messages should be displayed for any issues encountered during the process.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Retrieve policyholder details:  
 ```sql  
 SELECT ph\_res\_state, ph\_m\_state, ph\_res\_country, ph\_m\_country, acc\_bank\_name  
 INTO v\_ph\_res\_state, v\_ph\_m\_state, v\_ph\_res\_country, v\_ph\_m\_country, v\_acc\_bank\_name  
 FROM azbj\_proposal\_appln\_det  
 WHERE de\_flag = 'D2' AND appln\_no = NVL(:insured\_person.ip\_verf\_no, :insured\_person.ip\_sign\_card\_no);  
 ```  
  
- Check for existing EI Account:  
 ```sql  
 SELECT COUNT(1)  
 INTO v\_exists  
 FROM azbj\_eia\_details  
 WHERE application\_no = TO\_CHAR(NVL(:insured\_person.ip\_verf\_no, :insured\_person.ip\_sign\_card\_no))  
 AND top\_indicator = 'Y' AND eia\_account\_type = 'New\_Applicant';  
 ```  
  
- Insert new EI Account details:  
 ```sql  
 INSERT INTO azbj\_eia\_deqc\_de\_details (application\_no, policy\_no, first\_name, middle\_name, last\_name, fthr\_hus\_name, gender, dob, dob\_proof, id\_proof, pan\_card\_no, perm\_addr\_1, perm\_addr\_2, perm\_addr\_3, perm\_addr\_area, perm\_city, perm\_pin, perm\_state, prem\_county, perm\_add\_proof, corr\_addr\_1, corr\_addr\_2, corr\_addr\_3, corr\_addr\_area, corr\_city, corr\_pin, corr\_state, corr\_county, corr\_add\_proof, appl\_tele\_no, appl\_alt\_tel\_no, appl\_mob\_no, appl\_email\_id, bank\_acc\_type, bank\_acc\_no, bank\_name, branch\_name, micr\_code, ifsc\_code, uid\_no, module\_flag, top\_indicator, create\_user, agent\_code)  
 SELECT TO\_CHAR(appln\_no), SUBSTR(policy\_ref, 1, 15), SUBSTR(ph\_first\_name, 1, 25), SUBSTR(ph\_middle\_name, 1, 25), SUBSTR(ph\_last\_name, 1, 25), CASE WHEN ph\_fthr\_name IS NOT NULL THEN SUBSTR(ph\_fthr\_name, 1, 30) ELSE SUBSTR(ph\_spouse\_name, 1, 30) END, ph\_gender, ph\_dob, SUBSTR(ph\_age\_proof, 1, 3), NULL, SUBSTR(ph\_pan\_no, 1, 10), SUBSTR(ph\_res\_addr1, 1, 25), SUBSTR(ph\_res\_addr2, 1, 25), NULL, SUBSTR(ph\_res\_place, 1, 25), SUBSTR(ph\_res\_city, 1, 25), SUBSTR(ph\_res\_pin, 1, 10), v\_state, v\_nation, SUBSTR(ph\_address\_proof, 1, 3), SUBSTR(ph\_m\_addr1, 1, 25), SUBSTR(ph\_m\_addr2, 1, 25), NULL, SUBSTR(ph\_res\_place, 1, 25), SUBSTR(ph\_m\_city, 1, 25), SUBSTR(ph\_m\_pin, 1, 10), v\_m\_state, v\_m\_nation, SUBSTR(ph\_m\_address\_proof, 1, 3), SUBSTR(ph\_res\_tel, 1, 13), NULL, SUBSTR(ph\_mobile, 1, 11), SUBSTR(ph\_email, 1, 50), CASE WHEN UPPER(acc\_type) LIKE '%SAV%' THEN 'S' ELSE 'C' END, SUBSTR(account\_no, 1, 30), v\_bank, SUBSTR(acc\_bank\_brch, 1, 30), SUBSTR(micr\_code, 1, 15), SUBSTR(ifsc\_code, 1, 11), NULL, 'NB', 'Y', USER, :agents.ag\_agent\_code  
 FROM azbj\_proposal\_appln\_det  
 WHERE de\_flag = 'D2' AND appln\_no = NVL(:insured\_person.ip\_verf\_no, :insured\_person.ip\_sign\_card\_no);  
 ```  
  
- Commit the transaction:  
 ```sql  
 COMMIT;  
 ```

# Implement E-KYC Aadhaar Verification Button

Type: POLICY\_HOLDER

User Story: Implement E-KYC Aadhaar Verification Button  
  
Detailed description:   
As a user, I want to have a button labeled "E-KYC Aadhaar" on the POLICY\_HOLDER section of the form, which when pressed, will initiate the Aadhaar verification process. This button should be disabled by default and should be visually distinct with a yellow background and black text.  
  
Acceptance criteria:  
1. When the "E-KYC Aadhaar" button is pressed, it should:  
 - Create a parameter list named 'Param1'.  
 - Add the Aadhaar number and module name ('BBU') to the parameter list.  
 - Increment the eKYC review count.  
 - Call the form 'AZBJ\_ADHAAR\_KYC' with the created parameter list.  
  
Definition of Done:  
- The button is present on the POLICY\_HOLDER section with the specified label and visual properties.  
- The button is disabled by default.  
- Pressing the button initiates the Aadhaar verification process as described in the acceptance criteria.  
- The form 'AZBJ\_ADHAAR\_KYC' is called with the correct parameters when the button is pressed.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Not applicable as the provided logic does not include direct CRUD operations on the database tables.  
  
Explanation of Oracle Form Logic:  
- The button press event triggers a series of actions including parameter list creation, parameter addition, and form calling.  
- The parameter list is used to pass necessary data (Aadhaar number and module name) to the called form.  
- The eKYC review count is incremented to keep track of the number of times the verification process has been initiated.  
  
Please review the user story and provide feedback.

# Policy Holder Search Functionality

Type: POLICY\_HOLDER

User Story: Policy Holder Search Functionality  
  
Detailed Description:  
As a user, I want to search for a policy holder's details using a search button, so that I can retrieve and display relevant information about the policy holder.  
  
Acceptance Criteria:  
1. When the search button is pressed, the system should:  
 - Set a flag indicating that the partner ID is being used.  
 - Retrieve the partner ID of the policy holder.  
 - Call a function to search for customer or partner details based on the partner ID.  
 - Display a warning message if the agent code matches specific patterns and the partner ID is the same as the LG partner ID.  
 - Clear certain fields if the partner ID has changed.  
 - Update the policy holder's name in the sofa information data.  
 - Validate and rate the policy if the product ID is 11 (childcare) and the date of birth has changed.  
 - Display an error message if the calculated entry age is greater than 100.  
 - Retrieve and display the place of birth from the partner extension table.  
 - Calculate the entry age for the insured person and display a message if the insured person is a minor.  
 - Enable or disable certain fields based on the age of the insured person.  
 - Validate and rate the policy again if the product ID is 11 or the product definition is 'CHILDGAIN'.  
 - Automatically load classes for joint life or childcare products.  
 - Enable or disable nominee fields based on certain conditions.  
 - Navigate to the 'PH\_RELATION' field.  
  
Definition of Done:  
- The search button functionality is implemented and tested.  
- The system retrieves and displays the correct policy holder details.  
- All specified conditions and validations are met.  
- The system displays appropriate messages and updates fields as required.  
- The functionality is reviewed and approved by stakeholders.  
  
DB Queries for Table Reference CRUD Operations:  
- Retrieve place of birth from the partner extension table:  
 ```sql  
 SELECT PLACE\_OF\_BIRTH  
 INTO v\_PLACE\_OF\_BIRTH  
 FROM azbj\_partner\_extn  
 WHERE PART\_ID = :PH\_PART\_ID;  
 ```  
  
- Retrieve the entry age for the insured person:  
 ```sql  
 SELECT azbj\_bk\_All\_validations.AZBJ\_calc\_entry\_age(:insured\_person.ip\_date\_of\_birth,  
 NVL(:COVERHEAD.CH\_INCEPTION\_DATE, :CONTROL.CN\_EFFECTIVE\_DATE),  
 'P', 1)  
 INTO ip\_age  
 FROM dual;  
 ```  
  
- Retrieve the multiplier for the main cover code:  
 ```sql  
 SELECT multiplier  
 INTO v\_multiplier  
 FROM azbj\_tasa\_suc\_multiplier  
 WHERE product\_id = :control.cn\_product\_id  
 AND cover\_code = v\_main\_cover\_code  
 AND :coverhead.ch\_benefit\_term BETWEEN min\_bt AND max\_bt  
 AND :coverhead.ch\_prem\_term BETWEEN min\_pt AND max\_pt;  
 ```  
  
- Retrieve the date value for loading on SAR:  
 ```sql  
 SELECT date\_value  
 INTO V\_on\_sar\_date  
 FROM azbj\_system\_constants  
 WHERE sys\_type = 'NB'  
 AND SYS\_CODE = 'LOADING\_ON\_SAR'  
 AND CHAR\_VALUE = 'Y';  
 ```  
  
- Retrieve the frequency factor date:  
 ```sql  
 SELECT date\_value  
 INTO v\_freq\_fac\_date  
 FROM azbj\_system\_constants  
 WHERE sys\_type = 'NB'  
 AND SYS\_CODE LIKE 'FREQ\_FACTOR'  
 AND CHAR\_VALUE = 'Y';  
 ```  
  
- Retrieve the count of supervisor approval details:  
 ```sql  
 SELECT COUNT(1)  
 INTO v\_appr\_cnt  
 FROM azbj\_supervisor\_appr\_det  
 WHERE application\_no = TO\_CHAR(NVL(:insured\_person.ip\_verf\_no, :insured\_person.ip\_sign\_card\_no))  
 AND create\_date = (SELECT MAX(create\_date)  
 FROM azbj\_supervisor\_appr\_det  
 WHERE application\_no = TO\_CHAR(NVL(:insured\_person.ip\_verf\_no, :insured\_person.ip\_sign\_card\_no)))  
 AND pan\_std\_flag IN ('N', 'R');  
 ```  
  
- Retrieve the non-standard age proof rate:  
 ```sql  
 SELECT value\_consider  
 INTO v\_nonst\_rate  
 FROM azbj\_grp\_calc\_assumptions  
 WHERE master\_policy\_no = :control.cn\_master\_policy  
 AND TYPE = 'NONSTD\_BAND'  
 AND sub\_type = 'SUMASSURED'  
 AND :coverhead.ch\_sum\_assured BETWEEN value\_from AND value\_to  
 AND :covers.cv\_entry\_age BETWEEN NVL(age\_from, 0) AND NVL(age\_to, 99)  
 AND end\_date IS NULL;  
 ```  
  
- Retrieve the GST amount:  
 ```sql  
 SELECT azbj\_pk\_gst.get\_gst\_amount(p\_product\_id => :control.cn\_product\_id,  
 p\_cover\_code => :covers.cv\_cover\_code,  
 p\_premium => :coverhead.ch\_tg\_prem,  
 p\_effectve\_date => NVL(:control.service\_tax\_date, pme\_api.opus\_date),  
 p\_policy\_year => 1,  
 p\_mailadd\_pin\_code => :policy\_holder.ph\_mailing\_postcode,  
 p\_servadd\_pin\_code => azbj\_pk\_gst.get\_coll\_branch\_pincode(:control.cn\_branch\_code),  
 p\_sumassured => :covers.cv\_sum\_insured\_whole\_cover,  
 p\_event\_code => 'PREMIUM',  
 p\_error => v\_error,  
 p\_gst\_dtl\_string => v\_azbj\_gst\_tab,  
 p\_contract\_id => :control.cn\_contract\_id,  
 p\_due\_date => :coverhead.CH\_INCEPTION\_DATE)  
 INTO v\_state\_gst, v\_integrated\_gst, v\_central\_gst, v\_ut\_gst;  
 ```

# Display and Interact with SUSAC Section

Type: SUSAC

Title: Display and Interact with SUSAC Section  
  
Acceptance Criteria:  
1. The "SUSAC" section should display the following fields:  
 - Receipt Number (SA\_RCPTNO)  
 - Amount (SA\_AMT)  
 - Date Received (SA\_DATERECD)  
 - Cheque Status (CHEQUE\_STATUS)  
 - Premium Payer (PREMIUM\_PAYER)  
 - Receipting Branch Code (RCPT\_BRANCH\_CODE)  
 - Receipting Branch Name (RCPT\_BRANCH\_NAME)  
 - State of the Branch (RCPT\_BRANCH\_STATE)  
2. The "SUSAC" section should also display a calculated total amount (TOTAL\_SUSAC) which sums up the amounts from the "SA\_AMT" field.  
3. The "SUSAC" section should include a "Close" button to exit the section.  
4. The fields should be displayed in a vertical orientation and should be read-only.  
5. The "SUSAC" section should be visually organized with appropriate labels and prompts for each field.  
  
Definition of Done:  
- The "SUSAC" section is implemented and displays all the specified fields.  
- The total amount is correctly calculated and displayed.  
- The "Close" button functions as expected.  
- All fields are read-only and properly labeled.  
- The section is visually organized 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.

# Close Button Functionality

Type: SUSAC

Title: Close Button Functionality  
  
Acceptance Criteria:  
1. When the "Close" button is pressed, the system should check the value of a control variable `v\_canvas\_adj`.  
2. If the value of `v\_canvas\_adj` is 'Y':  
 - The current window should be hidden.  
 - The current view should be hidden.  
 - A new window named "Alert List" should be displayed.  
 - A new view named "Alert List" should be displayed.  
 - The focus should be moved to the "Alert" block.  
 - The focus should be moved to the "Cancel" button within the "Control" block.  
 - The value of `v\_canvas\_adj` should be set to 'N'.  
3. If the value of `v\_canvas\_adj` is not 'Y':  
 - The current window should be hidden.  
 - The current view should be hidden.  
 - The system should navigate to the "Covers" tab.  
 - The focus should be moved to the "Covers" block.  
  
Definition of Done:  
- The "Close" button functionality is implemented as per the acceptance criteria.  
- The system correctly hides and shows windows and views based on the value of `v\_canvas\_adj`.  
- The system correctly navigates to the specified blocks and items.  
- The functionality is tested and verified to ensure it works as expected.

# Manage Insurance-Related Attributes and Options

Type: SOL\_COVERHEAD

Title: Manage Insurance-Related Attributes and Options  
  
Acceptance Criteria:  
1. The interface should allow the user to input and update various insurance-related attributes, including but not limited to:  
 - Discount percentage  
 - Frequency of payments  
 - Percentage values for different insurance options  
 - Flags for specific conditions (e.g., NRI GST Waiver, Fully Vaccinated Flag)  
 - Sum assured on death  
 - Premium holidays  
 - Deferred periods  
 - Income payout dates and frequencies  
 - Annuity options and types  
 - Total premiums and discounts  
2. The interface should support different data types, such as text, number, and date, and should validate the input accordingly.  
3. The interface should provide options to select from predefined lists for certain attributes.  
4. The interface should include checkboxes for boolean attributes, with clear labels indicating their purpose.  
5. The interface should display prompts and labels for each attribute, ensuring that the user understands what information is required.  
6. The interface should allow for the visibility of certain fields to be toggled based on specific conditions or user actions.  
7. The interface should include buttons for actions such as deleting a cover or confirming the input.  
  
Definition of Done:  
- The user can successfully input and update all required insurance-related attributes.  
- The interface validates the input data types and constraints.  
- The user can select from predefined lists where applicable.  
- The user can toggle the visibility of certain fields based on specific conditions.  
- The interface includes clear prompts and labels for all attributes.  
- The user can perform actions such as deleting a cover or confirming the input using buttons provided in the interface.  
- The interface is tested and verified to ensure all functionalities work as expected.  
  
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 SPW Flag to control SPW Percentage field

Type: SOL\_COVERHEAD

Title: Manage SPW Flag to control SPW Percentage field  
  
Acceptance Criteria:  
1. When the SPW Flag is checked (value 'Y'), the SPW Percentage field should be enabled.  
2. When the SPW Flag is unchecked (value 'N'), the SPW Percentage field should be disabled.  
  
Definition of Done:  
- The SPW Flag can be checked or unchecked.  
- The SPW Percentage field's enabled/disabled state changes dynamically based on the SPW Flag's value.  
- The changes are reflected immediately without requiring a page refresh.  
  
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 Auto-Pay Rebate Option

Type: SOL\_COVERHEAD

Title: Manage Auto-Pay Rebate Option  
  
Acceptance Criteria:  
- The auto-pay rebate option should be represented as a checkbox.  
- The checkbox should have a default value of "N" (unchecked).  
- When the checkbox is checked, it should store the value "Y".  
- When the checkbox is unchecked, it should store the value "N".  
- The checkbox should be hidden from view by default.  
- The checkbox should be located within the "Solution Product" tab.  
  
Definition of Done:  
- The auto-pay rebate checkbox is implemented and behaves as described in the acceptance criteria.  
- The checkbox is hidden by default and only becomes visible when required.  
- The checkbox correctly stores "Y" when checked and "N" when unchecked.  
- The checkbox is placed within the "Solution Product" tab.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the checkbox is not directly tied to a database item.

# Implement 'Insurance for All' Checkbox

Type: SOL\_COVERHEAD

Title: Implement 'Insurance for All' Checkbox  
  
Acceptance Criteria:  
1. The checkbox for 'Insurance for All' should be available within the solution product section.  
2. The checkbox should have two states: checked (Y) and unchecked (N).  
3. The default state of the checkbox should be unchecked (N).  
4. When the checkbox state is changed, no additional actions or triggers should be executed.  
  
Definition of Done:  
1. The checkbox for 'Insurance for All' is implemented and visible within the solution product section.  
2. The checkbox correctly reflects the checked and unchecked states.  
3. The default state of the checkbox is set to unchecked.  
4. The checkbox state change does not trigger any additional actions.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Not applicable as there are no database operations specified in the provided XML content.

# Annuity Frequency Selection

Type: SOL\_COVERHEAD

Title: Annuity Frequency Selection  
  
Acceptance Criteria:  
1. The user should be able to see a dropdown list labeled "Annuity Freq" on the "SOLUTION\_PRODUCT2" tab.  
2. The dropdown list should contain five predefined frequency options.  
3. The dropdown list should be positioned correctly on the screen as per the design specifications.  
4. The selected frequency should be stored temporarily and should not be directly saved to the database.  
  
Definition of Done:  
1. The dropdown list for "Annuity Freq" is implemented and visible on the "SOLUTION\_PRODUCT2" tab.  
2. The dropdown list contains exactly five predefined options.  
3. The dropdown list is correctly positioned as per the design specifications.  
4. The selected frequency is stored temporarily and does not directly interact with the database.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the selected frequency is stored temporarily and does not directly interact with the database.

# Input Percentage Increase in SA

Type: SOL\_COVERHEAD

Title: Input Percentage Increase in SA  
  
Acceptance Criteria:  
1. The field should accept numerical input only.  
2. The field should be displayed on the "SOLUTION\_PRODUCT2" tab.  
3. The field should be labeled as "% Increasing in SA" and the label should be aligned to the right.  
4. The field should be visible only when required.  
5. The field should allow both insertion and updating of data.  
6. The field should display a list of predefined values for selection.  
  
Definition of Done:  
1. The field is implemented and can accept numerical input.  
2. The field is correctly placed on the "SOLUTION\_PRODUCT2" tab.  
3. The label "% Increasing in SA" is correctly aligned to the right.  
4. The field visibility is controlled as per the requirements.  
5. The field allows data insertion and updates.  
6. The field displays a list of predefined values for user selection.  
  
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.

# Frequency Selection Dropdown

Type: SOL\_COVERHEAD

Title: Frequency Selection Dropdown  
  
Acceptance Criteria:  
1. The frequency selection should be available as a dropdown list with predefined options.  
2. The dropdown list should be displayed on the "COVERS" tab of the user interface.  
3. The dropdown list should allow only uppercase characters.  
4. The dropdown list should be editable, allowing users to insert and update the frequency value.  
5. The dropdown list should be visually aligned and justified to the right of its prompt label "SPW Freq".  
  
Definition of Done:  
1. The dropdown list for frequency selection is implemented and visible on the "COVERS" tab.  
2. The dropdown list contains the predefined frequency options.  
3. The dropdown list accepts only uppercase characters.  
4. Users can insert and update the frequency value in the dropdown list.  
5. The prompt label "SPW Freq" is right-justified and aligned with the dropdown list.  
  
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 table references.

# Implement Fully Vaccinated Checkbox

Type: SOL\_COVERHEAD

Title: Implement Fully Vaccinated Checkbox  
  
Acceptance Criteria:  
1. The checkbox should have a label "FULLY\_VACCINATED\_FLAG".  
2. The checkbox should have two states: checked and unchecked.  
 - When checked, the value should be "Y".  
 - When unchecked, the value should be "N".  
3. The checkbox should be initialized to "N" (unchecked) by default.  
4. The checkbox should be located on the "SOLUTION\_PRODUCT2" tab page.  
5. The checkbox should be hidden from view by default.  
6. The checkbox should not be linked to any database item.  
  
Definition of Done:  
1. The checkbox is implemented and meets all acceptance criteria.  
2. The checkbox is tested to ensure it correctly reflects the vaccination status.  
3. The checkbox is hidden by default and only visible when required.  
4. The checkbox is not linked to any database item and does not affect database operations.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the checkbox is not linked to any database item.

# Manage Cash Bonus Information

Type: SOL\_COVERHEAD

Title: Manage Cash Bonus Information  
  
Acceptance Criteria:  
1. The cash bonus field should be displayed as a list item with predefined options.  
2. The field should be editable, allowing users to insert and update the cash bonus information.  
3. The cash bonus field should be visible only under specific conditions.  
4. The field should be aligned and formatted according to the design specifications, ensuring consistency in the user interface.  
  
Definition of Done:  
1. The cash bonus field is implemented as a list item with predefined options.  
2. Users can successfully insert and update the cash bonus information.  
3. The visibility of the cash bonus field is controlled based on predefined conditions.  
4. The field is properly aligned and formatted as per the design specifications.  
5. The feature is tested and verified to ensure it meets the acceptance criteria.  
  
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.

# Retention Flag Confirmation and Reason Entry

Type: SOL\_COVERHEAD

Title: Retention Flag Confirmation and Reason Entry  
  
Acceptance Criteria:  
1. When the retention flag is selected and its value is not 'NO', a confirmation message should be displayed asking, "Do you really want to select the Reinsurance flag for this case?"  
2. If the user confirms the selection:  
 - The field for entering the reason for the retention flag should become visible and enabled.  
 - A message should be displayed prompting the user to enter the reason for the retention flag.  
3. If the user does not confirm the selection:  
 - The retention flag should be reset to 'NO'.  
 - The field for entering the reason for the retention flag should be hidden and disabled.  
4. If the retention flag is set to 'NO':  
 - The field for entering the reason for the retention flag should be hidden and disabled.  
  
Definition of Done:  
- The confirmation message is displayed correctly when the retention flag is selected.  
- The field for entering the reason for the retention flag becomes visible and enabled upon confirmation.  
- The retention flag is reset to 'NO' and the reason field is hidden and disabled if the user does not confirm the selection.  
- The retention flag and reason field behave as expected when the retention flag is set to 'NO'.  
- All functionalities are tested and verified to work as described.

# Retention Flag Confirmation and Reason Entry

Type: SOL\_COVERHEAD

Title: Retention Flag Confirmation and Reason Entry  
  
Acceptance Criteria:  
1. When the retention flag is changed from 'NO' to any other value, a confirmation message should be displayed asking, "Do you really want to select the Reinsurance flag for this case?"  
2. If the user confirms the action, the field for entering the reason for the retention flag should become visible and enabled.  
3. If the user cancels the action, the retention flag should be reset to 'NO' and the reason field should be hidden and disabled.  
4. If the retention flag is set to 'NO', the reason field should be hidden and disabled.  
  
Definition of Done:  
- The confirmation message is displayed correctly when the retention flag is changed.  
- The reason field visibility and enabled state are correctly toggled based on user actions.  
- The retention flag is reset to 'NO' if the user cancels the action.  
- The reason field is hidden and disabled when the retention flag is 'NO'.

# Automatic Calculation of Sum Assured on Death

Type: SOL\_COVERHEAD

Detailed description: As a user, I want the system to automatically calculate the "Sum Assured on Death" based on specific conditions related to the product ID and booking frequency, so that I can ensure accurate insurance coverage amounts.  
  
Acceptance criteria:  
1. If the product ID is 343 and the booking frequency is not '01':  
 - The "Sum Assured on Death" should be calculated as the greater value between:  
 - The product of the multiplier, premium amount, and booking frequency.  
 - 105% of the product of the premium amount and booking frequency.  
 - The calculated value should be rounded and set to zero if null.  
2. If the product ID is 343 and the booking frequency is '01':  
 - The "Sum Assured on Death" should be calculated as the greater value between:  
 - The product of the multiplier and premium amount.  
 - 105% of the premium amount.  
 - The calculated value should be rounded and set to zero if null.  
  
Definition of Done:  
- The system correctly calculates and displays the "Sum Assured on Death" based on the specified conditions.  
- The calculations are accurate and reflect the business logic described.  
- The feature is tested and verified to ensure it meets the acceptance criteria.  
  
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.

# NRI GST Waiver Checkbox Functionality

Type: SOL\_COVERHEAD

Title: NRI GST Waiver Checkbox Functionality  
  
Acceptance Criteria:  
1. When the NRI GST Waiver checkbox is checked:  
 - The system should set a control form status flag to 'Y'.  
 - If the control flag `BKD\_FLG` is 'Y', the system should enable the inception date field if it is currently disabled.  
 - If the control flag `BKD\_FLG` is not 'Y', the system should disable the inception date field if it is currently enabled.  
  
Definition of Done:  
- The NRI GST Waiver checkbox should be functional and visible on the user interface.  
- The inception date field should dynamically change its enabled status based on the checkbox state and the control flag.  
- The control form status flag should be correctly set to 'Y' when the checkbox is checked.  
- All changes should be tested and verified to ensure they meet the acceptance criteria.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include any direct database queries.

# Enable/Disable RI Reference Number based on Refer To RI selection

Type: SOL\_COVERHEAD

Title: Enable/Disable RI Reference Number based on Refer To RI selection  
  
Acceptance Criteria:  
1. When the "Refer To RI" list item is set to 'Y':  
 - The "RI Reference Number" field should be enabled.  
2. When the "Refer To RI" list item is set to any value other than 'Y':  
 - The "RI Reference Number" field should be cleared and disabled.  
  
Definition of Done:  
- The "RI Reference Number" field is enabled when "Refer To RI" is 'Y'.  
- The "RI Reference Number" field is cleared and disabled when "Refer To RI" is not 'Y'.  
- The changes are tested and verified to work as expected in the user interface.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided logic does not include any direct database CRUD operations.

# Manage Salaried Rebate Flag for Solution Product

Type: SOL\_COVERHEAD

Title: Manage Salaried Rebate Flag for Solution Product  
  
Acceptance Criteria:  
- The salaried rebate flag should be represented as a checkbox.  
- The checkbox should have a default value of 'N' (unchecked).  
- When the checkbox is checked, it should store the value 'Y'.  
- When the checkbox is unchecked, it should store the value 'N'.  
- The checkbox should be located within the solution product section.  
- The checkbox should not be visible to the user by default.  
  
Definition of Done:  
- The salaried rebate flag checkbox is implemented and integrated within the solution product section.  
- The checkbox correctly reflects the checked and unchecked states with the values 'Y' and 'N' respectively.  
- The checkbox is hidden from the user interface by default.  
- The functionality is tested and verified to ensure it meets the acceptance criteria.  
  
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.

# Input Multiplier Value and Auto-Focus to Next Item

Type: SOL\_COVERHEAD

Title: Input Multiplier Value and Auto-Focus to Next Item  
  
Acceptance Criteria:  
- The "Multiplier" field should accept numeric values up to two decimal places.  
- The field should be hidden from view.  
- Upon entering a value in the "Multiplier" field, the focus should automatically move to the next item, which is the "Package" field.  
  
Definition of Done:  
- The "Multiplier" field accepts numeric values formatted to two decimal places.  
- The field is not visible to the user.  
- After entering a value, the focus shifts to the "Package" field automatically.  
  
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 Income Payout Frequency

Type: SOL\_COVERHEAD

Detailed description: As a user, I want to manage the frequency of income payouts for a solution product, so that I can ensure the correct payout schedule is maintained.  
  
Acceptance criteria:  
1. The user should be able to select the frequency of income payouts from a predefined list.  
2. The available options for payout frequency should be displayed in uppercase.  
3. The field for selecting payout frequency should be visible and enabled for user interaction.  
4. The user should be able to insert and update the payout frequency.  
5. The field should have a maximum length of 500 characters.  
6. The field should be displayed on the "Solution Product" tab.  
  
Definition of Done:  
1. The functionality to select and update the payout frequency is implemented.  
2. The field is visible and enabled for user interaction.  
3. The predefined list of payout frequencies is displayed in uppercase.  
4. The field adheres to the specified maximum length.  
5. The field is correctly placed on the "Solution Product" tab.  
6. All acceptance criteria are met and tested.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include specific database queries.

# Automatic Calculation of Sum Assured Based on GMI

Type: SOL\_COVERHEAD

Title: Automatic Calculation of Sum Assured Based on GMI  
  
Acceptance Criteria:  
1. When the product with ID 299 is selected:  
 - If the GMI field is empty, the system should prompt the user to enter a value in the GMI field.  
 - If the GMI field is not empty, the system should calculate the sum assured by multiplying the GMI value by 144.  
 - The sum assured field should be displayed as read-only (not editable).  
  
Definition of Done:  
- The system correctly identifies when the product with ID 299 is selected.  
- The system prompts the user to enter a GMI value if it is empty.  
- The system calculates the sum assured correctly based on the GMI value.  
- The sum assured field is displayed as read-only after calculation.  
- All acceptance criteria are met and tested successfully.

# Automatic Update of Benefit Term and Premium Term

Type: SOL\_COVERHEAD

Title: Automatic Update of Benefit Term and Premium Term  
  
Acceptance Criteria:  
1. When a premium term is entered in the 'sol\_coverhead' section and the package code is not null:  
 - The system should navigate to the 'sol\_covers' section.  
 - The system should iterate through all records in the 'sol\_covers' section.  
 - For each record, the benefit term and premium term should be updated to match the values from the 'sol\_coverhead' section.  
 - The iteration should stop when the last record is reached.  
  
Definition of Done:  
- The system correctly updates the benefit term and premium term for all records in the 'sol\_covers' section based on the values entered in the 'sol\_coverhead' section.  
- The functionality is tested and verified to ensure data consistency across related records.  
- The user interface reflects the changes without any errors or inconsistencies.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- No direct database queries are provided in the XML content. The logic is implemented within the application layer to ensure data consistency across related records.

# Auto-update Benefit and Premium Terms in Related Covers

Type: SOL\_COVERHEAD

Title: Auto-update Benefit and Premium Terms in Related Covers  
  
Acceptance Criteria:  
1. When a benefit term is entered in the cover head section and the package code is not null, the system should:  
 - Navigate to the related covers section.  
 - Iterate through all cover records.  
 - For each cover record where the cover code is not null, update the benefit term and premium term to match the values entered in the cover head section.  
 - Ensure the iteration stops when the last record is reached.  
  
Definition of Done:  
- The system correctly updates the benefit term and premium term for all related cover records based on the value entered in the cover head section.  
- The process should only occur if the package code in the cover head section is not null.  
- The user should be able to see the updated values in the related covers section without any errors.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided logic is specific to Oracle Forms and involves navigation and field updates within the form, which cannot be directly translated to SQL queries.

# Auto-update Sum Insured in sol\_covers based on Sum Assured in sol\_coverhead

Type: SOL\_COVERHEAD

Title: Auto-update Sum Insured in sol\_covers based on Sum Assured in sol\_coverhead  
  
Acceptance Criteria:  
1. When the "sum assured" field in the "sol\_coverhead" section is not null and the "solution name" field is not equal to 1, the system should:  
 - Navigate to the "sol\_covers" section.  
 - Iterate through all records in the "sol\_covers" section.  
 - Update the "sum insured whole cover" field in each record with the value from the "sum assured" field in the "sol\_coverhead" section.  
 - Stop the iteration when the last record is reached.  
  
Definition of Done:  
- The system correctly navigates to the "sol\_covers" section when the specified conditions are met.  
- All records in the "sol\_covers" section are updated with the value from the "sum assured" field in the "sol\_coverhead" section.  
- The process stops correctly at the last record.  
- The feature is tested and verified to ensure it works as expected without any errors.  
  
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 Calculation and Update of Benefit and Premium Terms

Type: SOL\_COVERHEAD

Title: Automatic Calculation and Update of Benefit and Premium Terms  
  
Acceptance Criteria:  
1. If the product definition matches specific criteria (e.g., 'NEW\_UG\_EASY\_PENSION\_PLUS%', 'MOD\_NEW\_UG\_EASY\_PEN\_PLUS', 'FUTURE\_SECURE') or if the pension flag is greater than 0:  
 - If the insured person's date of birth is not null and the vesting age is greater than 0:  
 - Calculate the entry age using the insured person's date of birth and the inception or effective date.  
 - Set the benefit term as the difference between the vesting age and the entry age.  
 - If the product definition matches specific criteria and the booking frequency is not '01', set the premium term as the difference between the vesting age and the entry age; otherwise, set the premium term to 0.  
 - If the package is not null, update all related records in the 'covers' block with the calculated benefit term and premium term.  
  
2. When the item is validated, if the global loading flag is 'F', set the form status to 'Y'.  
  
Definition of Done:  
- The system correctly calculates and updates the benefit term and premium term based on the specified conditions.  
- All related records in the 'covers' block are updated with the correct values.  
- The form status is updated appropriately when the item is validated.  
  
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.

# Implement Money Back Option in New Business Module

Type: SOL\_COVERHEAD

Title: Implement Money Back Option in New Business Module  
  
Acceptance Criteria:  
1. The "Money Back Option" should be a list item with three predefined options.  
2. The list item should be visible and enabled by default.  
3. The "Money Back Option" should be located within the "New Business" module and should be part of the "Solution Product" tab.  
4. The prompt for the "Money Back Option" should be centered and bold.  
5. The list item should allow both insertion and updates.  
  
Definition of Done:  
1. The "Money Back Option" list item is implemented and visible in the "New Business" module.  
2. The list item contains three predefined options.  
3. The prompt for the list item is centered and bold.  
4. The list item allows for insertion and updates.  
5. The feature is tested and verified to be working as expected.  
  
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 table references.

# Display and Select Discount Options

Type: SOL\_COVERHEAD

Title: Display and Select Discount Options  
  
Acceptance Criteria:  
1. The discount options should be displayed in a list format.  
2. The discount list should be centered and have a specific width and height.  
3. The discount list should be located at a specific position on the screen.  
4. The discount list should have a specific background color and font style.  
5. The discount list should be part of a tabbed interface within the solution module.  
6. The discount list should have a prompt labeled "Discount" that is bold and positioned correctly.  
  
Definition of Done:  
1. The discount options are displayed in a list format.  
2. The list is centered and adheres to the specified dimensions.  
3. The list is positioned correctly on the screen.  
4. The list has the specified background color and font style.  
5. The list is integrated into the tabbed interface of the solution module.  
6. The prompt "Discount" is bold and correctly positioned.  
  
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 table references.

# Automatic Calculation of Benefit and Premium Terms

Type: SOL\_COVERHEAD

Detailed description: As a user, I want the system to automatically calculate and update the benefit term and premium term based on the product definition, pension flag, and booking frequency when I navigate to the next item in the form.  
  
Acceptance criteria:  
1. If the product definition matches specific values or the pension flag is greater than 0, and the insured person's date of birth and vesting age are provided:  
 - Calculate the insured person's age using the date of birth and either the inception date or the effective date.  
 - Set the benefit term as the difference between the vesting age and the insured person's age.  
 - If the product definition matches specific values or the pension flag is greater than 0 and the booking frequency is not '01':  
 - Set the premium term as the difference between the vesting age and the insured person's age.  
 - Otherwise, set the premium term to 0.  
2. If a package is provided:  
 - Navigate to the 'covers' block.  
 - For each record in the 'covers' block, update the benefit term and premium term with the calculated values.  
 - Stop the loop if the cover code is null.  
  
Definition of Done:  
- The system correctly calculates and updates the benefit term and premium term based on the specified conditions.  
- The 'covers' block records are updated with the calculated values if a package is provided.  
- The form status is set to 'Y' when the item is validated and the global loading flag is 'F'.  
  
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.

# Validate and Populate Investment Amount

Type: SOL\_COVERHEAD

User Story: Validate and Populate Investment Amount  
  
Detailed Description:  
As a user, I need to ensure that the "Amount Invested" field is correctly validated and populated based on specific conditions and calculations. This will help in maintaining data integrity and ensuring that all necessary fields are filled out correctly before proceeding to the next steps.  
  
Acceptance Criteria:  
1. If the "Solution Name" field is empty and the "Amount Invested" field is not empty, prompt the user to select a solution name.  
2. If the "Solution Name" is equal to 1, calculate the beneficiary's age using the provided date of birth and current date. If the beneficiary's age is 12 or older, display an error message indicating that the nominee must be a minor with an age less than 12.  
3. Populate the solution product details and navigate to the "SOL\_COVERS" section.  
4. If the cover code in the "SOL\_COVERS" section starts with 'L', set the "Total Rider Invest" field to the value of the "Amount Invested" field and make it read-only.  
5. Navigate to the "Total Rider Invest" field in the "SOL\_COVERS" section.  
  
Definition of Done:  
- The system should prompt the user to select a solution name if it is not provided when the amount invested is entered.  
- The system should calculate the beneficiary's age and display an error if the age is 12 or older.  
- The solution product details should be populated correctly.  
- The "Total Rider Invest" field should be set and made read-only if the cover code starts with 'L'.  
- The user should be navigated to the "Total Rider Invest" field in the "SOL\_COVERS" section.  
  
DB Queries for Table Reference CRUD Operations:  
- Not applicable as the provided logic involves form-specific operations and validations that are not directly executable in the database without modification.

# Process and Update Insurance Cover Records on 'Ok' Button Press

Type: SOL\_COVERHEAD

Detailed description: As a user, I want to ensure that the system correctly processes and updates insurance cover records when I press the "Ok" button, so that the cover details are accurately maintained and any necessary alerts are displayed.  
  
Acceptance criteria:  
1. When the "Ok" button is pressed, the system should navigate to the 'SOL\_COVERS' section and iterate through all records.  
2. For each record in 'SOL\_COVERS':  
 - If the cover code starts with 'R' and is not 'R036A01', the system should calculate and update the 'SUM\_INSURED\_WHOLE\_COVER' field based on specific formulas.  
 - The system should then navigate to the 'COVERS' section and update or delete records based on the 'TOTAL\_RIDER\_INVEST' value.  
 - If the 'TOTAL\_RIDER\_INVEST' value is zero, the record should be deleted.  
3. If the cover code is 'R036A01', the system should prompt the user with an alert asking if they want the Waver of Premium Benefit.  
 - If the user declines, the corresponding record in 'COVERS' should be deleted.  
4. The system should ensure that all records are processed and updated correctly, and the user is navigated back to the appropriate section.  
  
Definition of Done:  
- The "Ok" button functionality is implemented as described.  
- The system correctly processes and updates records in 'SOL\_COVERS' and 'COVERS' sections.  
- Alerts are displayed as per the conditions specified.  
- The user is navigated back to the appropriate section after processing.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided logic involves Oracle Forms-specific constructs and cannot be directly executed in the database without modification.

# Determine Discount Type for Insurance Application

Type: SOL\_COVERHEAD

User Story: Determine Discount Type for Insurance Application  
  
Detailed description:   
As a system, I need to determine the discount type for an insurance application based on various conditions such as agent code, product ID, and other related parameters, so that the appropriate discount type can be applied to the application.  
  
Acceptance criteria:  
1. If the product ID is not in the list (269, 335, 337, 341, 343, 345), check if the agent code matches a specific discount agent code from the system constants table. If a match is found, set the discount agent flag to 'Y'; otherwise, set it to 'N'.  
2. Retrieve the employee code based on the application number from the transaction data table. If not found, set the employee code to NULL.  
3. Retrieve the permanent receipt date from the application cover details table based on the application number and solution ID. If not found, set it to the current date.  
4. Retrieve the IRDA launch date from the system constants table if it is not already set.  
5. Set the discount type to NULL initially.  
6. Check if the agent code matches specific offline/online codes from the system constants table. If a match is found, set the offline/online flag accordingly.  
7. Determine the discount type based on various conditions involving agent code, product ID, web aggregator flag, group employee flag, and offline/online status.  
8. If the product is a unit-linked product, clear and populate the employee list based on the discount type and other conditions.  
9. If the product is not a group product, clear and populate the solution discount list based on the discount type and other conditions.  
  
Definition of Done:  
- The discount type is correctly determined and set based on the specified conditions.  
- The employee list and solution discount list are correctly populated based on the discount type and other conditions.  
- All relevant data is retrieved and processed without errors.  
  
DB queries for Table reference CRUD operations only (With Usage):  
1. Retrieve discount agent flag:  
 ```sql  
 SELECT 'Y'  
 INTO v\_disc\_agnt  
 FROM azbj\_system\_constants  
 WHERE SYS\_TYPE = 'DISCOUNT'  
 AND sys\_CODE = 'DISCOUNT\_AGENT'  
 AND :agents.ag\_agent\_code LIKE '%' || char\_value || '%'  
 AND ROWNUM = 1;  
 ```  
  
2. Retrieve employee code:  
 ```sql  
 SELECT var30  
 INTO pk\_vars.v\_emp\_code  
 FROM websales.azbj\_cq\_trans\_data  
 WHERE application\_no = TO\_CHAR(NVL(:insured\_person.ip\_verf\_no, :insured\_person.ip\_sign\_card\_no));  
 ```  
  
3. Retrieve permanent receipt date:  
 ```sql  
 SELECT C\_RECPT\_NO  
 INTO v\_perm\_rec\_date  
 FROM azbj\_solution\_appln\_cover\_dtls  
 WHERE application\_no = TO\_CHAR(NVL(:insured\_person.ip\_verf\_no, :insured\_person.ip\_sign\_card\_no))  
 AND SOLUTION\_ID = pk\_vars.v\_sol\_product\_id;  
 ```  
  
4. Retrieve IRDA launch date:  
 ```sql  
 SELECT start\_date  
 INTO v\_ver2\_lunch\_date  
 FROM azbj\_system\_constants  
 WHERE sys\_type = 'IRDA' AND sys\_code = 'IRDA\_LAUNCH';  
 ```  
  
5. Retrieve offline/online flag:  
 ```sql  
 SELECT COUNT()  
 INTO v\_instab\_cnt  
 FROM azbj\_system\_constants  
 WHERE sys\_type = 'OFFLINE\_ONLINE'  
 AND sys\_code = 'OFFLINE\_ONLINE'  
 AND :agents.ag\_agent\_code LIKE '%' || char\_value || '%'  
 AND ROWNUM = 1;  
 ```  
  
6. Retrieve solution discount code:  
 ```sql  
 SELECT SYS\_CODE  
 INTO :SOL\_COVERHEAD.CH\_SOL\_DISC  
 FROM azbj\_system\_constants  
 WHERE sys\_type = 'PREM\_DISC'  
 AND char\_value = CASE  
 WHEN pk\_vars.v\_product\_id < 287 AND :SOL\_COVERHEAD.SOL\_DISCOUNT\_TYPE = 'D' THEN 'W'  
 ELSE :SOL\_COVERHEAD.SOL\_DISCOUNT\_TYPE  
 END  
 AND prod\_id = pk\_vars.v\_product\_id  
 AND sys\_desc = CASE  
 WHEN pk\_vars.v\_product\_id = 315 THEN :coverhead.ch\_package\_code  
 WHEN pk\_vars.v\_product\_id IN (339) AND :SOL\_COVERHEAD.SOL\_DISCOUNT\_TYPE = 'S' THEN :coverhead.ch\_package\_code  
 ELSE sys\_desc  
 END  
 AND ROWNUM = 1  
 AND TO\_DATE(pk\_vars.v\_perm\_rec\_date, 'dd/mm/rrrr') >= NVL(TO\_DATE(start\_date, 'dd/mm/rrrr'), TO\_DATE('01/01/2000', 'dd/mm/rrrr'))  
 AND TO\_DATE(pk\_vars.v\_perm\_rec\_date, 'dd/mm/rrrr') <= NVL(TO\_DATE(end\_date, 'dd/mm/rrrr'), TO\_DATE('01/01/3000', 'dd/mm/rrrr'));  
 ```

# Delete Solution Cover

Type: SOL\_COVERHEAD

User Story: Delete Solution Cover  
  
Detailed Description:  
As a user, I want to be able to delete a solution cover from the list of covers associated with a solution product, ensuring that all necessary validations and dependencies are checked before the deletion is performed.  
  
Acceptance Criteria:  
1. When the delete button is pressed, the system should navigate to the 'sol\_covers' section and iterate through all records.  
2. The system should check if the selected cover is a rider (cover code starting with 'R') and if it is compulsory with the selected package. If so, an error message should be displayed, and the deletion should be prevented.  
3. The system should identify and mark specific rider codes (R001A01, R004A01, R005A01, R008A01, R036A01, R036B01, R035A01) for deletion if they are flagged for deletion.  
4. The system should handle the deletion of top-up covers (cover codes starting with 'T') if certain conditions are met, such as the sum insured being zero.  
5. The system should ensure that certain riders are always together (e.g., R001A01 and R005A01 for Unitlink products) and adjust the deletion flags accordingly.  
6. The system should navigate back to the first record and delete the marked records from the 'covers' section.  
7. The system should update the deletion flags and ensure that the deletion process is completed successfully.  
  
Definition of Done:  
- The delete button functionality is implemented and tested.  
- All specified validations and dependencies are checked before deletion.  
- Appropriate error messages are displayed when necessary.  
- The system successfully deletes the marked records and updates the deletion flags.  
- The functionality is tested and verified to work as expected.  
  
DB Queries for Table Reference CRUD Operations:  
- Not applicable as the provided logic is specific to Oracle Forms and cannot be executed directly in the database without modification.  
  
Explanation of Oracle Form Logics:  
- The logic involves navigating through records, checking conditions, and updating flags within the Oracle Forms environment.  
- Specific rider codes and conditions are checked to ensure that the deletion process adheres to business rules.  
- The system uses Oracle Forms triggers and procedures to handle the deletion process and display appropriate messages to the user.

# Manage Insurance Cover Details

Type: COVERS

Title: Manage Insurance Cover Details  
  
Acceptance Criteria:  
1. When a new block instance is created, the system should:  
 - Delete any existing record group named 'THL\_COVER\_RG'.  
 - Create a new record group 'THL\_COVER\_RG' with a query that selects `COVER\_DESCRIPTION` and `COVER\_CODE` from the `CFG\_V\_PROD\_COVERS\_API` table where `PRODUCT\_ID` and `PROD\_VERSION` match the control parameters.  
 - Populate the record group and handle any errors during the population process.  
 - Set the LOV (List of Values) property to use the newly created record group.  
  
2. When a new item instance is created, the system should store the current cursor value in a global variable for later use.  
  
3. When the user presses the 'Up' key:  
 - If the `cv\_cover\_code` field is null, the system should delete the current record.  
 - If the `cv\_cover\_code` field is not null, the system should navigate to the previous record.  
  
4. When the user presses the 'Down' key, the system should navigate to the next record.  
  
Definition of Done:  
- The user can view and interact with insurance cover details within the specified section.  
- The system correctly handles the creation and population of the record group for the LOV.  
- The system stores the cursor value in a global variable when a new item instance is created.  
- The system correctly handles navigation and record deletion based on the `cv\_cover\_code` field value.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- The query to create the record group:  
 ```sql  
 SELECT COVER\_DESCRIPTION, COVER\_CODE   
 INTO :COVERS.CV\_COVER\_DESC, :COVERS.CV\_COVER\_CODE   
 FROM CFG\_V\_PROD\_COVERS\_API   
 WHERE PRODUCT\_ID = :CONTROL.CN\_PRODUCT\_ID   
 AND PROD\_VERSION = :CONTROL.CN\_PRODUCT\_VERSION;  
 ```  
  
- The query for the LOV record group:  
 ```sql  
 SELECT COVER\_DESCRIPTION, COVER\_CODE   
 FROM CFG\_V\_PROD\_COVERS\_API   
 WHERE PRODUCT\_ID = :PARAMETER.PRODUCT\_ID   
 AND PROD\_VERSION = 1;  
 ```

# Interest Rate Field Validation and Toolbar Control

Type: COVERS

Title: Interest Rate Field Validation and Toolbar Control  
  
Acceptance Criteria:  
1. The interest rate field should be validated when the user inputs a value.  
2. During the validation process, the "Commit Form" and "Exit Form" toolbar options should be disabled.  
3. If the global loading flag is set to 'F', the form status control should be updated to 'Y'.  
  
Definition of Done:  
- The interest rate field is validated as per the specified conditions.  
- The "Commit Form" and "Exit Form" toolbar options are disabled during the validation process.  
- The form status control is updated correctly based on the global loading flag.  
  
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 form status on frequency amount validation

Type: COVERS

Title: Update form status on frequency amount validation  
  
Acceptance Criteria:  
- When the frequency amount field is validated, the form status should be set to 'Y'.  
- The frequency amount field should be right-justified, disabled, and have a maximum length of 15 characters.  
- The field should be displayed on the Covers tab page.  
  
Definition of Done:  
- The form status is updated to 'Y' upon validation of the frequency amount field.  
- The frequency amount field meets the specified display and formatting requirements.  
- 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 queries mentioned in the provided XML content.

# Validation and Display Properties for CV\_FREQ\_NRI\_AMT Field

Type: COVERS

Title: Validation and Display Properties for CV\_FREQ\_NRI\_AMT Field  
  
Acceptance Criteria:  
1. When the "CV\_FREQ\_NRI\_AMT" field is validated, the form status should be set to 'Y'.  
2. The field should be disabled and have a maximum length of 15 characters.  
3. The field should be displayed with the prompt "Res Ldg" aligned to the top and right, with specific font and style settings.  
  
Definition of Done:  
1. The "CV\_FREQ\_NRI\_AMT" field is correctly validated and the form status is updated to 'Y'.  
2. The field is disabled and adheres to the specified display properties.  
3. The changes are tested and verified to ensure they meet the acceptance criteria.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include any specific database queries.

# Validate Frequency of Occurrence Amount and Track Changes

Type: COVERS

Title: Validate Frequency of Occurrence Amount and Track Changes  
  
Acceptance Criteria:  
- When the frequency of occurrence amount is modified, the system should set a control status flag to 'Y' to indicate that the form has been changed.  
  
Definition of Done:  
- The frequency of occurrence amount field is validated upon modification.  
- The control status flag is set to 'Y' when the field is modified.  
- The changes are accurately tracked and reflected in the system.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include any specific database queries.

# Validate and Update Premium Term Based on Product ID

Type: COVERS

Title: Validate and Update Premium Term Based on Product ID  
  
Acceptance Criteria:  
1. If the product ID is not in the specified list, and the cursor is on the first record, the system should store the premium term value in a global variable.  
2. The system should disable the "Commit Form" and "Exit Form" toolbar options.  
3. The form status should be updated to 'Y'.  
4. When a new item instance is created, if the product ID is not 343, the premium term should be set to the benefit term.  
  
Definition of Done:  
- The system correctly validates the premium term based on the product ID.  
- The premium term is stored in a global variable when conditions are met.  
- The "Commit Form" and "Exit Form" toolbar options are disabled as specified.  
- The form status is updated to 'Y'.  
- The premium term is set to the benefit term when a new item instance is created, given the product ID condition.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- No direct database queries are provided in the XML content.

# Update form status on field validation

Type: COVERS

Title: Update form status on field validation  
  
Acceptance Criteria:  
- When the user validates the field "CV\_FREQ\_SR\_AMT", the system should set the form status to 'Y'.  
- The field "CV\_FREQ\_SR\_AMT" should be disabled and should not accept any user input.  
- The field should be displayed on the "COVERS" tab page.  
- The field should be right-justified and have a maximum length of 15 characters.  
- The field should be of type "Number" and have a width of 47 units and a height of 18 units.  
  
Definition of Done:  
- The form status is updated to 'Y' upon validation of the "CV\_FREQ\_SR\_AMT" field.  
- The "CV\_FREQ\_SR\_AMT" field is correctly displayed and disabled on the "COVERS" tab page.  
- All specified display properties (justification, length, type, width, height) are correctly implemented.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include any specific database queries.

# Validate Entry Age and Perform Calculations for Child Care Products

Type: COVERS

Title: Validate Entry Age and Perform Calculations for Child Care Products  
  
Acceptance Criteria:  
1. When the entry age is validated:  
 - If the current record is the first record, the system should store the entry age in a global variable and a control variable.  
 - If the product is a Child Care Product (identified by product ID 11 or a specific product definition) and the current record is the second record, the system should store the entry age in a different control variable.  
2. For Child Care Products:  
 - If the current record is the first record, the system should calculate the benefit term and premium term based on the entry age and package code.  
 - The benefit term should be the difference between a specific part of the package code and the entry age.  
 - The premium term should be 18 minus the entry age.  
 - These calculated terms should be stored in the cover and coverhead sections.  
3. The form status should be updated to 'Y' after the validation and calculations.  
  
Definition of Done:  
- The system correctly validates the entry age and performs the necessary calculations based on the product type and record.  
- The calculated benefit term and premium term are accurately stored in the appropriate sections.  
- The form status is updated to 'Y' after the validation and calculations.  
- The functionality is tested and verified to ensure accuracy and reliability.

# Benefit Term Validation and Premium Term Adjustment

Type: COVERS

Detailed description: As a user, I want the system to validate the benefit term input and adjust the premium term based on specific product definitions, ensuring that the form's state and toolbar properties are correctly managed.  
  
Acceptance criteria:  
1. When the benefit term is entered, the system should check if the product ID is not in the specified list of product IDs.  
2. If the product ID is not in the list and the cursor is on the first record, the system should store the benefit term in a global variable.  
3. The system should disable the commit and exit form buttons on the toolbar.  
4. The form status should be set to 'Y'.  
5. When navigating to the next item, if the product definition matches 'ISECURE\_LOAN' or 'ISECLOANV2', the premium term should be calculated as two-thirds of the benefit term, truncated to an integer.  
  
Definition of Done:  
- The benefit term validation and premium term adjustment logic are implemented and tested.  
- The toolbar buttons' properties are correctly managed based on the validation logic.  
- The form status is updated as required.  
- The functionality is tested and verified to work as expected without any Oracle Forms-specific terminology or dependencies.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided logic does not include direct database CRUD operations.

# Calculate Entry Age for Insurance Covers

Type: COVERS

User Story: Calculate Entry Age for Insurance Covers  
  
Detailed description:   
As an insurance system, I need to calculate the entry age for different insurance covers based on the date of birth of the insured person or policyholder, the inception date of the cover, and the effective date of the control. This calculation should consider various conditions such as the type of product, the relationship between the insured person and the policyholder, and specific product requirements.  
  
Acceptance criteria:  
1. The system should calculate the entry age using the date of birth of the insured person or policyholder, the inception date of the cover, and the effective date.  
2. The system should handle different product definitions and adjust the entry age calculation accordingly.  
3. The system should consider specific conditions such as:  
 - If the proposal type is not 'HUF' and the insured person and policyholder are different, and the partner type is 'I' or 'P' and the insured person is 18 or older.  
 - Specific product codes and their associated conditions.  
 - The relationship between the insured person and the policyholder, including joint life products and minor insured persons.  
4. The system should update the entry age, benefit term, and premium term based on the calculated entry age and product-specific rules.  
5. The system should handle exceptions and provide appropriate messages in case of errors.  
  
Definition of Done:  
- The entry age calculation logic is implemented and tested.  
- The system correctly updates the entry age, benefit term, and premium term based on the calculated values.  
- All specified conditions and product-specific rules are handled.  
- The system provides appropriate error messages in case of exceptions.  
- The functionality is reviewed and approved by stakeholders.  
  
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.

# Validate and Calculate Insurance Premiums

Type: COVERS

User Story: Validate and Calculate Insurance Premiums  
  
Detailed Description:  
As an insurance system user, I want the system to validate and calculate the insurance premiums based on various conditions and parameters, so that accurate premium amounts are determined for different insurance products.  
  
Acceptance Criteria:  
1. The system should validate the booking frequency and ensure it is not null. If it is null, an error message should be displayed.  
2. For specific products, the system should call a health coverage calculation procedure.  
3. The system should iterate through policy members and count the number of members with specific relations (e.g., Mother, Father) and age proof types.  
4. The system should calculate the sum assured and premium amounts based on various conditions, including product definitions, booking frequencies, and age proof types.  
5. The system should handle different product IDs and apply specific calculations for each product, including setting standard premiums and frequency premiums.  
6. The system should calculate additional charges such as non-standard age proof loading and service tax based on predefined rules and conditions.  
7. The system should update the total premium payable and ensure it includes all applicable charges and taxes.  
8. The system should handle specific conditions for unit-linked products and calculate charges accordingly.  
9. The system should ensure that all validations and calculations are logged for auditing purposes.  
  
Definition of Done:  
- The system accurately validates and calculates premiums based on the provided conditions.  
- All error messages and validations are displayed correctly.  
- The calculated premiums include all applicable charges and taxes.  
- The system logs all validation and calculation steps for auditing purposes.  
- The user can proceed with the policy issuance only if all validations pass.  
  
DB Queries for Table Reference CRUD Operations:  
- Not applicable as the provided XML content does not include direct database queries that can be executed without modification.  
  
Explanation of Oracle Form Logics:  
- The logic involves validating the booking frequency and ensuring it is not null.  
- It includes iterating through policy members to count specific relations and age proof types.  
- The system calculates premiums based on product definitions, booking frequencies, and age proof types.  
- Additional charges such as non-standard age proof loading and service tax are calculated based on predefined rules.  
- The total premium payable is updated to include all applicable charges and taxes.  
- Specific conditions for unit-linked products are handled, and charges are calculated accordingly.  
- All validation and calculation steps are logged for auditing purposes.

# Manage Insurance Cover Details

Type: SOL\_COVERS

Detailed description: As a user, I want to manage the details of various insurance covers within a solution product, so that I can efficiently handle and display cover-related information.  
  
Acceptance criteria:  
1. The system should allow the user to input and display the base standard premium for each cover.  
2. The system should enable the user to enter and display the cover code and description.  
3. The system should allow the user to input and display the sum insured for the whole cover.  
4. The system should enable the user to enter and display the benefit term and premium term.  
5. The system should allow the user to input and display the entry age for the cover.  
6. The system should enable the user to enter and display various amounts such as annual and frequency amounts for different categories (e.g., standard premium, extra amount, etc.).  
7. The system should allow the user to input and display discount percentages and amounts.  
8. The system should enable the user to enter and display rounding offsets.  
9. The system should allow the user to mark covers for deletion.  
10. The system should enable the user to input and display interest rates.  
11. The system should allow the user to input and display dates (from and to) for the cover.  
12. The system should enable the user to input and display additional amounts and percentages for different categories (e.g., NRI percentage, loading amount, etc.).  
13. The system should allow the user to input and display the total rider sum assured for the solution rider.  
  
Definition of Done:  
1. The user can successfully input and display all required fields for each cover.  
2. The system validates the input data according to the specified criteria (e.g., data types, maximum lengths, etc.).  
3. The user interface is intuitive and user-friendly, allowing easy navigation between fields.  
4. The system handles and displays data accurately without any errors.  
5. The system meets all specified acceptance criteria and passes all test cases.  
  
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.

# Navigation through Cover Description Field

Type: SOL\_COVERS

Title: Navigation through Cover Description Field  
  
Acceptance Criteria:  
1. When the user is on the "Cover Description" field and presses the key to move to the next item:  
 - If the current record is the last record, the focus should move to the confirmation button.  
 - If the current record is not the last record, the focus should move to the next record and then to the "Total Rider Investment" field.  
  
Definition of Done:  
- The navigation logic is implemented and tested.  
- The user can seamlessly move through records and reach the confirmation step as described.  
- The solution is independent of any specific technology or platform.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include any database-specific queries.

# Implement Loading Type Selection Field

Type: SOL\_COVERS

Title: Implement Loading Type Selection Field  
  
Acceptance Criteria:  
- The loading type field should allow the user to select from a list of predefined options.  
- The field should be editable and allow updates to the selected value.  
- The field should have a maximum length of 50 characters.  
- The field should be positioned appropriately on the form for easy access.  
- The field should display a prompt above it indicating "Loading Type".  
  
Definition of Done:  
- The loading type field is implemented and visible on the form.  
- The field allows selection from a predefined list of 5 options.  
- The field can be edited and updated by the user.  
- The field adheres to the specified maximum length and positioning requirements.  
- The prompt "Loading Type" is displayed above the field.  
  
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 table references.

# Disable Commit and Exit buttons during interest rate validation

Type: SOL\_COVERS

Title: Disable Commit and Exit buttons during interest rate validation  
  
Acceptance Criteria:  
1. The "Commit" and "Exit" buttons on the toolbar should be disabled when the interest rate field is being validated.  
2. If the form is not in a loading state, the form status should be updated to indicate that changes have been made.  
  
Definition of Done:  
1. The "Commit" and "Exit" buttons are disabled during the validation of the interest rate field.  
2. The form status is updated correctly when the form is not in a loading state.  
3. The functionality is tested and verified to ensure that the buttons are disabled and the form status is updated as expected.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include any direct database queries.

# Validate SOL\_CV\_FREQ\_NRI\_AMT Field

Type: SOL\_COVERS

Title: Validate SOL\_CV\_FREQ\_NRI\_AMT Field  
  
Acceptance Criteria:  
- When the "SOL\_CV\_FREQ\_NRI\_AMT" field is validated, the form status should be set to 'Y'.  
  
Definition of Done:  
- The "SOL\_CV\_FREQ\_NRI\_AMT" field should be validated to ensure that the form status is updated to 'Y' upon validation.  
- The field should be displayed on the "SOLUTION\_PRODUCT2" tab page.  
- The field should be disabled for user input.  
- The field should have a maximum length of 15 characters and should be of type Number.  
- The field should be positioned correctly on the form as specified.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as there are no direct CRUD operations mentioned in the provided XML content.

# Automatic Navigation for Total Rider Sum Assured Input

Type: SOL\_COVERS

Title: Automatic Navigation for Total Rider Sum Assured Input  
  
Acceptance Criteria:  
1. When the user is on the last record and presses the "Next" or "Down" key, the system should navigate to the "OK" button in the "Solution Coverhead" section.  
2. When the user is not on the last record and presses the "Next" or "Down" key, the system should navigate to the next record and then return to the "Total Rider Sum Assured For Solution Rider" field.  
  
Definition of Done:  
- The navigation logic is implemented and tested.  
- The system correctly identifies the last record and navigates to the "OK" button.  
- The system correctly navigates to the next record and returns to the "Total Rider Sum Assured For Solution Rider" field when not on the last record.  
- User acceptance testing is completed and signed off.  
  
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 Loading Details for Cover Codes

Type: LOADINGS

Title: Manage Loading Details for Cover Codes  
  
Acceptance Criteria:  
1. If the cover code field is empty and the user presses the down arrow key, no action should be taken.  
2. If the cover code field is not empty and the user presses the down arrow key, the system should navigate to the next record.  
3. If the cover code field is empty and the user presses the up arrow key, the current record should be deleted.  
4. If the cover code field is not empty and the user presses the up arrow key, the system should navigate to the previous record.  
  
Definition of Done:  
- The user can view and interact with fields for cover code, cover description, various percentages, amounts, and dates.  
- The system correctly handles navigation and deletion based on the cover code field's value.  
- All fields are displayed with appropriate labels and are enabled or disabled as specified.  
- The user interface is intuitive and matches the specified layout and design.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include specific database queries.

# Validate 'From Date' against Inception Date

Type: LOADINGS

Title: Validate 'From Date' against Inception Date  
  
Acceptance Criteria:  
1. When the user enters a date in the 'From Date' field:  
 - If the entered date is earlier than the inception date and no extra amount is specified, the system should automatically set the 'From Date' to the inception date.  
 - If the entered date is earlier than the inception date and an extra amount is specified, the system should display an error message indicating that the 'From Date' cannot be earlier than the inception date.  
  
Definition of Done:  
- The 'From Date' field is validated against the inception date.  
- Appropriate actions (auto-correction or error message) are taken based on the conditions specified.  
- The feature is tested and verified to work as expected.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided logic is specific to form validation and does not involve direct database CRUD operations.

# Validate Extra Amount Field in Loadings Section

Type: LOADINGS

Detailed description: As a user, I want to ensure that the "Extra Amount" field in the Loadings section only accepts non-negative values, so that data integrity is maintained and negative amounts are not entered.  
  
Acceptance criteria:  
1. When a user enters a value in the "Extra Amount" field, the system should validate that the value is not negative.  
2. If the entered value is negative, the system should display an error message stating "Amount cannot be negative".  
3. The validation should occur immediately after the user attempts to leave the "Extra Amount" field.  
  
Definition of Done:  
1. The "Extra Amount" field should be present in the Loadings section.  
2. The system should validate the value of the "Extra Amount" field to ensure it is non-negative.  
3. An error message should be displayed if a negative value is entered.  
4. The validation logic should be tested and confirmed to work 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 CRUD operations.

# Validate Special Risk Loading (SRL) Percentage

Type: LOADINGS

Title: Validate Special Risk Loading (SRL) Percentage  
  
Acceptance Criteria:  
1. If the cursor is on the SRL percentage field and the cover code starts with 'R', the SRL percentage should be set to 0 if it is greater than 0 and the cover code is not 'R029A01'. An error message should be displayed stating that special loading is not applicable for any of the riders.  
2. If the product definition is 'CAPITAL\_SHIELD' and the SRL percentage is greater than 0, an error message should be displayed stating that the product is not allowed on extra premium.  
3. If the product definition is 'SARAL\_JEEVAN\_BIMA' and the SRL percentage is greater than 4, an error message should be displayed stating that special loading is not allowed greater than 4.  
  
Definition of Done:  
- The SRL percentage field should be validated based on the specified conditions.  
- Appropriate error messages should be displayed when the conditions are not met.  
- The form should prevent navigation to the next item if the validation fails.  
- The SRL percentage should be correctly propagated to other records in the block based on the cover code.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided logic does not include direct database CRUD operations.

# Validate 'To Date' Field in Loading Section

Type: LOADINGS

Title: Validate 'To Date' Field in Loading Section  
  
Acceptance Criteria:  
1. The 'To Date' field should not be less than the 'From Date'.  
2. The 'To Date' field should not exceed the calculated end date, which is determined by adding the benefit term (in months) to the inception date.  
3. If the 'To Date' exceeds the calculated end date and the extra amount is zero, the 'To Date' should be set to the calculated end date.  
4. If the 'To Date' exceeds the calculated end date and the extra amount is not zero, an error message should be displayed indicating that the 'To Date' cannot be greater than the term end date.  
  
Definition of Done:  
- The 'To Date' field validation logic is implemented and tested.  
- Error messages are displayed correctly when validation fails.  
- The 'To Date' field is adjusted correctly when necessary.  
- All acceptance criteria are met and verified through testing.

# Validate and Manage Loading Type for Specific Products

Type: LOADINGS

Title: Validate and Manage Loading Type for Specific Products  
  
Acceptance Criteria:  
1. When the loading type is changed, if the product ID is '277' or '271' and the loading type is not 'PER\_THOUSAND\_SA', the system should not allow the change and should display an appropriate error message.  
2. During item validation, if the product ID is '277' and the loading type is not 'PER\_THOUSAND\_SA', the system should not allow the change and should display an appropriate error message.  
3. When navigating to the next item, the system should:  
 - Store the current loading type if the cover code starts with 'L'.  
 - Apply the stored loading type to records with cover codes starting with 'T' or specific codes ('R024A01', 'R018A01').  
 - Ensure the navigation continues to the next item after processing all records.  
  
Definition of Done:  
- The system correctly validates and manages the loading type based on the specified conditions.  
- Appropriate error messages are displayed when validation fails.  
- The navigation logic for the next item works as expected, applying the correct loading type to relevant records.  
  
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.

# Validate Loading Percentage for Specific Products and Cover Codes

Type: LOADINGS

Detailed description: As a user, I want to ensure that the loading percentage for specific products and cover codes is validated correctly to prevent errors and ensure data integrity.  
  
Acceptance criteria:  
1. If the product definition is one of the specified products ('SWAYAM\_SHAKTI\_SURAKSHA', 'SARVE\_SHAKTI\_SURAKSHA', 'GROUP\_SEVA\_PLAN', 'GROUP\_LEAVE\_ENCASHMENT', 'GROUP\_CREDIT\_PROTECT', 'GROUP\_CREDIT\_PROTECTION\_PLUS', 'NIYAMIT\_SANCHAY\_SURAKSHA', 'NIYAMIT\_SANCHAY\_SURKSHA\_SINGLE') and the cover code does not start with 'L' and the loading percentage is greater than 0, an error message "Please load Main cover" should be displayed.  
2. The form status should be updated to 'Y' after validation.  
3. When navigating to the next item, if the cover code starts with 'L', the loading percentage should be stored in a variable. If the cover code starts with 'T' or is one of the specified codes ('R024A01', 'R018A01'), the loading percentage should be set to the stored value.  
  
Definition of Done:  
- The validation logic for loading percentages is implemented and tested.  
- Error messages are displayed correctly based on the specified conditions.  
- The form status is updated appropriately.  
- The navigation logic for setting the loading percentage based on cover codes is implemented and tested.  
- All acceptance criteria are met, and the functionality is verified through testing.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include any direct database queries.

# Validate Occupation Loading Percentage Based on Product Definition

Type: LOADINGS

Detailed description: As a user, I want to ensure that the system validates the percentage of occupation loading based on the product definition and specific conditions, so that the correct business rules are enforced.  
  
Acceptance criteria:  
1. If the product definition is 'CAPITAL\_SHIELD' and the occupation loading percentage is greater than 0, the system should display an error message: "Product not allowed on extra Premium".  
2. If the product definition is one of the following: 'GROUP\_CREDIT\_PROTECT', 'GROUP\_SURAKSHA', 'SWAYAM\_SHAKTI\_SURAKSHA', 'SARVE\_SHAKTI\_SURAKSHA', 'GROUP\_SEVA\_PLAN', 'GROUP\_LOAN\_PROTECTOR', 'GROUP\_LEAVE\_ENCASHMENT', 'GROUP\_CREDIT\_PROTECTION\_PLUS', 'NIYAMIT\_SANCHAY\_SURAKSHA', 'NIYAMIT\_SANCHAY\_SURKSHA\_SINGLE', and the cover code does not start with 'L' and the occupation loading percentage is greater than 0, the system should display an error message: "Please load Main cover".  
3. If the product definition is 'SARAL\_JEEVAN\_BIMA' and the occupation loading percentage is greater than 4, the system should display an error message: "Occupation Loading Cannot be greater than 4!".  
  
Definition of Done:  
- The system correctly validates the occupation loading percentage based on the specified product definitions and conditions.  
- Appropriate error messages are displayed when the conditions are met.  
- The form status is updated to 'Y' after validation.  
- The system correctly handles the navigation to the next item after validation.  
  
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.

# Medical Loading Percentage Validation

Type: LOADINGS

Title: Medical Loading Percentage Validation  
  
Acceptance Criteria:  
1. If the cover code is 'R003A01' or 'R003B01', the MLP must be one of the following values: 0, 25, 50, 75, 100, or 150. If not, an error message should be displayed: "The Medical Loading Percentage can be 0, 25, 50, 75, 100, 150 for this Rider."  
2. If the cover code starts with 'L' or is 'R002A01', the MLP must be between 0 and 500. If not, an error message should be displayed: "The Medical Loading Percentage can be between 0 and 500."  
3. If the cover code is 'R018A01', the MLP must be between 0 and 100. If not, an error message should be displayed: "The Medical Loading Percentage can be between 0 and 100."  
4. If the product definition is 'SARAL\_JEEVAN\_BIMA' and the MLP is greater than 200, an error message should be displayed: "Medical Loading is not allowed greater than 200!"  
5. If the cover code is 'L066A01', the MLP must be between 0 and 125. If not, an error message should be displayed: "The Medical Loading Percentage can be between 0 and 125."  
6. If the product definition is 'CAPITAL\_SHIELD' and the MLP is greater than 0, an error message should be displayed: "Product not allowed on extra Premium."  
7. If the product definition is one of the following: 'GROUP\_CREDIT\_PROTECT', 'GROUP\_SURAKSHA', 'SWAYAM\_SHAKTI\_SURAKSHA', 'SARVE\_SHAKTI\_SURAKSHA', 'GROUP\_SEVA\_PLAN', 'GROUP\_LOAN\_PROTECTOR', 'GROUP\_LEAVE\_ENCASHMENT', 'GROUP\_CREDIT\_PROTECTION\_PLUS', 'NIYAMIT\_SANCHAY\_SURAKSHA', 'NIYAMIT\_SANCHAY\_SURKSHA\_SINGLE', and the cover code does not start with 'L' and the MLP is greater than 0, an error message should be displayed: "Please load Main cover."  
  
Definition of Done:  
- The validation rules for Medical Loading Percentage are implemented and tested.  
- Error messages are displayed correctly based on the validation rules.  
- The system correctly navigates through records and applies the validation rules.  
- All acceptance criteria are met and verified through testing.

# Occupation Loading Class Validation

Type: LOADINGS

Title: Occupation Loading Class Validation  
  
Acceptance Criteria:  
1. If the cover code is one of 'R001A01', 'R003A01', 'R004A01', 'R020A01', 'R003B01', or 'R047A01' and the occupation loading class is not 1, 1.5, 2, 97, 98, 99, or 99.99, an error message should be displayed: "The Occupation loading class can be 1 or 1.5 or 2.0 for this Rider".  
2. If the cover code starts with 'L' or is one of 'R002A01', 'R006A01', or 'R006B01' and the occupation loading class is not 1, 2, 3, 4, 97, 98, 99, or 99.99, an error message should be displayed: "The Occupation loading class can be 1 or 2 or 3 or 4 for this Rider".  
3. If the cover code is one of 'R005A01', 'R021A01', or 'R046A01' and the occupation loading class is not 1, 2, 3, 97, 98, 99, or 99.99, an error message should be displayed: "The Occupation loading class can be 1 or 2 or 3 for this Rider".  
4. If the occupation loading class is 99.99 and the calculation type is 'D' or 'AZ':  
 - If the user ID does not start with 'P00', a warning message should be displayed: "Hazardous Occupation".  
 - If the user ID starts with 'P00' and the calculation type is 'D', a warning message should be displayed: "Hazardous Occupation, recommend decline".  
 - If the user ID starts with 'P00' and the calculation type is 'AZ', a warning message should be displayed: "Hazardous Occupation, refer to Allianz Re".  
  
Definition of Done:  
- The occupation loading class validation logic is implemented.  
- Appropriate messages are displayed based on the cover code and calculation type.  
- The functionality is tested and verified to ensure it meets the acceptance criteria.  
- The implementation is independent of any specific technology or Oracle Forms terminology.

# Manage Medical Underwriting Tests

Type: MED\_UW

Title: Manage Medical Underwriting Tests  
  
Acceptance Criteria:  
1. The system should allow users to view and manage the following fields for each medical underwriting test:  
 - Test Number  
 - Description  
 - Received (Y/N)  
 - Date Received  
 - Insurance Type  
 - Date Called  
 - ReTest (Y/N)  
 - Test Conducted From  
 - Locked Status  
 - Serial Number  
 - Test Done Date  
 - Partner ID  
 - Medical Test ID  
 - Request Count  
 - Underwriting Reason  
 - Supervisor Comments  
 - Raised By  
 - Medical Type  
 - Request Raised By  
  
2. The system should provide buttons for the following actions:  
 - Add Test  
 - Delete Test  
 - Validate Test  
 - Access Medicals Grid  
  
3. The system should handle navigation between records using the following logic:  
 - On pressing the "Up" key, the system should navigate to the previous record.  
 - On pressing the "Down" key, if the policy holder's equal insurance status is 1, the system should navigate to the next record and set the insurance type to 1. Otherwise, it should simply navigate to the next record.  
  
Definition of Done:  
- The user can view and manage all specified fields for each medical underwriting test.  
- The user can perform the specified actions using the provided buttons.  
- The system correctly handles navigation between records based on the specified logic.  
- The user interface is intuitive and user-friendly, ensuring ease of use for managing medical underwriting tests.

# Validate Date Received Against Opus Date

Type: MED\_UW

Title: Validate Date Received Against Opus Date  
  
Acceptance Criteria:  
- The system should validate the entered date received (MU\_DATE\_RECD) to ensure it is not greater than the Opus Date.  
- If the entered date received is greater than the Opus Date, an error message should be displayed stating, "Date cannot be greater than Opus Date."  
  
Definition of Done:  
- The date received field is validated against the Opus Date.  
- An appropriate error message is displayed if the validation fails.  
- The functionality is tested and confirmed to work as expected.

# Display Locked List Item in MED\_UW Section

Type: MED\_UW

Title: Display Locked List Item in MED\_UW Section  
  
Acceptance Criteria:  
- The list item should be labeled "Locked".  
- The list item should be displayed within the "MED\_UW" section.  
- The list item should be disabled and not editable.  
- The list item should be positioned at the specified coordinates within the section.  
- The list item should have a specific font style and size for its label.  
- The list item should be displayed on a tabbed canvas named "NBTABS".  
  
Definition of Done:  
- The list item labeled "Locked" is visible in the "MED\_UW" section.  
- The list item is disabled and not editable.  
- The list item is correctly positioned and styled as per the requirements.  
- The list item is displayed on the "NBTABS" tabbed canvas within the "New Business" window.

# Access Health Care Guidelines PDF

Type: MED\_UW

Title: Access Health Care Guidelines PDF  
  
Acceptance Criteria:  
1. When the "Medicals Grid" button is clicked, the system should open the PDF document located at the URL `http://10.3.2.133/med\_pdf/Health%20Care%20Goal%20UW%20Gudielines%201.1.pdf` in a new browser window or tab.  
2. If an error occurs while attempting to open the document, an error message should be displayed to the user indicating the issue.  
  
Definition of Done:  
1. The "Medicals Grid" button is visible and enabled on the user interface.  
2. Clicking the button successfully opens the specified PDF document in a new browser window or tab.  
3. Error handling is implemented to display an appropriate error message if the document cannot be opened.  
4. The functionality is tested and verified to work as expected in the target environment.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include any database CRUD operations.

# Ensure Test Retest Logic and Validation

Type: MED\_UW

Title: Ensure Test Retest Logic and Validation  
  
Acceptance Criteria:  
1. When a test is marked as a retest, it should display a message indicating that the status cannot be changed back.  
2. If a test is already marked as a retest, no other test with the same test number and IP type should be allowed to be marked as a retest.  
3. If there is only one test present, the system should display a message indicating that the test cannot be marked as a retest.  
  
Definition of Done:  
1. The functionality to mark a test as a retest is implemented and tested.  
2. Appropriate messages are displayed when a test is marked as a retest and when there are validation errors.  
3. The system ensures that only one test can be marked as a retest at a time.  
4. Unit tests and integration tests are written and passing.  
5. Code is reviewed and approved.  
6. Documentation is updated to reflect the new functionality.  
  
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.

# Interaction with 'Test No' Field

Type: MED\_UW

Title: Interaction with 'Test No' Field  
  
Acceptance Criteria:  
1. When the 'Test No' field is clicked, the focus should move to this field.  
2. Upon entering the 'Test No' field, the system should automatically populate the 'MU\_DATE\_CALLED' field with the current date from the `pme\_api.opus\_date` function.  
3. The 'MU\_DT\_CALLED' field should be set to the current system date (`SYSDATE`) when the 'Test No' field is entered.  
  
Definition of Done:  
- The 'Test No' field should be non-editable and should not allow user input directly.  
- The 'Test No' field should be keyboard navigable.  
- The 'Test No' field should be properly aligned and positioned within the 'MED\_UW' section.  
- The 'Test No' field should have a prompt labeled 'Test No' that is bold and aligned to the left.  
- The 'MU\_DATE\_CALLED' and 'MU\_DT\_CALLED' fields should be updated correctly when the 'Test No' field is entered.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Not applicable as the provided XML content does not include any direct database queries.

# Automatic IP Type Setting and Validation

Type: MED\_UW

Detailed description: As a user, I want the system to automatically set the IP type to '1' when the product ID is within a specific range, and to validate that the IP type is correctly set when the policy holder and insured person are the same, ensuring that the IP type is '1' in such cases.  
  
Acceptance criteria:  
1. When the product ID is 3, 4, 5, 9, or 10, the IP type should be automatically set to '1'.  
2. If the policy holder and insured person are the same (indicated by a specific flag), and the IP type is not '1', a warning message should be displayed indicating that the IP type should be '1' in the relevant tab.  
3. The system should update a specific variable to indicate whether the IP type validation was required.  
  
Definition of Done:  
1. The system correctly sets the IP type to '1' for the specified product IDs.  
2. The system displays a warning message when the policy holder and insured person are the same, and the IP type is not '1'.  
3. The variable indicating the need for IP type validation is correctly updated.  
4. All changes are tested and verified to ensure they meet the acceptance criteria.  
  
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.

# Automatic Update of Date Received Based on Result Received Status

Type: MED\_UW

Title: Automatic Update of Date Received Based on Result Received Status  
  
Acceptance Criteria:  
1. If the "Result Received" field is set to 'Y' (Yes), the "Date Received" field should be populated with the current date if it is not already set.  
2. If the "Result Received" field is set to 'N' (No), the "Date Received" field should be cleared.  
3. The system should also update the global flags `clean\_fmr\_flag` and `clean\_fmr` to 'N' when the "Result Received" field is 'N'.  
  
Definition of Done:  
- The "Date Received" field is correctly updated based on the "Result Received" field status when navigating away from the "Test Done" field.  
- The global flags are updated as specified.  
- The functionality 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 logic does not include direct database CRUD operations.

# Validate Test Conducted From Options

Type: MED\_UW

Title: Validate Test Conducted From Options  
  
Acceptance Criteria:  
1. If the agent code is '2000000761':  
 - If the test number is 'CTMT', 'XRAY', or '42' and the test conducted from option is 'ECPL\_HOME\_VISIT', the option should be cleared, and an error message should be displayed: "Cannot select ECPL Home Visit For This Test."  
 - If the test conducted from option is 'HOME\_VISIT', the option should be cleared, and an error message should be displayed: "Proposal logged through the ECPL. You cannot select Normal Home Visit."  
  
2. If the agent code is not '2000000761':  
 - If the test conducted from option is 'ECPL\_HOME\_VISIT', the option should be cleared, and an error message should be displayed: "Proposal not logged through the ECPL. You cannot select ECPL home visit."  
  
3. If the test conducted from option is 'THYROCARE', the option should be cleared, and an error message should be displayed: "Cannot Select thyrocare, Please click on validate Test."  
  
4. If the test conducted from option is 'HAT', the option should be cleared, and an error message should be displayed: "Cannot Select HAT, Please click on validate Test."  
  
5. If the test conducted from option is 'HOME\_VISIT':  
 - A confirmation alert should be displayed: "Are you sure you want to select Home Visit option for This test?"  
 - If the user cancels the confirmation, the option should be cleared.  
 - If the test number is 'CTMT', the option should be cleared, and an error message should be displayed: "Cannot select Home Visit For This Test."  
  
Definition of Done:  
- The system should validate the test conducted from options based on the specified conditions.  
- Appropriate error messages should be displayed when invalid options are selected.  
- The user should be prompted with a confirmation alert when selecting 'HOME\_VISIT'.  
- The functionality should be tested and verified to ensure it meets the acceptance criteria.

# Validate Medical Type Selection Based on Request Context

Type: MED\_UW

Title: Validate Medical Type Selection Based on Request Context  
  
Acceptance Criteria:  
1. If the request is not system-generated and the user selects the medical type 'REG\_M', display a warning message: "You cannot select regular medical type for user called REQ" and clear the selection.  
2. If the request is system-generated and the user selects any medical type other than 'REG\_M', display a warning message: "You cannot select other than regular medical type for system called REQ" and set the medical type to 'REG\_M'.  
  
Definition of Done:  
- The validation logic for medical type selection is implemented.  
- Appropriate warning messages are displayed based on the context of the request.  
- The medical type selection is cleared or set as required by the validation logic.  
- Unit tests are created to verify the validation logic and message display.  
  
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.

# Handle Receipt Status of Medical Test Results

Type: MED\_UW

Detailed description: As a user, I want to ensure that the system correctly handles the receipt status of medical test results, so that the appropriate actions are taken based on whether the results have been received or not.  
  
Acceptance criteria:  
1. If the test is conducted through a home visit, the system should prompt the user with a confirmation alert.  
2. If the user confirms that the test is conducted through a home visit, the system should proceed; otherwise, it should reset the test source field.  
3. If the receipt status is 'Y' (Yes), the system should:  
 - Set the receipt date to the current date if it is not already set.  
 - Update the receipt date variable to the current date.  
4. If the receipt status is 'N' (No), the system should:  
 - Clear the receipt date.  
 - Reset the receipt date variable to null.  
 - Set the global and agent flags to 'N'.  
5. If the receipt status is 'W' (Waived) or 'NR' (Not Required), the system should:  
 - Call a procedure to handle waived requirements.  
 - Navigate to the next or previous record based on the key pressed.  
6. The system should update the user who raised the medical request if the receipt status is 'Y' or 'N'.  
  
Definition of Done:  
- The system correctly prompts the user for confirmation when the test is conducted through a home visit.  
- The receipt date and related variables are updated correctly based on the receipt status.  
- The global and agent flags are set appropriately when the receipt status is 'N'.  
- The waived requirements procedure is called correctly for 'W' and 'NR' statuses.  
- The user who raised the medical request is updated correctly based on the receipt status.  
- All changes are tested and verified to ensure they meet the acceptance criteria.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include direct database CRUD operations.

# Enable Fields and Set Default Values on 'Add Test' Button Press

Type: MED\_UW

Detailed description: As a user, I want to enable specific fields and set default values when I press the "Add Test" button, so that I can prepare the form for entering new test details.  
  
Acceptance criteria:  
1. When the "Add Test" button is pressed, the following fields should be enabled:  
 - MED\_UW.MU\_TESTNO  
 - MED\_UW.MED\_TYPE  
2. The field MED\_FUR\_REQ.test\_no should be associated with the LOV (List of Values) named "AZBJ\_MED\_TEST\_LOV".  
3. The field MED\_FUR\_REQ.req\_type should be set to 'MED'.  
4. The focus should move to the field MED\_FUR\_REQ.TEST\_NO.  
5. The fields MED\_FUR\_REQ.TEST\_NO and MED\_FUR\_REQ.Test\_Desc should be cleared (set to null).  
6. If any error occurs during this process, an error message should be displayed with the error details.  
  
Definition of Done:  
- The "Add Test" button functionality is implemented as per the acceptance criteria.  
- The specified fields are enabled and set with default values.  
- The LOV is correctly associated with the field MED\_FUR\_REQ.test\_no.  
- The focus is moved to the correct field.  
- Error handling is in place to display appropriate error messages.  
- The functionality is tested and verified to work as expected.  
  
DB queries for Table reference CRUD operations only(With Usage):  
```sql  
SELECT sys\_code med\_code, UPPER(TRIM(sys\_desc)) med\_test  
FROM azbj\_system\_constants  
WHERE sys\_type = 'MED\_TEST'  
 AND NVL(TO\_DATE(:susac.sa\_daterecd, 'DD/MM/RRRR'), pme\_api.opus\_date)  
 BETWEEN start\_date  
 AND NVL(end\_date, TO\_DATE('01-jan-3000', 'dd/mm/rrrr'))  
 AND UPPER(TRIM(sys\_code)) = DECODE(:control.mer\_rights, 'Y', 'MER', UPPER(TRIM(sys\_code)))  
ORDER BY med\_test;  
```  
- This query is used to populate the LOV "AZBJ\_MED\_TEST\_LOV" with medical test codes and descriptions based on specific conditions.

# Validate Test Button Functionality

Type: MED\_UW

Title: Validate Test Button Functionality  
  
Acceptance Criteria:  
1. When the "Validate Test" button is pressed, the system should:  
 - Call the `validate\_test1` procedure with the parameter 'N'.  
 - If the agent code is '2000000761', call the `validate\_ecpl\_home\_visit` procedure.  
 - If the agent code is not '2000000761', iterate through the records in the 'MED\_UW' block and ensure that the `MU\_TEST\_CONDUCTED\_FROM` field is not 'ECPL\_HOME\_VISIT'. If it is, set it to NULL and display a message indicating that the proposal was not logged through ECPL.  
 - Check if the agent code does not start with '511'. If true, and if the product premium multiplied by the booking frequency is greater than or equal to 50,000 or less than 50,000, call the `CHECK\_MEDICAL\_CENTER` procedure if the `cn\_scrutiny\_medicals` field is not 'Y'.  
 - Call the `validate\_homevisit` procedure.  
 - Call the `azbj\_clean\_mer` procedure.  
 - Iterate through the records in the 'MED\_UW' block and ensure that the `MU\_TEST\_DONE` field is not null. If it is, display a message indicating that the test done cannot be blank.  
 - Check if the difference between `MU\_TEST\_DONE` and `MU\_DATE\_CALLED` is less than or equal to 1. If true, set a flag `v\_med\_frd\_ind` to 'Y' and display a warning message indicating that medicals are done within 1 day of calling requirements.  
 - Iterate through the records in the 'panel\_doctor' block and count the number of tests conducted by each doctor within a specific date range. If the count exceeds a predefined limit, display a warning message indicating that the entered doctor is a preferred doctor and to check the genuineness of the medicals.  
  
Definition of Done:  
- The test validation process is triggered when the "Validate Test" button is pressed.  
- All specified conditions and checks are performed as described in the acceptance criteria.  
- Appropriate messages are displayed to the user based on the validation results.  
- The system ensures that all necessary fields and conditions are validated before proceeding.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- The following query is used to count the number of tests conducted by each doctor within a specific date range:  
 ```sql  
 SELECT COUNT(1)  
 INTO v\_tst\_cnt  
 FROM (  
 SELECT doctor\_code, test\_code, date\_recd  
 FROM wip\_azbj\_panel\_doctor\_rep  
 WHERE date\_recd BETWEEN TO\_DATE(V\_fr\_dt, 'DD/MM/RRRR') AND TO\_DATE(V\_to\_dt, 'DD/MM/RRRR')  
 AND doctor\_code = :panel\_doctor.pd\_doc\_code  
 AND test\_code = :panel\_doctor.pd\_test\_code  
 UNION  
 SELECT doctor\_code, test\_code, date\_recd  
 FROM azbj\_panel\_doctor\_rep  
 WHERE date\_recd BETWEEN TO\_DATE(V\_fr\_dt, 'DD/MM/RRRR') AND TO\_DATE(V\_to\_dt, 'DD/MM/RRRR')  
 AND doctor\_code = :panel\_doctor.pd\_doc\_code  
 AND test\_code = :panel\_doctor.pd\_test\_code  
 ) a;  
 ```

# Delete Medical Test Record with Validation

Type: MED\_UW

Title: Delete Medical Test Record with Validation  
  
Acceptance Criteria:  
1. The system should check if the medical test has been locked and if it has been conveyed to a specific partner (e.g., 'THYROCARE'). If so, the deletion should be prevented, and an appropriate error message should be displayed.  
2. If the test has been conducted by specific partners (e.g., 'THYROCARE', 'HAT') and is not locked, the system should clear the 'Test Conducted From' field for all records in the block.  
3. The system should iterate through all records in the block to check if there are any re-tests present for the current test. If a re-test is found, the deletion should be prevented, and an appropriate error message should be displayed.  
4. The system should log the deletion action, including details such as contract ID, test number, IP type, user, and date.  
5. The system should delete the record if all conditions are met.  
  
Definition of Done:  
- The user can attempt to delete a medical test record.  
- The system performs all necessary checks and validations.  
- Appropriate error messages are displayed if any conditions prevent the deletion.  
- The record is deleted if all conditions are met.  
- The deletion action is logged with relevant details.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Select count() Into v\_cnt From azbj\_medical\_test\_doc\_details Where (partner\_id=:MED\_UW.MU\_PARTNER\_ID or appln\_no=to\_char(NVL(:INSURED\_PERSON.IP\_VERF\_NO,:INSURED\_PERSON.IP\_SIGN\_CARD\_NO))) And MED\_TEST\_ID=:MED\_UW.MU\_MED\_TEST\_ID And nvl(locked,'N') ='Y' And TEST\_CONDUCTED\_FROM ='THYROCARE';  
- For J in(Select From azbj\_scrutiny\_test\_dtl Where APPLICATION\_NO=to\_char(NVL(:INSURED\_PERSON.IP\_VERF\_NO,:INSURED\_PERSON.IP\_SIGN\_CARD\_NO))) Loop If nvl(:med\_uw.mu\_testno,'N')=J.TEST\_NO and :med\_uw.mu\_ip\_type = j.LIFE then azbj\_message('Q','It is verified with Branch & Doctor that this medical is not done'); End if; End loop;  
- Select count() into v\_incomplete\_flg from azbj\_phub\_add\_req\_tracker a, azbj\_phub\_req\_tracker b where a.APPLICATION\_NO=b.APPLICATION\_NO and a.REQ\_TYPE=b.REQ\_TYPE and a.application\_no=to\_char(nvl(:INSURED\_PERSON.IP\_VERF\_NO,:INSURED\_PERSON.IP\_SIGN\_CARD\_NO)) and b.RECD\_IN\_OPUS='N';

# Manage and View Insurance Policy Details

Type: COVERHEAD

Detailed description: As a user, I want to manage and view various insurance policy details, including sum assured, loan number, inception date, premium amount, and other related information, so that I can effectively handle and process insurance policies.  
  
Acceptance criteria:  
1. The system should allow the user to input and view the sum assured, loan number, inception date, premium amount, and other related details.  
2. The system should initialize the inception date and date of risk based on the effective date parameter.  
3. The system should calculate and display the total premium payable, including any applicable service tax, education cess, and higher education cess.  
4. The system should handle different types of insurance policies, including those with premium finance, non-standard premiums, and various loading factors.  
5. The system should validate the presence of required fields and display appropriate error messages if any mandatory information is missing.  
6. The system should support different types of input fields, such as text items, list items, checkboxes, and push buttons, to capture various policy details.  
7. The system should allow the user to perform actions like validating, rating, deleting covers, and submitting or canceling offers.  
  
Definition of Done:  
1. The user can successfully input and view all required insurance policy details.  
2. The system correctly initializes and calculates values based on the provided parameters.  
3. All validation checks are in place, and appropriate error messages are displayed for missing or incorrect information.  
4. The user can perform all necessary actions related to managing insurance policies, including validation, rating, and submission.  
5. The system handles different types of insurance policies and loading factors as specified.  
6. The user interface is intuitive and supports various input types for capturing policy details.  
7. The functionality is tested and verified to ensure accuracy and reliability.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- The system should execute the following query to fetch reasons for counter offers:  
 ```sql  
 SELECT SYS\_DESC FROM AZBJ\_SYSTEM\_CONSTANTS WHERE SYS\_TYPE='REASONS';  
 ```  
- The system should execute the following query to fetch package details based on product ID and receipt date:  
 ```sql  
 SELECT FROM azbj\_package\_master   
 WHERE product\_id=:cn\_product\_id   
 AND :control.CN\_FIRST\_RCPT\_DATE >= NVL(start\_date, TO\_DATE('01/09/2001', 'dd/mm/yyyy'))   
 AND :control.CN\_FIRST\_RCPT\_DATE < NVL(end\_date, TO\_DATE('01/01/3000', 'dd/mm/yyyy'));  
 ```

# Automatic Calculation of Benefit Term and Premium Term Based on Vesting Age and Date of Birth

Type: COVERHEAD

Detailed description: As a user, I want the system to automatically calculate the benefit term and premium term based on the vesting age and date of birth of the insured person, so that I can ensure accurate policy details are maintained.  
  
Acceptance criteria:  
1. If the product ID matches specific values or the pension flag is greater than 0, and the insured person's date of birth and vesting age are provided:  
 - Calculate the insured person's entry age using the date of birth and either the inception date or the effective date.  
 - Set the benefit term as the difference between the vesting age and the entry age.  
 - If the product ID matches specific values or the pension flag is greater than 0 and the booking frequency is not '01', set the premium term as the difference between the vesting age and the entry age. Otherwise, set the premium term to 0.  
 - If the package is not null, iterate through the covers and set the benefit term and premium term for each cover.  
  
2. When the vesting age is validated and the form is not in loading state, set the form status to 'Y'.  
  
Definition of Done:  
- The system correctly calculates and sets the benefit term and premium term based on the provided conditions.  
- The form status is updated to 'Y' when the vesting age is validated and the form is not in loading state.  
- The functionality is tested and verified to ensure accuracy and reliability.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided logic involves calculations and conditional checks rather than direct CRUD operations on the database.

# Dynamic Field Enable/Disable Based on Checkbox Selection

Type: COVERHEAD

Title: Dynamic Field Enable/Disable Based on Checkbox Selection  
  
Acceptance Criteria:  
1. When the checkbox is checked:  
 - The field "CH\_PACKAGE" should be enabled.  
 - The system should navigate to the "CH\_PACKAGE" field.  
 - The fields "cv\_cover\_code", "cv\_cover\_desc", "cv\_sum\_insured\_whole\_cover", "cv\_benefit\_term", and "cv\_premium\_term" should be disabled.  
  
2. When the checkbox is unchecked:  
 - The "CH\_PACKAGE" field should be cleared and disabled.  
 - The system should clear the data in the "COVERS", "LOADINGS", and "COVERHEAD" sections.  
 - The fields "cv\_cover\_code", "cv\_cover\_desc", "cv\_sum\_insured\_whole\_cover", "cv\_benefit\_term", and "cv\_premium\_term" should be enabled.  
  
Definition of Done:  
- The form dynamically enables or disables fields based on the checkbox state.  
- The system navigates to the appropriate fields and clears data as specified.  
- All acceptance criteria are met and tested successfully.

# Automatic Sum Assured Calculation Based on Premium Amount and Booking Frequency

Type: COVERHEAD

Detailed description: As a user, I want the system to automatically calculate the sum assured based on the premium amount and booking frequency when entering the premium amount for specific products, so that I can ensure accurate and consistent data entry.  
  
Acceptance criteria:  
1. If the product ID is not one of the following: 'SWAYAM\_SHAKTI\_SURAKSHA', 'SARVE\_SHAKTI\_SURAKSHA', 'GROUP\_SEVA\_PLAN', 'GROUP\_LEAVE\_ENCASHMENT', 'BIMADHAN\_SURAKSHA', 'BIMA\_SANCHAY', 'GUARANTEED\_SAVINGS', the system should navigate to the next item.  
2. If the product ID is 343 and the booking frequency is not '01', the sum assured should be calculated as the greater of:  
 - The product of the multiplier, premium amount, and booking frequency.  
 - 105% of the product of the premium amount and booking frequency.  
3. If the product ID is 343 and the booking frequency is '01', the sum assured should be calculated as the greater of:  
 - The product of the multiplier and premium amount.  
 - 105% of the premium amount.  
4. After the calculation, the system should navigate to the next item.  
  
Definition of Done:  
- The system correctly calculates the sum assured based on the given conditions.  
- The system navigates to the appropriate next item after the calculation.  
- The feature is tested and verified to ensure accuracy and consistency in data entry.  
  
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.

# Validate Date of Risk Field

Type: COVERHEAD

Title: Validate Date of Risk Field  
  
Acceptance Criteria:  
1. The "Date of Risk" must not be earlier than the "Date of Commencement".  
2. The "Date of Risk" must not be later than the current OPUS date.  
3. For products with ID 14, the "Date of Risk" must not be earlier than May 1, 2003.  
4. If the form is not in a loading state, the form status should be set to 'Y'.  
  
Definition of Done:  
- The "Date of Risk" field is disabled by default.  
- The field should display a date in the format "dd/mm/yyyy".  
- The validation rules are triggered when the user attempts to move to the next item.  
- Appropriate error messages are displayed if any of the validation rules are violated.  
- The form status is updated correctly based on the loading state.  
  
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.

# Automatic Age Proof Type Determination

Type: COVERHEAD

Title: Automatic Age Proof Type Determination  
  
Acceptance Criteria:  
1. When the age proof details are provided, the system should call a function to determine the type of age proof.  
2. If the function returns 'N', the system should display 'Non-standard Age Proof'.  
3. If the function returns 'Y', the system should display 'Standard Age Proof'.  
4. If an error occurs during the process, an error message should be displayed to the user.  
  
Definition of Done:  
- The system correctly identifies and displays the age proof type based on the provided details.  
- The error handling mechanism is in place to notify the user in case of any issues.  
- The functionality is tested and verified to ensure accuracy and reliability.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided logic does not include direct CRUD operations on the database.

# Dynamic Payment Method Update Based on Booking Frequency and Agent Code

Type: COVERHEAD

Title: Dynamic Payment Method Update Based on Booking Frequency and Agent Code  
  
Acceptance Criteria:  
1. When the booking frequency is set to '4' (Quarterly) or '12' (Monthly), or if the booking frequency is '2' (Half-yearly) and the agent code is not '2000003024', '7000005464', '7000002372', or '7000000702', the available payment methods should be:  
 - NACH  
 - Auto Debit Instruction (ADI)  
 - EMANDATE  
 - UPI\_M  
 - SSS  
  
2. For all other booking frequencies and agent codes, the available payment methods should be:  
 - Cash/Cheque/DD  
 - SSS  
 - Direct Debit (DD)  
 - ECS  
 - Credit Card - Standing instruction (CCSI)  
 - Group Cash  
 - Group Cheque  
 - Group Direct Credit (GDC)  
 - Bill Junction - ECS (BJECS)  
 - Auto Debit Instruction (ADI)  
 - Others  
 - BG  
 - NACH  
 - EMANDATE  
  
3. If the booking frequency is '12' (Monthly) and the product ID is '199', an error message should be displayed stating "Monthly Mode is not allowed for this product".  
  
4. If the booking frequency is '12' (Monthly) and the product ID is '10', and the receipt date is before '01-Aug-2003' and the OPUS date is on or after '31-Aug-2003', an error message should be displayed stating "Monthly Mode is not available for Risk and Term Care Products".  
  
5. If the product ID is '14' or '16' and the booking frequency is '01' (Single Premium), an error message should be displayed stating "Single Premium mode is not available for this product".  
  
6. If the product is identified as 'INVESTPLUS' or 'INVESTPLUS\_PREM', the system should navigate to the 'CH\_PREM\_AMT' field. Otherwise, it should navigate to the 'CH\_PACKAGE' field.  
  
Definition of Done:  
- The system correctly updates the payment methods based on the booking frequency and agent code.  
- Appropriate error messages are displayed for invalid booking frequency and product ID combinations.  
- The system navigates to the correct field based on the product type.  
- All functionalities are tested and verified to work as expected.

# Manage Discount Status for Policy

Type: COVERHEAD

Title: Manage Discount Status for Policy  
  
Acceptance Criteria:  
1. When the discount status is changed, if the policy frequency premium is not null, a warning message should be displayed indicating that the policy needs to be rated again.  
2. The commit form button should be disabled if the policy frequency premium is not null.  
3. If the product definition for the current product ID is 'CASH\_RICH', an alert should be shown asking if the user wants to give a staff discount.  
4. If the user selects 'No' in the alert, the discount status should be set to null.  
5. If the user selects 'Yes' in the alert, the discount status should be set to 'C'.  
6. The form status should be updated to 'Y' after the discount status is changed.  
  
Definition of Done:  
- The discount status management functionality is implemented and tested.  
- The warning message and alert are displayed correctly based on the specified conditions.  
- The commit form button is disabled when required.  
- The discount status is updated correctly based on user input.  
- The form status is updated to 'Y' after the discount status is changed.  
- All acceptance criteria are met and verified through testing.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include any direct database queries.

# Close Loading Window and Navigate to COVERS

Type: COVERHEAD

Title: Close Loading Window and Navigate to COVERS  
  
Acceptance Criteria:  
1. When the 'Close' button is pressed, the system should execute the function to populate riders.  
2. The system should then navigate to the 'COVERS' section.  
3. The system should focus on a specific item within the 'COVERS' section.  
4. The loading window should be hidden after the above actions are completed.  
  
Definition of Done:  
- The 'Close' button successfully triggers the function to populate riders.  
- The system navigates to the 'COVERS' section and focuses on the specified item.  
- The loading window is hidden after the actions are completed.  
- All actions are performed seamlessly without any errors or interruptions.  
  
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.

# Automatic Calculation and Validation of Sum Assured

Type: COVERHEAD

Title: Automatic Calculation and Validation of Sum Assured  
  
Acceptance Criteria:  
1. The system should enable or disable a tab page based on the product ID and the sum assured value.  
2. The system should calculate the entry age based on the date of birth and inception date or effective date.  
3. If a package is selected, the system should iterate through records to calculate the sum insured for each cover based on the product ID and cover code.  
4. The system should apply multipliers for top-up calculations based on product ID and age.  
5. The system should handle exceptions and set default multipliers if no data is found.  
6. The system should calculate the sum insured for whole cover based on various conditions and product definitions.  
7. The system should navigate to the next item based on the product ID and other conditions.  
8. The system should calculate the gross monthly income (GMI) for specific product IDs.  
9. The system should adjust the sum assured based on booking frequency and other parameters for specific product IDs.  
  
Definition of Done:  
1. The system correctly enables or disables the tab page based on the product ID and sum assured.  
2. The entry age is accurately calculated and used in further calculations.  
3. The sum insured for each cover is correctly calculated based on the product ID and cover code.  
4. Top-up multipliers are applied correctly, and default values are used when necessary.  
5. The sum insured for whole cover is accurately calculated based on various conditions.  
6. The system navigates to the appropriate next item based on the product ID and other conditions.  
7. The GMI is correctly calculated for specific product IDs.  
8. The sum assured is adjusted correctly based on booking frequency and other parameters for specific product IDs.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- No specific DB queries provided in the XML content that can be executed directly in the database without modification.

# Implement Loading Button Functionality

Type: COVERHEAD

Detailed description: As a user, I want to ensure that when I press the "Loading" button, the system will copy cover details from the 'covers' section to the 'loadings' section, and default certain date fields if they are not already populated.  
  
Acceptance criteria:  
1. When the "Loading" button is pressed, the system should:  
 - Navigate to the 'covers' section and determine the number of records.  
 - Navigate to the 'loadings' section and iterate through each record.  
 - For each record in the 'loadings' section, copy the cover code and cover description from the corresponding record in the 'covers' section.  
 - If the 'from date' in the 'loadings' section is null, it should be set to the inception date from the 'coverhead' section.  
 - If the 'to date' in the 'loadings' section is null, it should be set to the inception date from the 'coverhead' section plus the benefit term in months from the 'covers' section.  
 - Continue this process for all records, ensuring that the navigation between 'covers' and 'loadings' sections is handled correctly.  
  
Definition of Done:  
- The "Loading" button functionality is implemented as described.  
- The system correctly copies cover details and defaults date fields as specified.  
- The navigation between 'covers' and 'loadings' sections works seamlessly.  
- The feature is tested and verified to ensure it meets the acceptance criteria.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided logic involves Oracle Forms-specific constructs and cannot be directly executed in the database without modification.

# Validate Inception Date for Insurance Policy

Type: COVERHEAD

User Story: Validate Inception Date for Insurance Policy  
  
Detailed Description:  
As a user, I need to ensure that the inception date of an insurance policy is correctly validated and updated across related records, so that the policy adheres to business rules and regulatory requirements.  
  
Acceptance Criteria:  
1. The inception date should be initialized with the effective date parameter.  
2. When the user navigates to the next item, the system should:  
 - Check if the extra amount in the loadings block is zero.  
 - If zero, update the from and to dates in the loadings block based on the inception date and benefit term.  
 - Validate the inception date against the allowed date for the product and display an error message if it is less than the allowed date.  
 - Ensure the inception date is not greater than the date of risk.  
3. When the item is validated:  
 - Ensure the inception date is not null.  
 - Validate the inception date against the allowed date for the product and display an error message if it is less than the allowed date.  
 - Ensure the inception date is not greater than the 28th of the month. If it is, adjust it to the 28th and ensure it is not less than the effective date.  
 - Disable certain form buttons if the frequency of premium is not null.  
 - Set the form status to 'Y'.  
  
Definition of Done:  
- The inception date is correctly initialized and validated according to the specified business rules.  
- Appropriate error messages are displayed for invalid dates.  
- Related records in the loadings block are updated based on the inception date.  
- Form buttons are enabled/disabled based on the frequency of premium.  
- The form status is updated correctly.  
  
DB Queries for Table Reference CRUD Operations:  
- The following SQL queries are used to validate and update the inception date:  
 ```sql  
 SELECT COUNT() INTO v\_cnt FROM azbj\_system\_constants  
 WHERE sys\_type = 'GROUP\_NB'  
 AND sys\_code = 'COMMENT RECEIPT VALIDATION'  
 AND sys\_desc = TO\_CHAR(:control.cn\_product\_id);  
  
 SELECT COUNT() INTO v\_cnt FROM azbj\_system\_constants  
 WHERE sys\_type = 'GROUP\_NB'  
 AND sys\_code = 'COMMENT RECEIPT VALIDATION'  
 AND sys\_desc LIKE '%' || TO\_CHAR(:control.cn\_product\_id) || '%';  
 ```

# Handle Visibility and Value Assignment Based on Product ID and Package Code

Type: COVERHEAD

Detailed description: As a user, I want the system to handle the visibility and value assignment of various fields based on the product ID and package code, so that the correct information is displayed and processed according to the business rules.  
  
Acceptance criteria:  
1. When a user double-clicks on the package field:  
 - If the product ID is in the set [91, 269, 335, 337, 341] and the cover code is not in ['L057F01', 'L151B01', 'L151E01'], the net purchase price field should be visible, and the sum assured and sum insured for the whole cover should be set to 0.  
 - If the product ID is in the set [289, 291, 311] and the package contains 'SINGLE', or the product ID is 317, or the product ID is 321 and the package code contains 'SING', the premium term and premium term for the cover should be set to 0.  
 - Otherwise, the net purchase price field should be hidden.  
  
2. When the package field value changes:  
 - If the package contains 'SINGLE' and the product ID is not in the set [329, 343, 299, 319, 309, 331, 241, 316, 321, 351, 353, 277, 283, 301, 307, 287, 271, 259], or the product ID is in the set [247, 317], or the product ID is 321 and the package code contains 'SING', or the product ID is 343 and the package code contains '\_CREATION\_JOINT', the premium term and premium term for the cover should be set to 0.  
 - If the product ID is not in the set [12, 4, 227, 249, 241, 253, 277, 271, 279, 283, 287, 291, 297, 315, 329, 339, 343, 345, 321, 351, 359, 319, 331, 265, 301, 316, 353, 309, 259, 365, 311, 317, 335], and the product is not a unit-linked product, and the product definition is not in ['GROUP\_CREDIT\_PROTECT', 'GROUP\_SEVA\_PLAN', 'GROUP\_LOAN\_PROTECTOR', 'ISECURE\_LOAN', 'GROUP\_CREDIT\_PROTECTION\_PLUS', 'ISECLOANV2', 'INCOME\_SINGLE\_LIFE'], and the solution name is 0, the premium term should be set to the benefit term.  
 - If the product ID is 167, the premium term should be set to the benefit term.  
 - If the product ID is in the set [91, 269, 335, 337, 341] and the cover code is not in ['L057F01', 'L151B01', 'L151E01'], the net purchase price field should be visible, and the sum assured and sum insured for the whole cover should be set to 0. Otherwise, the net purchase price field should be hidden.  
 - If the product ID is 343 and the package code contains 'STEP\_UP\_', the deferral period should be set to 0, and the income period should be set to 20.  
 - If the product ID is 343 and the package code is 'INCOME\_SINGLE\_LIFE', the income period should be set to the benefit term minus the premium term minus the deferral period.  
 - If the product ID is 343 and the package code contains 'WEALTH\_CREATION\_', the deferral period should be set to 0.  
 - If the product ID is 291 and the package contains 'SINGLE', the booking frequency should be set to '01'. If the package contains 'REGULAR', the booking frequency should be set to '1'.  
 - If the product ID is 299, the premium term should be set to the benefit term minus 12, and the premium term field should be disabled for both insert and update operations.  
 - If the product ID is in the set [315, 331], the premium term field should be enabled and allowed for both insert and update operations.  
  
Definition of Done:  
- The system correctly handles the visibility and value assignment of fields based on the product ID and package code.  
- All specified conditions and business rules are implemented and tested.  
- The functionality is verified through unit tests and user acceptance tests.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- The LOV query for selecting packages:  
 ```sql  
 SELECT FROM azbj\_package\_master   
 WHERE product\_id = :cn\_product\_id   
 AND :control.CN\_FIRST\_RCPT\_DATE >= NVL(start\_date, TO\_DATE('01/09/2001', 'dd/mm/yyyy'))   
 AND :control.CN\_FIRST\_RCPT\_DATE < NVL(end\_date, TO\_DATE('01/01/3000', 'dd/mm/yyyy'))  
 ```

# Validate and Update Insurance Coverage Details

Type: COVERHEAD

User Story: Validate and Update Insurance Coverage Details  
  
Detailed Description:  
As an insurance system user, I need the system to validate and update insurance coverage details when changes are made to the coverage type or package, so that the correct coverage information is maintained and any discrepancies are flagged for review.  
  
Acceptance Criteria:  
1. If the selected package code is 'SAFE GAIN - SINGLE' and the coverage amount is not zero, the system should display a warning message indicating that only a single benefit is allowed and reset the coverage amount to zero.  
2. If the package code is null, the system should prompt the user to select a package.  
3. If the package code is not null, the system should call a procedure to update the benefit type.  
4. If the frequency of premium is not null, the system should disable certain form actions to prevent further changes.  
5. If the package code is not null, the system should iterate through the records in the 'covers' block and update the properties of the 'CV\_SUM\_INSURED\_WHOLE\_COVER' field based on the product ID and other conditions.  
  
Definition of Done:  
- The system correctly validates the package code and coverage amount.  
- Appropriate messages are displayed to the user based on the validation results.  
- The system updates the benefit type and coverage details as per the defined logic.  
- Certain form actions are disabled when the frequency of premium is not null.  
- The 'CV\_SUM\_INSURED\_WHOLE\_COVER' field properties are updated correctly based on the product ID and other conditions.  
  
DB Queries for Table Reference CRUD Operations:  
- Not applicable as the provided XML content does not include direct database queries that can be executed independently of Oracle Forms constructs.  
  
Explanation of Oracle Form Logic:  
- The logic checks the package code and coverage amount to ensure compliance with business rules.  
- It prompts the user to select a package if none is selected.  
- It updates the benefit type and coverage details based on the selected package.  
- It disables certain form actions to prevent further changes when the frequency of premium is not null.  
- It iterates through records in the 'covers' block to update field properties based on specific conditions.

# Adjustment from ICC/STM Deposit Confirmation

Type: COVERHEAD

Title: Adjustment from ICC/STM Deposit Confirmation  
  
Acceptance Criteria:  
1. When the user checks the "Adjustment from ICC/STM Deposit?" checkbox, the system should display a confirmation message asking, "Please Confirm If You Want Adjustment from ICC/STM Deposit?".  
2. If the user confirms the adjustment, the checkbox remains checked.  
3. If the user does not confirm the adjustment, the checkbox should be unchecked.  
4. If the "Adjustment from ICC/STM Deposit?" checkbox is unchecked, the related field "XCHG\_ADJ" should also be set to 'N'.  
  
Definition of Done:  
- The checkbox for "Adjustment from ICC/STM Deposit?" is functional and prompts the user for confirmation.  
- The system correctly updates the checkbox state based on user confirmation.  
- The related field "XCHG\_ADJ" is updated to 'N' when the "Adjustment from ICC/STM Deposit?" checkbox is unchecked.  
- All changes are tested and verified to ensure they meet the acceptance criteria.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Not applicable as the provided XML content does not include any direct database operations.

# Premium and Charges Calculation for Policies

Type: COVERHEAD

Title: Premium and Charges Calculation for Policies  
  
Acceptance Criteria:  
1. The system should check if the booking frequency is null and display an error message if it is.  
2. If the product definition is 'CARE\_FIRST', the system should call a specific procedure to handle health covers.  
3. The system should loop through policy members and check their relationship and age proof type, incrementing a counter if certain conditions are met.  
4. The system should calculate the standard premium and frequency premium based on the product ID and cover code.  
5. The system should handle different product IDs and calculate the sum assured rate (SAR) based on various conditions, including the booking frequency and benefit term.  
6. The system should calculate the non-standard age proof loading based on the package code and entry age.  
7. The system should handle GST calculations and apply service tax if applicable.  
8. The system should ensure that the total premium payable is calculated correctly, including any service tax, education cess, and higher education cess.  
9. The system should validate if the allocated premium is less than the frequency charges when loadings are present and display a warning message if necessary.  
10. The system should handle specific validations for 'FAMILY\_CARE\_FIRST' product and incomplete inwarded applications.  
  
Definition of Done:  
- The premium and associated charges are calculated accurately based on the provided conditions and parameters.  
- Appropriate error and warning messages are displayed as per the business rules.  
- The system handles different product IDs and package codes correctly.  
- The system applies GST and service tax calculations where applicable.  
- The total premium payable is calculated and displayed correctly.  
- All validations and checks are performed as per the acceptance criteria.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- The system should execute the following queries to fetch and update data:  
 - Fetch the receipt date from `azbj\_batch\_items` table.  
 - Fetch the multiplier from `azbj\_tasa\_suc\_multiplier` table.  
 - Fetch the date value from `azbj\_system\_constants` table.  
 - Fetch the count of records from `azbj\_supervisor\_appr\_det` table.  
 - Fetch the non-standard age proof from `azbj\_master\_policy\_extn` table.  
 - Fetch the value consider from `azbj\_grp\_calc\_assumptions` table.  
 - Fetch the GST amount and details from `azbj\_pk\_gst` package.  
 - Fetch the minimum admin charge from `azbj\_grp\_pol\_charge` table.  
 - Fetch the allocation charge from `azbj\_grp\_upload\_policy\_pkg\_Ext` package.  
  
Note: The above queries should be executed directly in the database without modification because of the use of Oracle Forms-specific constructs like `:GLOBAL.PARA` and `:MODE`.

# Validate Premium Term and Benefit Term Consistency

Type: COVERHEAD

User Story: Validate Premium Term and Benefit Term Consistency  
  
Detailed Description:  
As a user, I want to ensure that the premium term and benefit term are consistent for specific products and packages, so that the policy terms are correctly aligned and compliant with business rules.  
  
Acceptance Criteria:  
1. If the product ID is 3 and the package code does not contain 'SINGLE', or if the product ID is 5, 14, or 16:  
 - The premium term must be the same as the benefit term. If they are not the same, an error message should be displayed: "Premium Term should be same as Benefit Term for this product".  
2. The premium term cannot be greater than the benefit term. If it is, an error message should be displayed: "Premium Term cannot be more than benefit term".  
3. If the package is not null:  
 - Iterate through all records in the 'covers' block.  
 - For specific cover codes ('R004A01', or if the product ID is 4 and the cover code starts with 'R', or if the product ID is 331 and the cover code starts with 'R048'):  
 - Set the benefit term and premium term in the 'covers' block to the premium term from the 'coverhead' block.  
 - For other cover codes:  
 - Set the benefit term in the 'covers' block to the benefit term from the 'coverhead' block.  
 - Set the premium term in the 'covers' block to the premium term from the 'coverhead' block.  
 - If a cover code is null, exit the loop.  
4. Move the cursor to the 'coverhead.ch\_booking\_frequency' item.  
  
Definition of Done:  
- The system correctly validates and enforces the consistency between premium term and benefit term based on the specified product IDs and package codes.  
- Appropriate error messages are displayed when the terms are not consistent.  
- The terms in the 'covers' block are updated correctly based on the logic provided.  
- The cursor moves to the 'coverhead.ch\_booking\_frequency' item after validation.  
  
DB Queries for Table Reference CRUD Operations:  
- Not applicable as the provided logic does not include direct database CRUD operations.

# Calculate and Display Premium Terms for Loan Protection Products

Type: COVERHEAD

User Story: Calculate and Display Premium Terms for Loan Protection Products  
  
Detailed description:   
As a user, I need the system to automatically calculate and display the premium terms for various loan protection products based on the product definition, cover code, and benefit term. This functionality should ensure that the premium terms are accurately derived and displayed for different product types, including handling exceptions and specific conditions for certain products.  
  
Acceptance criteria:  
1. The system should calculate the premium term (`ch\_prem\_term`) based on the product definition and cover code.  
2. If the product is 'LOAN\_PROTEC' and the package is not null, the system should:  
 - Retrieve the premium term from the `azbj\_mortgage\_term` table using the product ID, cover code, and benefit term.  
 - Set the `ch\_prem\_term` and `cv\_premium\_term` to the retrieved value.  
 - If no data is found, set both terms to 0 and display an error message.  
3. For specific products like 'GROUP\_CREDIT\_PROTECT', 'GROUP\_LOAN\_PROTECTOR', and 'GROUP\_CREDIT\_PROTECTION\_PLUS', if the package does not contain 'SINGLE' and the cover type is 'R', the premium term should be calculated as two-thirds of the benefit term.  
4. For other products, the premium term should be set to the benefit term unless specific conditions apply (e.g., 'ISECURE\_LOAN' products should have the premium term as two-thirds of the benefit term).  
5. The system should handle special cases for products like 'SUPER\_CASHGAIN', 'SAVE\_ASSURE', and 'CASH\_ASSURE' by subtracting 5 from the benefit term to get the premium term.  
6. The system should iterate through all records in the 'covers' block and update the benefit and premium terms accordingly.  
7. If the booking frequency is '01', the premium term should be set to 0.  
8. The system should navigate to the appropriate item based on whether the `ch\_prem\_term` is enabled.  
9. For 'InvestGain' products, if the premium term equals the benefit term, set the limited premium flag to 'N'; otherwise, set it to 'Y'.  
  
Definition of Done:  
- The premium terms are correctly calculated and displayed for all relevant products.  
- The system handles all specified conditions and exceptions.  
- The user can see the updated premium terms without any errors.  
- The system navigates to the correct item based on the enabled state of `ch\_prem\_term`.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- The query to retrieve the premium term from the `azbj\_mortgage\_term` table:  
 ```sql  
 SELECT premium\_term  
 INTO v\_pt  
 FROM azbj\_mortgage\_term  
 WHERE product\_id = :control.cn\_product\_id  
 AND cover\_code = :covers.cv\_cover\_code  
 AND loan\_term = :coverhead.ch\_benefit\_term;  
 ```

# Adjust Short Remittance Flag and Year-End Adjustment

Type: COVERHEAD

Title: Adjust Short Remittance Flag and Year-End Adjustment  
  
Acceptance Criteria:  
1. When the short remittance adjustment flag is checked:  
 - The system should check if the service tax is applicable for the policy holder's email ID.  
 - If the service tax is not applicable and the product is exempt from service tax, the system should display a message indicating that the service tax is not applicable and prompt the user to unselect the short remittance adjustment flag.  
2. If the year-end adjustment flag is set to 'Y', the system should prompt the user to confirm if they want the adjustment from ICC/STM Deposit.  
 - If the user does not confirm, the year-end adjustment flag should be set to 'N'.  
3. If the short remittance adjustment flag is checked and the product is not a group product, and the unit link is not 'Y' or 'P', and bulk adjustment is allowed, the year-end adjustment flag should be automatically set to 'Y'.  
  
Definition of Done:  
- The system correctly checks the service tax applicability and displays the appropriate message.  
- The system prompts the user for confirmation regarding the year-end adjustment and updates the flag based on the user's response.  
- The system automatically sets the year-end adjustment flag based on the specified conditions when the short remittance adjustment flag is checked.  
- All functionalities are tested and verified to work as expected without any Oracle Forms-specific terminology or dependencies.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include any direct database queries.

# Display and Interact with Child Cover Details

Type: COVERHEAD

Title: Display and Interact with Child Cover Details  
  
Acceptance Criteria:  
1. When the "CHILD COVERS" button is pressed, the system should display the "Nominee/Child Cover Details" window.  
2. The focus should automatically move to the "Child Name" field within the "Nominee/Child Cover Details" window.  
  
Definition of Done:  
- The "CHILD COVERS" button is visible and functional.  
- Pressing the "CHILD COVERS" button opens the "Nominee/Child Cover Details" window.  
- The "Child Name" field is focused when the window is displayed.  
- The functionality is tested and verified to work as expected.

# Validate Application Data on Button Press

Type: COVERHEAD

User Story: Validate Application Data on Button Press  
  
Title: Validate Application Data on Button Press  
  
Acceptance Criteria:  
1. When the "Validate" button is pressed, the system should initialize various variables and perform a series of checks and validations.  
2. The system should verify if the application number exists in the approval details and if the PAN issuance date is different from the stored date.  
3. If the review count is zero or the PAN issue date is different, the system should call a procedure to verify the PAN and age proof.  
4. The system should handle any exceptions that occur during the validation process and set a control flag to 'N'.  
5. If specific product IDs and agent codes are present, and the booking frequency and premium conditions are met, the system should set a flag based on the count of GSIP details.  
6. The system should call a general validation procedure to perform additional checks.  
7. If certain policy values are present, the system should disable specific fields related to beneficiaries.  
8. The system should handle cases where the policy is issued under the MWP Act by disabling beneficiary fields and setting appropriate messages.  
9. The system should call an auto BBU procedure for further processing.  
10. The system should handle counter-offer scenarios by enabling and executing specific triggers if certain conditions are met.  
11. The system should validate the application data based on various conditions and set appropriate error messages if any validation fails.  
  
Definition of Done:  
- The "Validate" button triggers the validation process.  
- All specified checks and validations are performed.  
- Appropriate flags and messages are set based on the validation results.  
- Any exceptions are handled gracefully.  
- The system is ready for further processing after validation.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- The system should execute a query to count the number of records in the approval details table where the application number matches and the PAN issuance date is different.  
- The system should execute a query to count the number of GSIP details where the application number matches.  
- The system should execute a query to update the approval details table if certain conditions are met.

# Initiate Counter Offer Process

Type: COVERHEAD

Detailed description: As a user, I want to be able to initiate a counter offer process when certain conditions are met, so that I can provide alternative terms to the client.  
  
Acceptance criteria:  
1. When the "Counter Offer" button is pressed, the system should check if there are any further requirements (e.g., specific test numbers) that need to be met.  
2. If the product ID is 321, the system should verify the existence of specific system constants and receipt dates.  
3. The system should log various stages of the counter offer process for auditing purposes.  
4. If the product definition is 'LIFE\_ASSURE\_PLAN', the system should loop through the covers and set specific flags.  
5. The system should calculate the premium based on various conditions, including product ID, cover code, and other parameters.  
6. If the product definition is 'FAMILY\_CARE\_FIRST', the system should handle specific logic for revised offers.  
7. The system should update the revised offer details, including policy reference, contract ID, activity number, and premium details.  
8. The system should handle specific logic for different cover codes, including 'R032A01', 'R043A01', 'R080A01', and 'R081A01'.  
9. The system should enable or disable certain fields based on the counter offer status.  
10. The system should display a message summarizing the new package, sum assured, benefit term, premium term, and premium.  
  
Definition of Done:  
- The counter offer process is initiated correctly when the button is pressed.  
- All necessary checks and validations are performed.  
- Logs are created at various stages of the process.  
- Premium calculations are accurate and based on the specified conditions.  
- Revised offer details are updated correctly.  
- Appropriate fields are enabled or disabled based on the counter offer status.  
- A summary message is displayed to the user.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- The system should query the `azbj\_system\_constants` table to check for specific system constants.  
- The system should query the `azbj\_batch\_items` table to retrieve receipt dates.  
- The system should query the `azbj\_revised\_offer\_partners` and `azbj\_revised\_offer` tables to handle revised offer logic.  
- The system should query the `azbj\_solution\_appln\_cover\_dtls` table to retrieve solution details.

# Validation and Calculation of Multiplier Field

Type: COVERHEAD

Title: Validation and Calculation of Multiplier Field  
  
Acceptance Criteria:  
1. If the "Multiplier" field is not null and the product ID is 291:  
 - If the "Multiplier" contains a fractional value and the package does not include "SINGLE", an error message "Fraction is not allowed in Multiplier for RP/LP" should be displayed.  
 - If the booking frequency is not '01', the sum assured should be calculated as the rounded value of the product of the multiplier, premium amount, and booking frequency.  
 - If the booking frequency is '01', the sum assured should be calculated as the rounded value of the product of the multiplier and premium amount.  
2. If the product ID is 343 and the "Multiplier" field is not null:  
 - If the booking frequency is not '01', the sum assured should be calculated as the rounded value of the product of the multiplier, premium amount, and booking frequency.  
 - If the booking frequency is '01', the sum assured should be calculated as the rounded value of the product of the multiplier and premium amount.  
  
Definition of Done:  
- The system correctly validates the "Multiplier" field based on the specified conditions.  
- The sum assured is accurately calculated and updated in the system.  
- Appropriate error messages are displayed for invalid input.  
- The functionality is tested and verified to ensure it meets the acceptance criteria.

# Implement Annuity Type Dropdown

Type: COVERHEAD

Title: Implement Annuity Type Dropdown  
  
Acceptance Criteria:  
1. The annuity type field should be a dropdown list containing six predefined options.  
2. The dropdown list should be disabled by default.  
3. The dropdown list should be positioned at the specified coordinates on the user interface.  
4. The dropdown list should have a white background and black text.  
5. The prompt for the dropdown list should be "Annuity Type" and should be right-aligned.  
6. The prompt should be bold and in the "MS Sans Serif" font.  
  
Definition of Done:  
1. The annuity type dropdown list is implemented and displays six predefined options.  
2. The dropdown list is disabled by default.  
3. The dropdown list is correctly positioned on the user interface.  
4. The dropdown list has the specified visual attributes (white background, black text).  
5. The prompt "Annuity Type" is displayed, right-aligned, and in bold "MS Sans Serif" font.  
6. The feature is tested and verified to meet the acceptance criteria.  
  
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.

# Activate Button Functionality

Type: COVERHEAD

Title: Activate Button Functionality  
  
Acceptance Criteria:  
1. When the "Activate" button is pressed, the system should:  
 - Retrieve the current username.  
 - Set a control flag to indicate the button was clicked.  
 - Populate a text field with a predefined message.  
 - Concatenate and set a comments field with values from various fields, using default values if any field is null.  
 - Freeze the covers tab.  
 - If the product ID is one of 335, 337, or 341:  
 - Enable the "Premium Term" field if it is currently disabled.  
 - Enable the "Annuity Option" field if it is currently disabled.  
 - Make the "Net Purchase Price" field visible and enabled if it is currently hidden or disabled.  
 - Allow updates and inserts on the "Net Purchase Price" field.  
 - Enable the "Validate" button.  
 - Move the cursor to the "Sum Assured" field.  
2. If any error occurs during the process, display an error message with the SQL error message.  
  
Definition of Done:  
- The "Activate" button functionality is implemented as described.  
- All specified fields are correctly updated and enabled based on the conditions.  
- Error handling is in place to display appropriate messages.  
- The functionality is tested and verified to work as expected.

# Cancel Offer and Navigate to Agent Code Input

Type: COVERHEAD

Title: Cancel Offer and Navigate to Agent Code Input  
  
Acceptance Criteria:  
1. When the "Cancel" button is pressed, the system should navigate to the agent code input field.  
2. The current view should be hidden upon pressing the "Cancel" button.  
  
Definition of Done:  
- The "Cancel" button should be functional and perform the specified actions.  
- The system should successfully navigate to the agent code input field.  
- The current view should be hidden without any errors.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- No specific database queries are mentioned in the provided XML content.

# Annuity Frequency Selection

Type: COVERHEAD

Title: Annuity Frequency Selection  
  
Acceptance Criteria:  
1. The user should be able to see a dropdown list labeled "Annuity Freq" on the "COVERS" tab.  
2. The dropdown list should contain five predefined frequency options.  
3. The dropdown list should be positioned correctly on the screen and be easily accessible.  
4. The selected frequency should be stored temporarily and marked as "dirty" if changed, indicating that the data has been modified but not yet saved.  
  
Definition of Done:  
1. The dropdown list for annuity frequency is implemented and visible on the "COVERS" tab.  
2. The list contains exactly five options for the user to choose from.  
3. The dropdown list is correctly positioned and styled according to the design specifications.  
4. Any changes made to the selected frequency are tracked and marked as modified until saved.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the dropdown list is not directly tied to a database item in this context.

# Calculate and Validate Benefit Term and Premium Term

Type: COVERHEAD

User Story: Calculate and Validate Benefit Term and Premium Term  
  
Detailed Description:  
As a user, I need the system to calculate and validate the benefit term and premium term based on the product definition, insured person's age, and other related parameters. This will ensure that the correct terms are applied to the insurance policy, and any discrepancies are flagged for review.  
  
Acceptance Criteria:  
1. The system should calculate the insured person's entry age using their date of birth and the inception date or effective date.  
2. If the product is 'LOAN\_PROTEC' and a package is selected, the system should:  
 - Retrieve the premium term from the database based on the product ID, cover code, and loan term.  
 - Set the premium term in the relevant fields.  
 - If no premium term is found, set the premium term to 0 and display an error message.  
3. For specific products, the system should adjust the premium term based on predefined rules (e.g., 'GROUP\_CREDIT\_PROTECT' products should have the premium term set to two-thirds of the benefit term).  
4. If the product ID is 281, the premium term should be set equal to the benefit term.  
5. For certain products, the system should adjust the benefit term based on the insured person's age and predefined rules (e.g., 'LIFE\_SECURE' should have the benefit term set to 100 minus the insured person's age).  
6. The system should loop through all related records and update the benefit term and premium term accordingly.  
7. If the booking frequency is '01', the premium term should be set to 0.  
8. The system should navigate to the next item based on whether the premium term field is enabled.  
  
Definition of Done:  
- The system correctly calculates and sets the benefit term and premium term based on the provided rules.  
- All related records are updated with the correct terms.  
- Any errors or discrepancies are flagged with appropriate messages.  
- The system navigates to the next item as expected.  
  
DB Queries for Table Reference CRUD Operations:  
- Retrieve premium term:  
 ```sql  
 SELECT premium\_term  
 INTO v\_pt  
 FROM azbj\_mortgage\_term  
 WHERE product\_id = :control.cn\_product\_id  
 AND cover\_code = :covers.cv\_cover\_code  
 AND loan\_term = :coverhead.ch\_benefit\_term;  
 ```  
  
- Retrieve multiplier:  
 ```sql  
 SELECT multiplier  
 INTO v\_multiplier  
 FROM azbj\_tasa\_suc\_multiplier  
 WHERE product\_id = :control.cn\_product\_id  
 AND cover\_code = v\_main\_cover\_code  
 AND :coverhead.ch\_benefit\_term BETWEEN min\_bt AND max\_bt  
 AND :coverhead.ch\_prem\_term BETWEEN min\_pt AND max\_pt;  
 ```  
  
- Retrieve date value:  
 ```sql  
 SELECT date\_value  
 INTO V\_on\_sar\_date  
 FROM azbj\_system\_constants  
 WHERE sys\_type = 'NB'  
 AND SYS\_CODE = 'LOADING\_ON\_SAR'  
 AND CHAR\_VALUE = 'Y';  
 ```  
  
- Retrieve non-standard age proof:  
 ```sql  
 SELECT non\_standered\_age\_proof  
 INTO v\_non\_standered\_age\_proof  
 FROM azbj\_master\_policy\_extn  
 WHERE master\_policy\_no = :control.cn\_master\_policy;  
 ```  
  
- Retrieve value consider:  
 ```sql  
 SELECT value\_consider  
 INTO v\_nonst\_rate  
 FROM azbj\_grp\_calc\_assumptions  
 WHERE master\_policy\_no = :control.cn\_master\_policy  
 AND TYPE = 'NONSTD\_BAND'  
 AND sub\_type = 'SUMASSURED'  
 AND :coverhead.ch\_sum\_assured BETWEEN value\_from AND value\_to  
 AND :covers.cv\_entry\_age BETWEEN NVL (age\_from, 0) AND NVL (age\_to, 99)  
 AND end\_date IS NULL;  
 ```  
  
This user story ensures that the system accurately calculates and validates the benefit and premium terms, providing a seamless experience for the user while maintaining data integrity and compliance with business rules.

# Display Counter Offer Reasons in LOV

Type: COVERHEAD

Title: Display Counter Offer Reasons in LOV  
  
Acceptance Criteria:  
- When the user double-clicks on the "Counter Offer Reason" field, a list of values (LOV) should be displayed.  
- The LOV should fetch and display reasons from the database where the system type is 'REASONS'.  
- The user should be able to select a reason from the LOV, and the selected reason should be populated in the "Counter Offer Reason" field.  
  
Definition of Done:  
- The "Counter Offer Reason" field is present on the form.  
- Double-clicking the field triggers the LOV to display.  
- The LOV fetches data from the database and displays it correctly.  
- The user can select a reason from the LOV, and the selected reason is populated in the "Counter Offer Reason" field.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- SELECT SYS\_DESC FROM AZBJ\_SYSTEM\_CONSTANTS WHERE SYS\_TYPE='REASONS';

# Submit Counter Offer Reason

Type: COVERHEAD

Title: Submit Counter Offer Reason  
  
Acceptance Criteria:  
1. When the "Submit" button is pressed, if no counter offer reason is selected, the system should prompt the user to select a reason.  
2. If more than three reasons are selected, the system should display an error message stating that no more than three reasons can be selected.  
3. For specific product conditions (ULIP/TRAD/GROUP), the system should prompt the user to generate a counter offer letter.  
4. If the user chooses to modify counter offer details, the system should navigate to the appropriate section and allow modifications.  
5. The system should handle any errors gracefully and display appropriate error messages.  
  
Definition of Done:  
- The "Submit" button functionality is implemented and tested.  
- The system correctly prompts the user to select a counter offer reason if none is selected.  
- The system displays an error message if more than three reasons are selected.  
- The system prompts the user to generate a counter offer letter for specific product conditions.  
- The system allows the user to modify counter offer details as needed.  
- All error handling and messaging are implemented and tested.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include specific CRUD operations for database tables.

# Navigate to Reinsurance Cover Details

Type: COVERHEAD

Title: Navigate to Reinsurance Cover Details  
  
Acceptance Criteria:  
- When the "Reins" button is pressed, the system should navigate to the "Reinsurance Cover Details" section.  
- If an error occurs during navigation, an error message should be displayed to the user.  
  
Definition of Done:  
- The "Reins" button is functional and navigates to the "Reinsurance Cover Details" section.  
- Error handling is implemented to display an error message if navigation fails.  
- The feature is tested and verified to ensure it works as expected.

# Manage visibility and value of CH\_ORIGINAL\_SA based on PREMIUM\_FINANCE selection and product ID

Type: COVERHEAD

Title: Manage visibility and value of CH\_ORIGINAL\_SA based on PREMIUM\_FINANCE selection and product ID  
  
Acceptance Criteria:  
1. When the "PREMIUM\_FINANCE" option is set to 'Y' and the product ID is not 233:  
 - The "CH\_ORIGINAL\_SA" field should become visible.  
2. When the "PREMIUM\_FINANCE" option is not set to 'Y' or the product ID is 233:  
 - The "CH\_ORIGINAL\_SA" field should be set to 0.  
 - The "PREMIUM\_FINANCE" option should be reset to 'N'.  
 - The "CH\_ORIGINAL\_SA" field should become invisible.  
  
Definition of Done:  
- The system correctly adjusts the visibility and value of the "CH\_ORIGINAL\_SA" field based on the conditions specified.  
- The changes are tested and verified to ensure they work as expected.  
- The user interface updates dynamically without requiring a page refresh.  
- The functionality is documented for future reference and maintenance.

# Implement Risk With Balic Button Functionality

Type: COVERHEAD

Title: Implement Risk With Balic Button Functionality  
  
Acceptance Criteria:  
1. When the user presses the "Risk With Balic" button, the system should navigate to the "BALIC\_RISK" section and count the number of records.  
2. If there are records in the "BALIC\_RISK" section and the insured person's ID does not match the ID in the "BALIC\_RISK" section, the system should call a function to update the risk data for the insured person.  
3. If there are no records in the "BALIC\_RISK" section, the system should call a function to update the risk data for the insured person.  
4. The system should then navigate to the "BALIC\_RISK\_PH" section and count the number of records.  
5. If there are records in the "BALIC\_RISK\_PH" section and the policy holder's ID does not match the ID in the "BALIC\_RISK\_PH" section, the system should call a function to update the risk data for the policy holder.  
6. If there are no records in the "BALIC\_RISK\_PH" section, the system should call a function to update the risk data for the policy holder.  
  
Definition of Done:  
- The "Risk With Balic" button functionality is implemented and tested.  
- The system correctly navigates to the appropriate sections and counts the records.  
- The system correctly updates the risk data for both the insured person and the policy holder based on the conditions specified.  
- All acceptance criteria are met and verified through testing.

# Loan Type Validation and Navigation

Type: COVERHEAD

Title: Loan Type Validation and Navigation  
  
Acceptance Criteria:  
1. When the loan type is validated, if the global loading flag is set to 'F', the form status should be updated to 'Y'.  
2. Upon moving to the next item:  
 - The system should calculate the `Perm\_receipt\_date` based on the received date or a default date.  
 - If the booking frequency is monthly and the product ID is 10, the system should check the receipt date against specific date ranges and display an error message if the conditions are not met.  
 - If the product ID is 14 or 16 and the booking frequency is '01', an error message should be displayed indicating that the single premium mode is not available for this product.  
 - For specific product IDs, the system should navigate to the premium amount field; otherwise, it should navigate to the package field.  
3. Error handling should be in place to display any SQL errors encountered during the process.  
  
Definition of Done:  
- The loan type validation logic is implemented and tested.  
- The navigation logic based on the conditions is implemented and tested.  
- Error messages are displayed correctly based on the specified conditions.  
- The system navigates to the correct field based on the product ID and booking frequency.  
- All acceptance criteria are met and verified through testing.

# Automatic Sum Assured Calculation

Type: COVERHEAD

Title: Automatic Sum Assured Calculation  
  
Acceptance Criteria:  
1. When the "Calculate Sum Assured" checkbox is checked, the system should:  
 - Retrieve the insured person's sex.  
 - Retrieve the entry age from the covers section.  
 - Retrieve the benefit term from the coverhead section.  
 - Retrieve the product ID from the control section.  
 - Retrieve the total sum assured from the SUSAC section.  
 - Determine the age proof type using the insured person's age proof.  
 - Calculate the sum assured using the retrieved parameters.  
 - Update the sum assured in the coverhead section.  
 - Update the age proof extra in the coverhead section.  
  
Definition of Done:  
- The system correctly calculates and updates the sum assured and age proof extra when the checkbox is checked.  
- All relevant fields are updated with the correct values based on the calculation.  
- The functionality is tested and verified to ensure accuracy.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided logic involves procedural calculations and updates within the application, not direct database CRUD operations.

# Manage Joint Life Option in Insurance Policy Form

Type: COVERHEAD

Title: Manage Joint Life Option in Insurance Policy Form  
  
Acceptance Criteria:  
1. When the "Joint Life" checkbox is checked:  
 - The system should set the `cn\_joint\_life` control value to 'JL'.  
 - If the insured person (IP) and policy holder (PH) have the same partner reference, the system should:  
 - Uncheck the "Joint Life" checkbox.  
 - Set the `cn\_joint\_life` control value to 'SL'.  
 - Display an error message: "For Joint Life IP and PH cannot be same".  
 - If the "Joint Life" checkbox remains checked:  
 - The system should make the "QUESTIONNAIRE\_PH" tab page visible if it is not already visible.  
 - Enable the following fields for the policy holder:  
 - Age proof  
 - Age proof ID  
 - Height  
 - Weight  
 - Weight change  
 - Disable the "PH\_EQUAL\_INS" field.  
 - If the selected package does not include "Joint" and the product definition is not one of the specified products, the system should:  
 - Clear the package selection.  
 - Navigate to the "LOADINGS" block, clear it, and go to the first record.  
 - Navigate back to the "COVERS" block, clear it, and go to the first record.  
 - Display a warning message: "Please Select The Joint Life Package".  
2. When the "Joint Life" checkbox is unchecked:  
 - The system should set the `cn\_joint\_life` control value to 'SL'.  
 - If the "QUESTIONNAIRE\_PH" tab page is visible, the system should make it invisible.  
 - Disable the following fields for the policy holder:  
 - Age proof  
 - Age proof ID  
 - Height  
 - Weight  
 - Weight change  
 - Enable the "PH\_EQUAL\_INS" field.  
3. If the global loading status is 'F', the system should set the form status control value to 'Y'.  
4. The system should navigate to the "CH\_PACKAGE" item after processing the "Joint Life" checkbox change.  
  
Definition of Done:  
- The "Joint Life" checkbox functionality is implemented as per the acceptance criteria.  
- All specified fields and controls are enabled/disabled correctly based on the "Joint Life" checkbox state.  
- Appropriate error and warning messages are displayed as per the conditions.  
- The form navigation and field updates are performed correctly.  
- The feature is tested and verified to work as expected.

# Implement Money Back Option List

Type: COVERHEAD

Title: Implement Money Back Option List  
  
Acceptance Criteria:  
1. The "Money Back Option" should be presented as a list with three predefined choices.  
2. The list should be enabled and allow both insertion and updates.  
3. The "Money Back Option" should be located within the "New Business" module and should be visible on the "COVERS" tab.  
4. The list should be styled with a specific font and color scheme to ensure consistency with the rest of the application.  
  
Definition of Done:  
1. The "Money Back Option" list is implemented and integrated into the "New Business" module.  
2. The list is functional, allowing users to select from the predefined choices.  
3. The list adheres to the specified styling and positioning requirements.  
4. The feature has been tested and verified to work as expected.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include specific database queries.

# Implement Cover Type Dropdown and Validation Logic

Type: COVERHEAD

Title: Implement Cover Type Dropdown and Validation Logic  
  
Acceptance Criteria:  
1. The cover type field should be a dropdown list with predefined options.  
2. The field should be disabled when the form is initially loaded.  
3. When the cover type is validated, if the form is not in loading mode, the form status should be set to 'Y'.  
4. When navigating to the next item:  
 - If the booking frequency is monthly and the product ID is 10, the system should check the receipt date and display an error message if the date conditions are not met.  
 - If the product ID is 14 or 16 and the booking frequency is '01', an error message should be displayed indicating that the single premium mode is not available for this product.  
 - If the product is unit-linked or defined as 'INVESTPLUS' or 'INVESTPLUS\_PREM', the focus should move to the premium amount field; otherwise, it should move to the package field.  
  
Definition of Done:  
- The cover type dropdown list is implemented and contains the predefined options.  
- The field is disabled upon form load.  
- The form status is correctly updated upon validation.  
- The navigation logic and error messages are implemented as per the specified conditions.  
- The focus moves to the correct field based on the product type and booking frequency.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include any direct database queries.

# Handle Consent for Adjustment of Survival Benefits

Type: COVERHEAD

Title: Handle Consent for Adjustment of Survival Benefits  
  
Acceptance Criteria:  
1. When the "Cash Back" flag is checked and the product ID is 201:  
 - The system should navigate to the "Further Requirements" section.  
 - It should iterate through the records to check if the test number 'M555' exists.  
 - If 'M555' is found, the flag should be set to 'Y' and the loop should break.  
 - If 'M555' is not found, a new record should be created with the test number 'M555' and a description indicating consent for adjustment of survival benefits towards premium.  
 - The new record should have the date called set to the current date and the raised by field set to 'USER CALLED'.  
2. If the "Cash Back" flag is unchecked:  
 - The system should navigate to the "Further Requirements" section.  
 - It should iterate through the records to find and delete any record with the test number 'M555'.  
 - If the control flag for WIP continue is set to 'T', the system should also delete the corresponding record from the external table 'WIP\_AZBJ\_MED\_UW'.  
  
Definition of Done:  
- The system correctly handles the creation or deletion of records based on the "Cash Back" flag status.  
- All specified conditions and actions are met without errors.  
- The functionality 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 logic involves Oracle Forms-specific constructs and cannot be directly executed in the database without modification.

# Delete Cover Functionality

Type: COVERHEAD

User Story: Delete Cover Functionality  
  
Detailed Description:  
As a user, I want to be able to delete a cover from the list of covers associated with a policy, ensuring that all necessary validations and dependencies are checked before the deletion is allowed.  
  
Acceptance Criteria:  
1. When the delete cover button is pressed, the system should navigate to the 'Loadings' section, then back to the 'Covers' section, and determine the total number of records.  
2. The system should iterate through each cover record and perform the following checks:  
 - If the cover code starts with 'R' and the product ID is 4, and the cover is marked for deletion, display an error message indicating that the selected riders are compulsory with the selected package.  
 - If the cover code starts with 'L', store the entry age of the cover.  
 - Track the presence of specific rider codes ('R001A01', 'R004A01', 'R005A01', 'R008A01', 'R036A01', 'R036B01', 'R035A01') and mark them accordingly.  
 - If a cover code starts with 'T' and the sum insured for the whole cover is 0, mark the top-up for deletion.  
3. After iterating through all records, the system should enforce the following rules:  
 - Certain rider codes must always be together (e.g., 'R001A01' and 'R005A01').  
 - If specific conditions are met, adjust the deletion flags for the covers accordingly.  
4. The system should then iterate through the records again to apply the deletion flags based on the rules and conditions checked earlier.  
5. If the product ID is 315, check if any policy members exist for the cover being deleted and display an error message if any members are found, preventing the deletion.  
6. Finally, disable the commit and exit buttons on the toolbar to prevent further changes until the form is revalidated.  
  
Definition of Done:  
- The delete cover functionality should be fully implemented and tested.  
- All specified validations and rules should be enforced correctly.  
- Appropriate error messages should be displayed when conditions are not met.  
- The system should prevent further changes until the form is revalidated after a deletion attempt.  
  
DB Queries for Table Reference CRUD Operations:  
- Not applicable as the provided logic involves form-specific operations and validations that are not directly executable as standalone database queries.

# Automatic Sum Assured Calculation Based on GMI

Type: COVERHEAD

Detailed description: As a user, I want the system to automatically calculate and display the sum assured based on the GMI (Gross Monthly Income) when a specific product is selected, ensuring that the sum assured field is not editable.  
  
Acceptance criteria:  
1. When the product with ID 299 is selected:  
 - If the GMI field is empty, the system should prompt the user to enter a value for GMI.  
 - If the GMI field is not empty, the system should calculate the sum assured as GMI multiplied by 144.  
 - The sum assured field should be displayed but not editable by the user.  
  
Definition of Done:  
- The system correctly identifies when the product ID is 299.  
- The system prompts the user to enter a GMI value if it is empty.  
- The system calculates the sum assured as GMI 144 and displays it.  
- The sum assured field is set to read-only.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided logic does not include direct database CRUD operations.

# Annuity Option Dropdown Implementation

Type: COVERHEAD

Detailed description: As a user, I want to select an annuity option from a predefined list on the "New Business" form, so that I can specify the type of annuity for the business transaction.  
  
Acceptance criteria:  
1. The annuity option field should be a dropdown list with three predefined options.  
2. The field should be disabled by default and only enabled under specific conditions.  
3. The field should be visible and properly aligned on the "COVERS" tab of the "New Business" form.  
4. The field should have a label "Annuity Option" that is centered and bold.  
5. The field should allow insertion and updates when enabled.  
  
Definition of Done:  
1. The annuity option dropdown list is implemented and contains three predefined options.  
2. The field is disabled by default and can be enabled based on specific business logic.  
3. The field is visible and correctly aligned on the "COVERS" tab.  
4. The label "Annuity Option" is centered and bold.  
5. The field allows insertion and updates when enabled.  
  
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.

# Select Income Payout Frequency

Type: COVERHEAD

Title: Select Income Payout Frequency  
  
Acceptance Criteria:  
1. The user should be able to see a list of predefined payout frequencies.  
2. The list should be displayed in uppercase.  
3. The list should be visible on the "COVERS" tab.  
4. The user should be able to select an option from the list.  
5. The selected option should be saved and updated in the system.  
6. The list should have a maximum length of 500 characters.  
7. The list should be displayed with a white background and black text.  
8. The prompt for the list should be "Income Payout Frequency" and should be right-justified.  
9. The prompt should be displayed in bold and plain style with a font size of 800.  
10. The list should be enabled and allow for insertion and updates.  
  
Definition of Done:  
1. The user can successfully select a payout frequency from the list.  
2. The selected payout frequency is saved and updated in the system.  
3. The list and its prompt are displayed as specified in the acceptance criteria.  
4. 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 XML content does not include specific SQL queries or table references.

# Enable/Disable RI Reference Number based on 'Refer To RI' selection

Type: COVERHEAD

Title: Enable/Disable RI Reference Number based on 'Refer To RI' selection  
  
Acceptance Criteria:  
- When the "Refer To RI" option is set to 'Y':  
 - The RI Reference Number field should be enabled.  
- When the "Refer To RI" option is set to any value other than 'Y':  
 - The RI Reference Number field should be cleared and disabled.  
  
Definition of Done:  
- The functionality should be implemented and tested to ensure that the RI Reference Number field behaves as described in the acceptance criteria.  
- The user interface should reflect the changes immediately upon selection of the "Refer To RI" option.  
- The changes should be documented and reviewed for accuracy and completeness.

# View PAN Approval Details

Type: COVERHEAD

Title: View PAN Approval Details  
  
Acceptance Criteria:  
1. When the "PAN Approved Details" button is pressed, the system should navigate to the PAN approval details section.  
2. The system should clear any existing data in the PAN approval details section.  
3. The system should retrieve and display the following details for each approval:  
 - Approved User  
 - User Name (retrieved using a function)  
 - Approval Date (truncated to date only)  
 - Approval Time (formatted as 'hh:mi:ss am')  
 - Status (Standard, Non-Standard, or Review based on a flag)  
4. The system should only display records where the age proof is 'PC' and the PAN standard flag is not 'NA'.  
5. The system should order the records by the creation date in descending order.  
6. If no PAN details are found for the application, the system should display a message indicating this and return focus to the "PAN Approved Details" button.  
  
Definition of Done:  
- The "PAN Approved Details" button should function as described in the acceptance criteria.  
- The PAN approval details should be displayed correctly based on the retrieved data.  
- Appropriate error handling should be in place to manage any exceptions.  
- The user should receive a message if no PAN details are found for the application.  
  
DB queries for Table reference CRUD operations only(With Usage):  
```sql  
SELECT APPROVED\_USER,  
 az\_pk2\_general.getusername(APPROVED\_USER) AS user\_name,  
 TRUNC(CREATE\_DATE) AS apprv\_date,  
 TO\_CHAR(CREATE\_DATE, 'hh:mi:ss am') AS apprv\_time,  
 (CASE  
 WHEN PAN\_STD\_FLAG = 'Y' THEN 'Standard'  
 WHEN PAN\_STD\_FLAG = 'N' THEN 'Non-Standard'  
 WHEN PAN\_STD\_FLAG = 'R' THEN 'Review'  
 END) AS status  
FROM azbj\_supervisor\_appr\_det  
WHERE AGE\_PROOF = 'PC'   
 AND NVL(PAN\_STD\_FLAG, 'NA') <> 'NA'  
 AND application\_no = TO\_CHAR(NVL(:INSURED\_PERSON.IP\_VERF\_NO, :INSURED\_PERSON.IP\_SIGN\_CARD\_NO))  
ORDER BY CREATE\_DATE DESC;  
```

# Implement Loan Disbursed Date Field

Type: COVERHEAD

Title: Implement Loan Disbursed Date Field  
  
Acceptance Criteria:  
1. The loan disbursed date field should be enabled and allow the user to input a date.  
2. The field should accept a maximum of 11 characters in date format.  
3. The field should be positioned correctly on the form and be visually distinct with a custom background color.  
4. The field should be part of the "COVERS" tab page.  
5. The field should allow both insert and update operations.  
6. The field should trigger the `azbj\_backdate` logic when the user navigates to the next item.  
  
Definition of Done:  
1. The loan disbursed date field is implemented and visible on the form.  
2. The field accepts date input and adheres to the specified maximum length.  
3. The field is correctly positioned and styled as per the requirements.  
4. The field is functional within the "COVERS" tab page.  
5. The field allows data entry and updates.  
6. The `azbj\_backdate` logic is executed when navigating to the next item.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include any specific database queries.

# Dynamic Adjustment of Field Properties Based on Solution Name

Type: COVERHEAD

Title: Dynamic Adjustment of Field Properties Based on Solution Name  
  
Acceptance Criteria:  
1. When the "Solution Name" field is empty:  
 - The "SOLUTION\_PRODUCT2" tab should be hidden.  
 - The "AMOUNT\_INVESTED" field should be hidden.  
 - The "MULTI\_NOMINEE" field should be enabled.  
 - The "CH\_SUM\_ASSURED" field should allow insert and update operations.  
 - The "ASSIGNED" field should be enabled.  
 - The "TOT\_PREM\_PAYBLE" field should display the prompt text "Total Prem Payable".  
 - The "CV\_SUM\_INSURED\_WHOLE\_COVER" field should be enabled.  
 - The "SOL\_CV\_SUM\_INSURED\_WHOLE\_COVER" field should be enabled.  
 - The "sol\_tot\_prem\_payble" and "sol\_tot\_prem\_payble1" fields should be set to 0.  
  
2. When the "Solution Name" field is not empty:  
 - The "SOLUTION\_PRODUCT2" tab should be visible.  
 - The "AMOUNT\_INVESTED" field should be visible and enabled.  
 - The "MULTI\_NOMINEE" field should be disabled.  
 - The "CH\_SUM\_ASSURED" field should not allow insert and update operations.  
 - The "ASSIGNED" field should be disabled.  
 - The "TOT\_PREM\_PAYBLE" field should display the prompt text "Total Prem Payable Prod 1".  
 - The "CV\_SUM\_INSURED\_WHOLE\_COVER" field should be disabled.  
 - The "SOL\_CV\_SUM\_INSURED\_WHOLE\_COVER" field should be disabled.  
  
3. Additional logic:  
 - Calculate the entry age using the function `azbj\_calc\_entry\_age` with parameters `ip\_date\_of\_birth`, `ch\_inception\_date` or `cn\_effective\_date`, 'P', and 1.  
 - If the "Solution Name" field value is not equal to 1:  
 - Hide the "SOL\_SAR" field if it is visible.  
 - Set the prompt text of the "SOL\_CH\_SUM\_ASSURED" field to "Sum Assured".  
 - Set the "sol\_ch\_benefit\_term" field to 100 minus the calculated entry age.  
 - Allow insert and update operations on the "CH\_SUM\_ASSURED" field.  
 - Allow insert and update operations on the "SOL\_CH\_PREM\_TERM" field.  
  
Definition of Done:  
- The system dynamically adjusts the visibility and properties of fields and tabs based on the value of the "Solution Name" field.  
- All acceptance criteria are met.  
- The functionality is tested and verified to ensure it works as expected.  
- The user interface is updated to reflect the changes without any Oracle Forms-specific terminology.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Not applicable as the provided XML content does not include any specific CRUD operations.

# Automatic Calculation of Sum Assured on Death

Type: COVERHEAD

Title: Automatic Calculation of Sum Assured on Death  
  
Acceptance Criteria:  
1. If the product ID is 343 and the booking frequency is not '01':  
 - The "Sum Assured on Death" should be calculated as the greater value between:  
 - The product of the multiplier, premium amount, and booking frequency.  
 - 105% of the product of the premium amount and booking frequency.  
 - The calculated value should be rounded and any null values should default to 0.  
2. If the product ID is 343 and the booking frequency is '01':  
 - The "Sum Assured on Death" should be calculated as the greater value between:  
 - The product of the multiplier and premium amount.  
 - 105% of the premium amount.  
 - The calculated value should be rounded and any null values should default to 0.  
  
Definition of Done:  
- The system correctly calculates the "Sum Assured on Death" based on the specified conditions.  
- The calculated value is displayed in the appropriate field without any manual intervention.  
- The logic is thoroughly tested to ensure accuracy and consistency.  
- The feature is documented for future reference and maintenance.

# Manage Auto-Pay Rebate Option

Type: COVERHEAD

Title: Manage Auto-Pay Rebate Option  
  
Acceptance Criteria:  
1. The auto-pay rebate option should be represented as a checkbox.  
2. The checkbox should have two states: checked (Y) and unchecked (N).  
3. The default state of the checkbox should be unchecked (N).  
4. When the checkbox state is changed, no additional actions or triggers should be executed.  
  
Definition of Done:  
1. The auto-pay rebate checkbox is implemented and visible in the user interface.  
2. The checkbox correctly reflects the checked and unchecked states.  
3. The default state of the checkbox is set to unchecked (N).  
4. Changing the state of the checkbox does not trigger any additional actions.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the checkbox is not directly linked to a database item.

# Implement Salaried Rebate Flag Checkbox

Type: COVERHEAD

Title: Implement Salaried Rebate Flag Checkbox  
  
Acceptance Criteria:  
- The checkbox should have a label "Salaried Rebate Flag".  
- The checkbox should be initialized to "N" (unchecked) by default.  
- When checked, the value should be "Y".  
- When unchecked, the value should be "N".  
- The checkbox should be located on the "COVERS" tab page.  
- The checkbox should be positioned at coordinates (25, 250) on the form.  
- The checkbox should be 175 units wide and 21 units high.  
- The checkbox should not be visible by default.  
- The checkbox should not display horizontal or vertical scrollbars.  
- The checkbox should use the "MS Sans Serif" font, with a size of 8, plain style, and bold weight.  
- The background color of the tab should be gray, and the foreground color should be black.  
- The tab should be 1076 units wide and 980 units high.  
- The window containing the tab should be titled "New Business", with dimensions 1023 units wide and 550 units high, and should display a vertical scrollbar.  
  
Definition of Done:  
- The checkbox "Salaried Rebate Flag" is implemented according to the acceptance criteria.  
- The checkbox is tested to ensure it initializes correctly and toggles between "Y" and "N" values.  
- The checkbox is positioned correctly on the "COVERS" tab page.  
- The form is reviewed and approved by stakeholders.  
  
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.

# Validate and Populate Investment Amount

Type: COVERHEAD

User Story: Validate and Populate Investment Amount  
  
Detailed description:   
As a user, I want to ensure that the "Annual Amount to be Invested" field is validated and populated correctly based on the selected solution name and other related conditions, so that the investment details are accurate and compliant with business rules.  
  
Acceptance criteria:  
1. If the "Solution Name" is not selected and the "Annual Amount to be Invested" is provided, the system should prompt the user to select a solution name.  
2. If the "Solution Name" is selected and its value is 1, the system should calculate the beneficiary's age. If the age is 12 or above, an error message should be displayed indicating that the nominee must be a minor with an age less than 12.  
3. The system should populate the solution product details and navigate to the "Solution Covers" section.  
4. If the cover code in the "Solution Covers" section starts with 'L', the "Total Rider Investment" should be set to the "Annual Amount to be Invested" and should be made non-editable.  
5. The system should navigate to the "Total Rider Investment" field in the "Solution Covers" section.  
  
Definition of Done:  
- The validation and population logic for the "Annual Amount to be Invested" field is implemented as per the acceptance criteria.  
- The system prompts and error messages are displayed correctly based on the conditions.  
- The "Total Rider Investment" field is correctly populated and made non-editable when applicable.  
- The navigation between fields and sections works seamlessly as per the defined logic.  
- The functionality is tested and verified to ensure it meets the requirements.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Not applicable as the provided XML content does not include any direct database queries that can be executed independently of Oracle Forms constructs.

# Implement Fully Vaccinated Checkbox

Type: COVERHEAD

Title: Implement Fully Vaccinated Checkbox  
  
Acceptance Criteria:  
- The checkbox should have two states: checked and unchecked.  
- When the checkbox is checked, it should store the value 'Y'.  
- When the checkbox is unchecked, it should store the value 'N'.  
- The checkbox should be initialized to 'N' (unchecked) by default.  
- The checkbox should be labeled "FULLY\_VACCINATED\_FLAG".  
- The checkbox should not be visible on the form by default.  
- The checkbox should be located within the "COVERS" tab of the form.  
- The checkbox should trigger no additional actions when its state is changed.  
  
Definition of Done:  
- The checkbox is implemented and can be interacted with as described.  
- The checkbox correctly stores 'Y' or 'N' based on its state.  
- The checkbox is initialized to 'N' by default.  
- The checkbox is labeled correctly and is not visible by default.  
- The checkbox is located within the specified tab.  
- No additional actions are triggered when the checkbox state is changed.

# Implement 'Insurance for All' Checkbox

Type: COVERHEAD

Title: Implement 'Insurance for All' Checkbox  
  
Acceptance Criteria:  
- The checkbox for 'Insurance for All' should be available on the 'Covers' tab.  
- The checkbox should have a default value of 'No' (unchecked).  
- When the checkbox is selected, it should store a value of 'Yes'.  
- When the checkbox is deselected, it should store a value of 'No'.  
- The checkbox should not be visible to the user by default.  
  
Definition of Done:  
- The 'Insurance for All' checkbox is implemented and functional as per the acceptance criteria.  
- The checkbox is correctly positioned on the 'Covers' tab.  
- The default value is set to 'No'.  
- The checkbox correctly updates its value when selected or deselected.  
- The checkbox is hidden from the user interface by default.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the checkbox is not directly linked to a database item.

# Handle Joint Life Option Selection and Validation

Type: COVERHEAD

Title: Handle Joint Life Option Selection and Validation  
  
Acceptance Criteria:  
1. When the "Joint Life" option is selected:  
 - If the insured person (IP) and policy holder (PH) are the same, display an error message indicating that they cannot be the same for a joint life policy.  
 - Enable the fields for age proof, age proof ID, height, weight, and weight change for the policy holder.  
 - Disable the "PH\_EQUAL\_INS" field.  
 - If the selected package does not include "Joint" and the product definition is not one of the specified types, clear the package selection and prompt the user to select a joint life package.  
 - Make the "QUESTIONNAIRE\_PH" tab page visible.  
  
2. When the "Joint Life" option is not selected:  
 - Disable the fields for age proof, age proof ID, height, weight, and weight change for the policy holder.  
 - Enable the "PH\_EQUAL\_INS" field.  
 - Hide the "QUESTIONNAIRE\_PH" tab page.  
  
3. Ensure that the form status is updated correctly when the "Joint Life" option is changed.  
  
Definition of Done:  
- The system correctly handles the selection and validation of the "Joint Life" option.  
- Appropriate fields and packages are enabled or disabled based on the user's input.  
- Error messages and prompts are displayed as specified.  
- The form status is updated correctly.  
- The "QUESTIONNAIRE\_PH" tab page visibility is managed based on the "Joint Life" option selection.

# Automatic Calculation of Sum Assured

Type: COVERHEAD

Title: Automatic Calculation of Sum Assured  
  
Acceptance Criteria:  
1. If the product ID is 343 and the booking frequency is not '01':  
 - The sum assured should be calculated as the greater value between:  
 - The product of the multiplier, premium amount, and booking frequency.  
 - 105% of the product of the premium amount and booking frequency.  
 - The calculated value should be rounded and set to zero if null.  
2. If the product ID is 343 and the booking frequency is '01':  
 - The sum assured should be calculated as the greater value between:  
 - The product of the multiplier and premium amount.  
 - 105% of the premium amount.  
 - The calculated value should be rounded and set to zero if null.  
  
Definition of Done:  
- The sum assured is correctly calculated and displayed based on the given conditions.  
- The calculation logic is triggered when the user navigates to the next item or validates the item.  
- The system handles null values appropriately by setting them to zero.  
- The feature is tested and verified to ensure accuracy and consistency in different scenarios.

# Automatic Discount Type Determination

Type: COVERHEAD

Title: Automatic Discount Type Determination  
  
Acceptance Criteria:  
1. The system should check if the product ID is not in a specified list and then determine if the agent code matches certain criteria to set a discount agent flag.  
2. The system should retrieve and set the employee code based on the application number.  
3. The system should determine and set the permanent receipt date and IRDA launch date if they are not already set.  
4. The system should initialize the discount type to null.  
5. The system should check if the agent code matches certain criteria to determine if the transaction is offline or online.  
6. The system should set the discount type based on various conditions involving the agent code, employee ID, product ID, and other parameters.  
7. The system should clear and populate a list of employee types based on the determined discount type and other conditions.  
8. The system should execute a trigger to handle changes in the list of employee types.  
9. The system should clear and populate a list of premium discounts based on the determined discount type, product ID, and other conditions.  
  
Definition of Done:  
- The discount type is correctly set based on the specified conditions.  
- The list of employee types is correctly populated and updated based on the discount type.  
- The list of premium discounts is correctly populated and updated based on the discount type and product ID.  
- All relevant data is retrieved and set correctly without errors.  
- The system handles exceptions gracefully and sets default values where necessary.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Retrieve discount agent flag:  
 ```sql  
 SELECT 'Y'  
 INTO v\_disc\_agnt  
 FROM azbj\_system\_constants  
 WHERE SYS\_TYPE = 'DISCOUNT'  
 AND sys\_CODE = 'DISCOUNT\_AGENT'  
 AND :agents.ag\_agent\_code LIKE '%' || char\_value || '%'  
 AND ROWNUM = 1;  
 ```  
  
- Retrieve employee code:  
 ```sql  
 SELECT var30  
 INTO pk\_vars.v\_emp\_code  
 FROM websales.azbj\_cq\_trans\_data  
 WHERE application\_no = TO\_CHAR(NVL(:insured\_person.ip\_verf\_no, :insured\_person.ip\_sign\_card\_no));  
 ```  
  
- Retrieve permanent receipt date:  
 ```sql  
 SELECT perm\_receipt\_date  
 INTO v\_perm\_rec\_date  
 FROM azbj\_batch\_items  
 WHERE perm\_receipt\_no = :control.cn\_permrcpt\_no  
 AND NVL(PRINT, 'X') <> 'C'  
 AND TRANSACTION\_TYPE = 'FRP';  
 ```  
  
- Retrieve IRDA launch date:  
 ```sql  
 SELECT start\_date  
 INTO v\_ver2\_lunch\_date  
 FROM azbj\_system\_constants  
 WHERE sys\_type = 'IRDA' AND sys\_code = 'IRDA\_LAUNCH';  
 ```  
  
- Retrieve offline/online status:  
 ```sql  
 SELECT COUNT()  
 INTO v\_instab\_cnt  
 FROM azbj\_system\_constants  
 WHERE sys\_type = 'OFFLINE\_ONLINE'  
 AND sys\_code = 'OFFLINE\_ONLINE'  
 AND :agents.ag\_agent\_code LIKE '%' || char\_value || '%'  
 AND ROWNUM = 1;  
 ```  
  
- Retrieve premium discount:  
 ```sql  
 SELECT SYS\_CODE  
 INTO :COVERHEAD.CH\_DISC  
 FROM azbj\_system\_constants  
 WHERE sys\_type = 'PREM\_DISC'  
 AND char\_value = CASE  
 WHEN :control.cn\_product\_id < 287 AND :COVERHEAD.DISCOUNT\_TYPE = 'D' THEN 'W'  
 ELSE :COVERHEAD.DISCOUNT\_TYPE  
 END  
 AND prod\_id = :control.cn\_product\_id  
 AND sys\_desc = CASE  
 WHEN :control.cn\_product\_id = 315 THEN :coverhead.ch\_package\_code  
 WHEN :control.cn\_product\_id IN (339) AND :COVERHEAD.DISCOUNT\_TYPE = 'S' THEN :coverhead.ch\_package\_code  
 ELSE sys\_desc  
 END  
 AND ROWNUM = 1  
 AND TO\_DATE(pk\_vars.v\_perm\_rec\_date, 'dd/mm/rrrr') >= NVL(TO\_DATE(start\_date, 'dd/mm/rrrr'), TO\_DATE('01/01/2000', 'dd/mm/rrrr'))  
 AND TO\_DATE(pk\_vars.v\_perm\_rec\_date, 'dd/mm/rrrr') <= NVL(TO\_DATE(end\_date, 'dd/mm/rrrr'), TO\_DATE('01/01/3000', 'dd/mm/rrrr'));  
 ```

# Handle Joint Life Checkbox Selection

Type: COVERHEAD

Title: Handle Joint Life Checkbox Selection  
  
Acceptance Criteria:  
1. When the "Joint Life" checkbox is checked:  
 - The system should set the `cn\_joint\_life` control to 'JL'.  
 - If the insured person's partner reference matches the policyholder's partner reference, the system should:  
 - Set the "Joint Life" checkbox to unchecked.  
 - Set the `cn\_joint\_life` control to 'SL'.  
 - Display an error message: "For Joint Life IP and PH cannot be same".  
 - If the "Joint Life" checkbox remains checked:  
 - The system should make the "QUESTIONNAIRE\_PH" tab page visible.  
 - Enable the following fields:  
 - `ph\_Age\_proof`  
 - `ph\_Age\_prf\_Id`  
 - `ph\_height`  
 - `ph\_weight`  
 - `ph\_wt\_chg`  
 - Disable the `PH\_EQUAL\_INS` field.  
 - If the `Ch\_Package` field is not null and does not contain 'Joint', and the product definition is not one of the specified products, the system should:  
 - Clear the `Ch\_Package` field.  
 - Navigate to the "LOADINGS" block, clear it, and go to the first record.  
 - Navigate back to the "COVERS" block, clear it, and go to the first record.  
 - Display a warning message: "Please Select The Joint Life Package".  
2. When the "Joint Life" checkbox is unchecked:  
 - The system should set the `cn\_joint\_life` control to 'SL'.  
 - If the "QUESTIONNAIRE\_PH" tab page is visible, the system should make it invisible.  
 - Disable the following fields:  
 - `ph\_Age\_proof`  
 - `ph\_Age\_prf\_Id`  
 - `ph\_height`  
 - `ph\_weight`  
 - `ph\_wt\_chg`  
 - Enable the `PH\_EQUAL\_INS` field.  
3. If the global loading status is 'F', the system should set the `cn\_form\_stat` control to 'Y'.  
4. The system should navigate to the `CH\_PACKAGE` item after processing the checkbox change.  
  
Definition of Done:  
- The system correctly handles the enabling and disabling of fields based on the "Joint Life" checkbox selection.  
- Appropriate error and warning messages are displayed as per the acceptance criteria.  
- The system navigates to the correct items and blocks as specified.  
- All conditions and logic are implemented and tested to ensure they work as expected.

# Handle GPB\_OPTED Checkbox Selection

Type: COVERHEAD

Detailed description: As a user, I want the system to handle the selection of the "GPB\_OPTED" checkbox in the "COVERHEAD" section, so that the appropriate fields and properties are updated based on the selection.  
  
Acceptance criteria:  
1. When the "GPB\_OPTED" checkbox is checked:  
 - If the "Joint\_Life" field is set to 'T':  
 - Set the "cn\_joint\_life" control field to 'JL'.  
 - If the "IP\_PARTNER\_REF" of the insured person matches the "PH\_PARTNER\_REF" of the policy holder:  
 - Set the "Joint\_Life" field to 'F'.  
 - Set the "cn\_joint\_life" control field to 'SL'.  
 - Display an error message indicating that the insured person and policy holder cannot be the same for joint life.  
 - If the "QUESTIONNAIRE\_PH" tab page is not visible, make it visible.  
 - Enable the following fields in the "policy\_holder" section:  
 - "ph\_Age\_proof"  
 - "ph\_Age\_prf\_Id"  
 - "ph\_height"  
 - "ph\_weight"  
 - "ph\_wt\_chg"  
 - Disable the "PH\_EQUAL\_INS" field in the "policy\_holder" section.  
 - If the "Ch\_Package" field is not null and does not contain 'Joint', and the product definition is not one of the specified products, clear the "Ch\_Package" field and display a warning message to select the joint life package.  
 - If the "Joint\_Life" field is not set to 'T':  
 - Set the "cn\_joint\_life" control field to 'SL'.  
 - If the "QUESTIONNAIRE\_PH" tab page is visible, make it invisible.  
 - Disable the following fields in the "policy\_holder" section:  
 - "ph\_Age\_proof"  
 - "ph\_Age\_prf\_Id"  
 - "ph\_height"  
 - "ph\_weight"  
 - "ph\_wt\_chg"  
 - Enable the "PH\_EQUAL\_INS" field in the "policy\_holder" section.  
 - If the global "LOADING" variable is set to 'F', set the "cn\_form\_stat" control field to 'Y'.  
 - Navigate to the "CH\_PACKAGE" item.  
  
Definition of Done:  
- The system correctly updates the fields and properties based on the selection of the "GPB\_OPTED" checkbox.  
- Appropriate error and warning messages are displayed as per the conditions.  
- The visibility and enabled/disabled state of the relevant fields are correctly managed.  
- The navigation to the "CH\_PACKAGE" item is correctly handled.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include any direct database queries.

# Modify Insurance Offer

Type: COVERHEAD

Detailed description: As a user, I want to be able to modify an insurance offer by updating various details such as cover name, service tax, and premium amounts, so that I can ensure the offer reflects the most accurate and up-to-date information.  
  
Acceptance criteria:  
1. The system should calculate loading percentages and amounts for different categories (e.g., medical, occupation, special, residential) based on the existing records.  
2. The system should retrieve and display the cover name for riders and update the rider description and cover amount accordingly.  
3. The system should log the modification details for audit purposes.  
4. The system should update the insurance offer details, including product ID, sum assured, benefit term, premium term, premium amount, cover name, cover code, and GST.  
5. The system should calculate and apply the appropriate GST amounts based on the premium and other parameters.  
6. The system should handle exceptions and provide meaningful error messages if any issues occur during the modification process.  
7. The system should update the counter-offer reasons if they are not already provided.  
8. The system should display the modified history of the offer for reference.  
  
Definition of Done:  
- The user can successfully modify an insurance offer by updating the necessary details.  
- The system accurately calculates and updates loading percentages and amounts.  
- The system retrieves and displays the correct cover names and updates rider descriptions and cover amounts.  
- All modification details are logged for audit purposes.  
- The insurance offer details are updated correctly, including GST calculations.  
- The system handles exceptions gracefully and provides meaningful error messages.  
- Counter-offer reasons are updated if not already provided.  
- The modified history of the offer is displayed for reference.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Retrieve cover name for riders:  
 ```sql  
 SELECT COVER\_NAME  
 INTO v\_rider\_desc  
 FROM cfg\_v\_prod\_covers\_api a  
 WHERE a.product\_id = :control.cn\_product\_id AND a.cover\_code = :azbj\_revised\_offer.cover\_code;  
 ```  
  
- Retrieve service tax date:  
 ```sql  
 SELECT v\_doc := azbj\_pk0\_acc.get\_doc(:control.cn\_contract\_id);  
 ```  
  
- Retrieve GST amount:  
 ```sql  
 azbj\_pk\_gst.get\_gst\_amount (  
 p\_product\_id => :azbj\_revised\_offer.product\_id,  
 p\_cover\_code => :azbj\_revised\_offer.cover\_code,  
 p\_premium => :AZBJ\_CO\_DETAILS.PREMIUM,  
 p\_effectve\_date => NVL (v\_service\_tax\_date, pme\_api.opus\_date),  
 p\_policy\_year => 1,  
 p\_mailadd\_pin\_code => :policy\_holder.ph\_mailing\_postcode,  
 p\_servadd\_pin\_code => azbj\_pk\_gst.get\_coll\_branch\_pincode(:control.cn\_branch\_code),  
 p\_sumassured => :azbj\_revised\_offer.sum\_assured,  
 p\_event\_code => 'PREMIUM',  
 p\_error => v\_error,  
 p\_gst\_dtl\_string => v\_azbj\_gst\_tab  
 );  
 ```  
  
- Retrieve counter-offer reason:  
 ```sql  
 SELECT c\_offer\_reason  
 INTO :Coverhead.COUNTER\_OFFER\_REASON  
 FROM azbj\_revised\_offer  
 WHERE policy\_ref = :control.cn\_policy\_ref  
 AND event\_type = 'NEW\_OFFER'  
 AND cover\_code LIKE 'L%'  
 AND activity\_no = (SELECT MAX (activity\_no)  
 FROM azbj\_revised\_offer  
 WHERE policy\_ref = :control.cn\_policy\_ref  
 AND event\_type = 'NEW\_OFFER'  
 AND cover\_code LIKE 'L%');  
 ```  
  
- Retrieve modified offer details:  
 ```sql  
 SELECT CO\_REASON1, CO\_REASON2, CO\_REASON3  
 INTO v\_reason\_1, v\_reason\_2, v\_reason\_3  
 FROM azbj\_modified\_offer a  
 WHERE policy\_ref = :control.cn\_policy\_ref  
 AND activity\_date = (SELECT MAX (activity\_date)  
 FROM azbj\_modified\_offer a  
 WHERE policy\_ref = :control.cn\_policy\_ref)  
 AND top\_indicator = 'Y'  
 AND ROWNUM = 1;  
 ```  
  
- Retrieve modified offer history:  
 ```sql  
 SELECT   
 FROM azbj\_modified\_offer  
 WHERE policy\_ref = :control.cn\_policy\_ref  
 ORDER BY ACTIVITY\_DATE;  
 ```

# Efficient Navigation and Field Display in Medical Underwriting Records

Type: W\_NR\_MED

Title: Efficient Navigation and Field Display in Medical Underwriting Records  
  
Acceptance Criteria:  
1. When navigating to the previous record, the system should move to the previous record in the dataset.  
2. When navigating to the next record:  
 - If the policy holder's equal insurance status is 1, the system should move to the next record and set the insurance type to 1.  
 - Otherwise, the system should simply move to the next record.  
  
Definition of Done:  
1. The navigation functionality should be implemented and tested to ensure it works as described.  
2. Fields such as "Test No", "Life", "Recd (W/NR)", "Comments", "Date Waived", "Waived By", and "Partner Id" should be displayed with the appropriate properties (e.g., enabled, visible, editable).  
3. The system should handle the navigation logic correctly, updating fields as necessary based on the context.  
4. The user interface should be intuitive and user-friendly, with clear prompts and labels for each field.  
5. All acceptance criteria should be met, and the functionality should be verified through testing.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Not applicable as the provided XML content does not include any direct database operations or queries.

# Enforce Minimum Length and Tooltip for Comments Field

Type: W\_NR\_MED

Detailed description: As a user, I want to ensure that the "Comments" field in the "W\_NR\_MED" section of the form captures meaningful input by enforcing a minimum character length and providing helpful tooltips.  
  
Acceptance criteria:  
1. The "Comments" field should not accept input less than 10 characters.  
2. If the input in the "Comments" field is less than 10 characters, an error message should be displayed: "Comments should not be less than 10 characters."  
3. When the mouse pointer enters, moves over, or double-clicks the "Comments" field, a tooltip should be displayed with the content from the "FURTHER\_REQ.COMMENTS" field.  
  
Definition of Done:  
- The "Comments" field enforces a minimum character length of 10.  
- An appropriate error message is displayed if the input is less than 10 characters.  
- Tooltips are correctly displayed when interacting with the "Comments" field, showing the content from the "FURTHER\_REQ.COMMENTS" field.  
- All functionalities are tested and verified to work as expected.  
  
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.

# Automatic Insurance Policy Type Setting

Type: W\_NR\_MED

Title: Automatic Insurance Policy Type Setting  
  
Acceptance Criteria:  
- When the product ID is one of the following values: 3, 4, 5, 9, or 10, the insurance policy type should be automatically set to 1.  
- The system should not set the insurance policy type for any other product IDs.  
  
Definition of Done:  
- The functionality is implemented and tested to ensure that the insurance policy type is correctly set based on the specified product IDs.  
- Unit tests are created to cover the scenarios where the product ID is within the specified range and outside of it.  
- The feature is reviewed and approved by the product owner.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided logic does not involve direct database CRUD operations.

# Interaction with 'Test No' Field in New Business Module

Type: W\_NR\_MED

Title: Interaction with 'Test No' Field in New Business Module  
  
Acceptance Criteria:  
1. The 'Test No' field should be displayed on the 'New Business' tab.  
2. The field should be disabled by default and should not allow user input.  
3. When the 'Test No' field is clicked, it should navigate to the same field.  
4. Upon entering the 'Test No' field, the system should automatically populate the 'Date Called' field with the current date and a specific date from an external API.  
  
Definition of Done:  
1. The 'Test No' field is visible and disabled by default.  
2. Clicking on the 'Test No' field navigates to the same field.  
3. Entering the 'Test No' field triggers the population of the 'Date Called' field with the current date and a date from an external API.  
4. All functionalities are tested and verified to work as expected.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include any direct database operations.

# Handle Receipt Status of Medical Tests

Type: W\_NR\_MED

Detailed description: As a user, I want the system to handle the receipt status of medical tests so that the appropriate actions are taken based on whether the test results have been received or not.  
  
Acceptance criteria:  
1. If the test is conducted through a home visit, the system should prompt the user with a confirmation alert.  
2. If the user confirms that the test is conducted through a home visit, the system should allow the user to proceed. If the user cancels, the system should reset the test conducted from the field to null.  
3. If the receipt status is 'Y' (Yes), the system should:  
 - Set the receipt date to the current date if it is not already set.  
 - Update the receipt date in the system variables.  
4. If the receipt status is 'N' (No), the system should:  
 - Clear the receipt date.  
 - Update the system variables to indicate that the receipt date is not set.  
 - Set specific global flags to 'N'.  
5. When the receipt status is validated:  
 - If the status is 'Y', the receipt date should be set to the current date if it is not already set.  
 - If the status is 'N', the receipt date should be cleared, and specific global flags should be set to 'N'.  
6. When the receipt status changes:  
 - If the status is 'Y' or 'N', the system should update the user who raised the medical request.  
7. When navigating through records:  
 - If the receipt status is 'Y' or 'N', the system should call a specific procedure to handle waived requirements and then delete the current record.  
 - If the receipt status is not 'Y' or 'N', the system should move to the next or previous record.  
  
Definition of Done:  
- The system correctly prompts the user for confirmation when the test is conducted through a home visit.  
- The system updates the receipt date and system variables based on the receipt status.  
- The system correctly handles the validation and change of receipt status.  
- The system navigates through records and handles waived requirements as specified.  
- All functionalities are tested and verified to work as expected.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Not applicable as the provided XML content does not include direct database queries that can be executed independently of Oracle Forms constructs.

# Manage Further Requirements for Policy Holder

Type: W\_NR\_FUR

Title: Manage Further Requirements for Policy Holder  
  
Acceptance Criteria:  
1. The system should display the following fields for further requirements:  
 - Request Number  
 - Description  
 - Type  
 - Comments  
 - Received Status  
 - Date Waived  
 - Waived By  
2. The "Request Number" field should be validated against a predefined list of values.  
3. The "Type" field should be a list item with a maximum length of 2 characters and should be restricted to uppercase input.  
4. The "Comments" field should allow a maximum length of 500 characters and should be restricted to uppercase input.  
5. The "Received Status" field should allow a maximum length of 2 characters and should be restricted to uppercase input.  
6. The "Date Waived" field should be a date field and should be disabled for user input.  
7. The "Waived By" field should allow a maximum length of 20 characters and should be restricted to uppercase input.  
8. Navigation between records should be possible using the up and down arrow keys.  
9. When navigating down, if the policy holder's equal insurance status is 1, the "Type" field should be set to 1.  
  
Definition of Done:  
- The user interface displays all the required fields with the specified properties.  
- The "Request Number" field is validated against the list of values.  
- The "Type", "Comments", and "Received Status" fields enforce uppercase input and their respective maximum lengths.  
- The "Date Waived" field is displayed but disabled for user input.  
- The "Waived By" field enforces uppercase input and its maximum length.  
- Navigation between records works as specified, including the conditional logic for setting the "Type" field.  
- All acceptance criteria are met and tested.  
  
DB queries for Table reference CRUD operations only (With Usage):  
```sql  
SELECT sys\_code AS DOC, UPPER(TRIM(sys\_desc)) AS DESCRIPTION  
FROM azbj\_system\_constants  
WHERE sys\_type = 'FR\_REQ'  
AND NVL(TO\_DATE(:susac.sa\_daterecd, 'DD/MM/RRRR'), pme\_api.opus\_date) BETWEEN start\_date AND NVL(end\_date, TO\_DATE('01-jan-3000', 'dd/mm/rrrr'))  
ORDER BY DESCRIPTION;  
```  
- This query is used to populate the list of values for the "Request Number" field.

# Dropdown Menu for Insurance Policy Type Selection

Type: W\_NR\_FUR

Title: Dropdown Menu for Insurance Policy Type Selection  
  
Acceptance Criteria:  
1. The list of insurance policy types should be displayed in a dropdown menu.  
2. The dropdown menu should be positioned appropriately on the user interface.  
3. The dropdown menu should be navigable using the keyboard.  
4. The selected policy type should be stored in uppercase format.  
5. The dropdown menu should have a maximum length of 2 characters for each policy type.  
6. The dropdown menu should be labeled "Life" and the label should be bold and aligned to the center.  
7. The dropdown menu should be available on the "MED\_UW" tab page.  
  
Definition of Done:  
1. The dropdown menu for selecting the insurance policy type is implemented and visible on the user interface.  
2. The dropdown menu is functional and allows users to select from a predefined list of policy types.  
3. The selected policy type is stored in uppercase format.  
4. The dropdown menu is keyboard navigable.  
5. The label "Life" is displayed correctly and is bold and center-aligned.  
6. The dropdown menu is accessible on the "MED\_UW" tab page.  
7. All acceptance criteria are met and tested successfully.

# Requirement Number Selection and Date Population

Type: W\_NR\_FUR

Title: Requirement Number Selection and Date Population  
  
Acceptance Criteria:  
1. When the user clicks on the "Req No." field, the system should navigate to the "Req No." field.  
2. Upon entering the "Req No." field, the system should automatically populate the current date and a specific date from an external API into the respective fields.  
3. The "Req No." field should display a list of values (LOV) containing requirement numbers and their descriptions, filtered based on specific date criteria.  
  
Definition of Done:  
- The "Req No." field is not editable directly by the user.  
- The system navigates to the "Req No." field upon clicking.  
- The current date and the date from the external API are populated correctly when the user enters the "Req No." field.  
- The LOV for the "Req No." field displays the correct requirement numbers and descriptions based on the specified date criteria.  
  
DB queries for Table reference CRUD operations only(With Usage):  
```sql  
SELECT sys\_code AS DOC, UPPER(TRIM(sys\_desc)) AS DESCRIPTION  
FROM azbj\_system\_constants  
WHERE sys\_type = 'FR\_REQ'  
AND NVL(TO\_DATE(:susac.sa\_daterecd, 'DD/MM/RRRR'), pme\_api.opus\_date) BETWEEN start\_date AND NVL(end\_date, TO\_DATE('01-jan-3000', 'dd/mm/rrrr'))  
ORDER BY DESCRIPTION;  
```  
- This query is used to fetch the requirement numbers and their descriptions for the LOV, filtered by the specified date criteria.

# Validate and Provide Tooltip for Comments Field

Type: W\_NR\_FUR

User Story:  
As a user, I want to ensure that the comments entered in the "Comments" field are validated to be at least 10 characters long, so that meaningful feedback is provided. Additionally, I want to see a tooltip with additional information when interacting with the "Comments" field.  
  
Acceptance Criteria:  
1. If the "Comments" field is not empty, the system should check the length of the comments.  
2. If the length of the comments is less than 10 characters, an error message should be displayed: "Comments should not be less than 10 characters."  
3. The "Comments" field should display a tooltip with additional information when the mouse pointer enters, moves over, or double-clicks the field.  
  
Definition of Done:  
1. The validation logic for the "Comments" field is implemented and tested.  
2. The error message is displayed correctly when the comments are less than 10 characters.  
3. The tooltip functionality is implemented and tested for mouse enter, move, and double-click events.  
4. All acceptance criteria are met and verified through testing.  
5. The feature is reviewed and approved by stakeholders.  
  
SQL Query for Reference:  
N/A

# Handle Input for FR\_RES\_RECD Field

Type: W\_NR\_FUR

Title: Handle Input for FR\_RES\_RECD Field  
  
Acceptance Criteria:  
1. When the user navigates to the next item, the system should set the "fr\_date\_called" field to the current date.  
2. When the "FR\_RES\_RECD" field is changed, the system should:  
 - Check if the field value is either 'Y' or 'N'.  
 - If the value is valid, set the "fr\_req\_raised\_by" field to the current user's ID.  
 - If an error occurs, display a warning message.  
3. When the user presses the down arrow key:  
 - If the "FR\_RES\_RECD" field is not null and contains 'N' or 'Y', call the "azbj\_waived\_requirements" procedure with the appropriate parameters and delete the current record.  
 - If the field is null or contains an invalid value, move to the next record.  
4. When the user presses the up arrow key:  
 - If the "FR\_RES\_RECD" field is not null and contains 'N' or 'Y', call the "azbj\_waived\_requirements" procedure with the appropriate parameters and delete the current record.  
 - If the field is null or contains an invalid value, move to the previous record.  
  
Definition of Done:  
- The system correctly handles the input for the "FR\_RES\_RECD" field as per the acceptance criteria.  
- The system performs the necessary actions and validations without any errors.  
- The user is able to navigate through records and trigger the appropriate procedures based on the input values.  
- All error messages and warnings are displayed correctly to the user.

# Manage Further Requirements for Policy Holder

Type: FURTHER\_REQ

Title: Manage Further Requirements for Policy Holder  
  
Acceptance Criteria:  
1. The system should allow the user to add a new requirement with details such as requirement number, description, date received, and comments.  
2. The system should allow the user to delete an existing requirement.  
3. The system should enable the user to update the status of a requirement, including marking it as received and recording the date it was received.  
4. The system should validate the requirement number against a predefined list of valid requirements.  
5. The system should display a list of the top 20 frequent requirements for quick selection.  
6. The system should allow the user to add comments and remarks for each requirement.  
7. The system should ensure that the requirement details are displayed in a vertical orientation with a scrollbar for navigation.  
8. The system should handle specific conditions such as enabling or disabling fields based on user actions and predefined rules.  
  
Definition of Done:  
1. The user can successfully add, delete, and update requirement details.  
2. The system validates the requirement number and displays appropriate error messages if the number is invalid.  
3. The top 20 frequent requirements are displayed for quick selection.  
4. All fields related to the requirement are displayed and navigable as per the user actions.  
5. The system handles specific conditions and rules as defined in the acceptance criteria.  
6. The user interface is intuitive and user-friendly, allowing for efficient management of further requirements.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- The system should use the following query to fetch the list of valid requirements:  
 ```sql  
 SELECT sys\_code DOC, upper(trim(sys\_desc)) DESCRIPTION   
 FROM azbj\_system\_constants   
 WHERE sys\_type = 'FR\_REQ'   
 AND nvl(to\_date(:susac.sa\_daterecd,'DD/MM/RRRR'),pme\_api.opus\_date) between start\_date and nvl(end\_date,to\_date('01-jan-3000','dd/mm/rrrr'))  
 ORDER BY DESCRIPTION;  
 ```  
  
- The system should use the following query to fetch the list of frequent requirements for PhonePe:  
 ```sql  
 SELECT sys\_code DOC, UPPER (TRIM (sys\_desc)) DESCRIPTION  
 FROM azbj\_system\_constants  
 WHERE sys\_type = 'FR\_REQ'  
 AND sys\_code IN ('M583', 'M200', 'M543', 'M017', 'M755', 'm073', 'M363', 'M268', 'M106', 'M253', 'M202')  
 AND NVL (TO\_DATE (:susac.sa\_daterecd, 'DD/MM/RRRR'), pme\_api.opus\_date) BETWEEN start\_date AND NVL (end\_date, TO\_DATE ('01-jan-3000', 'dd/mm/rrrr'))  
 ORDER BY DESCRIPTION;  
 ```

# Validate Remarks Field in Further Requirements Section

Type: FURTHER\_REQ

Title: Validate Remarks Field in Further Requirements Section  
  
Acceptance Criteria:  
1. If the remarks field is not empty, it should contain at least 10 characters.  
2. If the remarks field contains fewer than 10 characters, an error message should be displayed: "Comments should not be less than 10 characters."  
  
Definition of Done:  
- The remarks field validation logic is implemented.  
- The error message is displayed correctly when the validation fails.  
- The validation logic is tested and confirmed to work as expected.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include any database-specific operations.

# Secondary FRAR Request Button Functionality

Type: FURTHER\_REQ

Title: Secondary FRAR Request Button Functionality  
  
Acceptance Criteria:  
1. When the user presses the button labeled "2nd + FRAR Req.", the system should check if the primary FRAR count is greater than the secondary FRAR count.  
2. If the primary FRAR count is greater, the system should populate the secondary FRAR details.  
  
Definition of Done:  
1. The button labeled "2nd + FRAR Req." is functional and triggers the appropriate logic.  
2. The system correctly checks the primary and secondary FRAR counts.  
3. The secondary FRAR details are populated if the primary FRAR count exceeds the secondary FRAR count.  
4. The functionality 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 specific database queries.

# Add Comments Field in Further Requirements Section

Type: FURTHER\_REQ

Title: Add Comments Field in Further Requirements Section  
  
Acceptance Criteria:  
1. The comments field should accept up to 500 characters.  
2. The comments should be displayed in uppercase.  
3. The comments field should not be keyboard navigable.  
4. The comments field should have a tooltip that displays the current content of the comments when the mouse enters, moves over, or double-clicks the field.  
5. The comments field should be associated with a list of values (LOV) that fetches data from the discrepancy codes table based on specific conditions.  
  
Definition of Done:  
1. The comments field is implemented and visible in the "Further Requirements" section.  
2. The field accepts up to 500 characters and displays them in uppercase.  
3. The tooltip functionality works as expected when the mouse interacts with the field.  
4. The LOV is correctly associated and fetches data based on the specified conditions.  
5. All acceptance criteria are met and tested.  
  
DB queries for Table reference CRUD operations only(With Usage):  
```sql  
SELECT LIST\_OF\_VALUES  
FROM AZBJ\_DISCREPANCY\_LOVS a, AZBJ\_DISCREPANCY\_CODES b  
WHERE b.SYS\_TYPE = 'FR\_REQ'  
AND b.REQ\_CODE = :FURTHER\_REQ.FR\_TESTNO  
AND b.MAIN\_REMARKS = a.MAIN\_REMARKS;  
```  
This query is used to fetch the list of values for the comments field based on the discrepancy codes.

# Populate Top 20 Frequent Requirements

Type: FURTHER\_REQ

Detailed description: As a user, I want to populate the top 20 frequent requirements in the system so that I can quickly access and review the most common requirements.  
  
Acceptance criteria:  
1. When the "POPULATE TOP 20 REQS" button is pressed, the system should navigate to the "Further Requirements" section.  
2. The system should clear any existing data in the "Further Requirements" section.  
3. The system should retrieve the top 20 frequent requirements from the database where the system type is 'FR\_REQ' and the character value is 'TOP20'.  
4. For each retrieved requirement, the system should populate the following fields:  
 - Requirement Number  
 - Requirement Description  
 - IP Type (set to 1)  
 - Date Called (set to the current date)  
5. The system should iterate through the records and populate them sequentially.  
6. After populating the records, the system should navigate to the first record.  
  
Definition of Done:  
- The "POPULATE TOP 20 REQS" button successfully populates the top 20 frequent requirements in the "Further Requirements" section.  
- All specified fields are correctly populated for each requirement.  
- The system navigates to the first record after populating the data.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- The query to retrieve the top 20 frequent requirements:  
 ```sql  
 SELECT sys\_code AS DOC, sys\_desc AS DESCRIPTION  
 FROM azbj\_system\_constants  
 WHERE sys\_type = 'FR\_REQ'  
 AND char\_value = 'TOP20';  
 ```  
- The current date can be retrieved using:  
 ```sql  
 SELECT pme\_api.opus\_date FROM dual;  
 ```

# Manage RCU Requirements

Type: FURTHER\_REQ

Detailed description: As a user, I want to be able to view and manage the RCU requirements for an insured person, so that I can ensure all necessary documentation is tracked and updated.  
  
Acceptance criteria:  
1. When the "RCU Requirements" button is pressed, the system should:  
 - Retrieve distinct alternate requirements from the `azbj\_phub\_add\_req\_tracker` table where `accept\_req` is 'N' and the `application\_no` matches the insured person's verification or sign card number.  
 - Navigate to the "DOCS" section and clear any existing records.  
 - For each retrieved requirement, populate the `REQ\_TYPE` field in the "DOCS" section.  
 - Retrieve the system description for each requirement from the `azbj\_system\_constants` table where `sys\_TYPE` is 'FR\_REQ' and `SYS\_CODE` matches the alternate requirement, and populate the `REQ\_DESC` field in the "DOCS" section.  
 - Handle any exceptions by setting the `REQ\_DESC` field to an empty string if the system description is not found.  
 - Display the "DOCS" window and set focus to the `REQ\_TYPE` field.  
  
Definition of Done:  
- The "RCU Requirements" button functionality is implemented and tested.  
- The system correctly retrieves and displays the alternate requirements and their descriptions.  
- The "DOCS" window is displayed with the focus set to the `REQ\_TYPE` field.  
- All acceptance criteria are met and verified through testing.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Retrieve distinct alternate requirements:  
 ```sql  
 SELECT DISTINCT alternate\_req   
 FROM azbj\_phub\_add\_req\_tracker  
 WHERE accept\_req = 'N'  
 AND application\_no = :application\_no;  
 ```  
  
- Retrieve system description:  
 ```sql  
 SELECT DISTINCT SYS\_DESC   
 INTO :docs.req\_desc   
 FROM azbj\_system\_constants   
 WHERE sys\_TYPE = 'FR\_REQ'  
 AND SYS\_CODE = :alternate\_req;  
 ```

# Automatic IP Type Setting and Validation

Type: FURTHER\_REQ

Title: Automatic IP Type Setting and Validation  
  
Acceptance Criteria:  
1. When the product ID is one of the following: 3, 4, 5, 9, 10, 12, 14, 16, 20, 23, 24, 25, 31, 32, 33, 34, 37, 49, 50, the system should automatically set the "IP Type" to 1.  
2. If the policy holder and insured person are the same (indicated by a specific flag), and the "IP Type" is not set to 1, the system should display a warning message: "For IP And PH Same Case IP Type Should Be In LA Only IN FURTHER\_REQ TAB".  
3. The system should update a specific variable to indicate whether the "IP Type" validation was required.  
  
Definition of Done:  
- The system correctly sets the "IP Type" to 1 for the specified product IDs.  
- The system displays the appropriate warning message when the policy holder and insured person are the same, and the "IP Type" is not set to 1.  
- The specific variable indicating the validation requirement is updated accordingly.  
- All changes are tested and verified to ensure they meet the acceptance criteria.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Not applicable as the provided logic does not include direct database CRUD operations.

# Add Trustee/Beneficiary Functionality

Type: FURTHER\_REQ

Title: Add Trustee/Beneficiary Functionality  
  
Acceptance Criteria:  
1. When the "Add Trustee/Beneficiary" button is pressed, the system should check if the contract has the "MWP" (Married Women's Property) flag set to 'Y'.  
2. If the "MWP" flag is 'Y', the system should:  
 - Navigate to the "MWP\_ACT" section.  
 - Clear any existing records in the "MWP\_ACT" section.  
 - Retrieve all records from the `customer.azbj\_beneficiary\_trustee\_rep` table where the `contract\_id` matches the current contract.  
 - For each retrieved record, populate the "MWP\_ACT" section with the following details:  
 - Serial number  
 - Beneficiary name  
 - Date of birth  
 - Share percentage  
 - Relationship  
 - Trustee name  
 - Trustee address  
 - Sign date  
 - Witness name  
 - Ensure the "MWP\_ACT" section is updated with the current date if the sign date is not available.  
 - Set focus on the "Beneficiary Name" field in the "MWP\_ACT" section.  
  
Definition of Done:  
- The "Add Trustee/Beneficiary" button functionality is implemented and tested.  
- The system correctly navigates to the "MWP\_ACT" section and populates it with the relevant details from the database.  
- The "MWP\_ACT" section is updated with the current date if the sign date is not available.  
- The focus is set on the "Beneficiary Name" field after the section is populated.  
- All acceptance criteria are met and verified through testing.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Retrieve records from `customer.azbj\_beneficiary\_trustee\_rep` where `contract\_id` matches the current contract:  
 ```sql  
 SELECT FROM customer.azbj\_beneficiary\_trustee\_rep  
 WHERE contract\_id = :control.cn\_contract\_id;  
 ```

# Handle 'Further Requirements' Field

Type: FURTHER\_REQ

Title: Handle 'Further Requirements' Field  
  
Acceptance Criteria:  
1. When the 'Further Requirements' field is clicked, the system should navigate to the same field.  
2. When the 'Further Requirements' field is accessed, it should automatically populate the current date and a specific date from an external API.  
3. When the 'Further Requirements' field is double-clicked:  
 - If a global condition related to rule confidence is not met and a specific item is visible, the system should display an error message instructing the user to raise requirements for the failed confidence rule first.  
 - The field should be enabled.  
 - Depending on a specific flag, the system should set the list of values (LOV) for the field to either a general or a specific LOV.  
 - The system should reset certain fields and navigate to a specific item.  
 - If an error occurs, the system should display the error message.  
  
Definition of Done:  
- The 'Further Requirements' field should behave as described in the acceptance criteria.  
- The system should handle all specified conditions and provide appropriate feedback to the user.  
- The functionality should be tested and verified to ensure it meets the requirements.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- The LOV 'AZBJ\_FR\_REQ\_LOV' should fetch data using the query:  
 ```sql  
 SELECT sys\_code DOC, UPPER(TRIM(sys\_desc)) DESCRIPTION  
 FROM azbj\_system\_constants  
 WHERE sys\_type = 'FR\_REQ'  
 AND NVL(TO\_DATE(:susac.sa\_daterecd, 'DD/MM/RRRR'), pme\_api.opus\_date) BETWEEN start\_date AND NVL(end\_date, TO\_DATE('01-jan-3000', 'dd/mm/rrrr'))  
 ORDER BY DESCRIPTION;  
 ```  
  
- The LOV 'AZBJ\_FR\_REQ\_PHONEPE' should fetch data using the query:  
 ```sql  
 SELECT sys\_code DOC, UPPER(TRIM(sys\_desc)) DESCRIPTION  
 FROM azbj\_system\_constants  
 WHERE sys\_type = 'FR\_REQ'  
 AND sys\_code IN ('M583', 'M200', 'M543', 'M017', 'M755', 'm073', 'M363', 'M268', 'M106', 'M253', 'M202')  
 AND NVL(TO\_DATE(:susac.sa\_daterecd, 'DD/MM/RRRR'), pme\_api.opus\_date) BETWEEN start\_date AND NVL(end\_date, TO\_DATE('01-jan-3000', 'dd/mm/rrrr'))  
 ORDER BY DESCRIPTION;  
 ```

# Add New Requirement Button Functionality

Type: FURTHER\_REQ

Title: Add New Requirement Button Functionality  
  
Acceptance Criteria:  
1. When the button is pressed, if the global rule confidence window is not open and the rule confidence window is visible, an error message should be displayed instructing the user to first raise requirements for the failed confidence rule through the rule confidence window.  
2. The field for entering the test number should be enabled.  
3. If the PhonePe flag is set to 'Y', the list of values (LOV) for the test number should be set to 'AZBJ\_FR\_REQ\_PHONEPE'. Otherwise, it should be set to 'AZBJ\_FR\_REQ\_LOV'.  
4. The requirement type should be set to 'FR-REQ'.  
5. The field for who raised the requirement should be set to 'USER CALLED'.  
6. The focus should move to the test number field.  
7. The test number and test description fields should be cleared.  
  
Definition of Done:  
- The button functionality should be implemented as per the acceptance criteria.  
- The error message should be displayed correctly when the conditions are met.  
- The appropriate LOV should be set based on the PhonePe flag.  
- The requirement type and raised by fields should be set correctly.  
- The focus should move to the test number field, and the test number and test description fields should be cleared.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Query for 'AZBJ\_FR\_REQ\_LOV':  
 ```sql  
 SELECT sys\_code DOC, UPPER(TRIM(sys\_desc)) DESCRIPTION  
 FROM azbj\_system\_constants  
 WHERE sys\_type = 'FR\_REQ'  
 AND NVL(TO\_DATE(:susac.sa\_daterecd, 'DD/MM/RRRR'), pme\_api.opus\_date) BETWEEN start\_date AND NVL(end\_date, TO\_DATE('01-jan-3000', 'dd/mm/rrrr'))  
 ORDER BY DESCRIPTION;  
 ```  
  
- Query for 'AZBJ\_FR\_REQ\_PHONEPE':  
 ```sql  
 SELECT sys\_code DOC, UPPER(TRIM(sys\_desc)) DESCRIPTION  
 FROM azbj\_system\_constants  
 WHERE sys\_type = 'FR\_REQ'  
 AND sys\_code IN ('M583', 'M200', 'M543', 'M017', 'M755', 'm073', 'M363', 'M268', 'M106', 'M253', 'M202')  
 AND NVL(TO\_DATE(:susac.sa\_daterecd, 'DD/MM/RRRR'), pme\_api.opus\_date) BETWEEN start\_date AND NVL(end\_date, TO\_DATE('01-jan-3000', 'dd/mm/rrrr'))  
 ORDER BY DESCRIPTION;  
 ```

# Delete Requirement Button Functionality

Type: FURTHER\_REQ

Title: Delete Requirement Button Functionality  
  
Acceptance Criteria:  
1. When the "Delete Requirement" button is pressed, the system should:  
 - Navigate to the "Further Requirements" section.  
 - Identify the current record in the "Further Requirements" section.  
 - Loop through all records in the "Further Requirements" section to perform various checks based on the requirement type (`fr\_testno`).  
 - Validate if the requirement is mandatory based on different conditions such as relationship with the premium payer, agent state differences, product-specific requirements, etc.  
 - If any validation fails, display an appropriate error message and prevent the deletion.  
 - If all validations pass, delete the record from the "Further Requirements" section.  
  
Definition of Done:  
- The "Delete Requirement" button functionality is implemented and tested.  
- All validations and conditions are correctly checked before deletion.  
- Appropriate error messages are displayed if any validation fails.  
- The record is successfully deleted if all validations pass.  
- The functionality is tested and verified in the user interface.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Not applicable as the provided XML content does not include direct database queries that can be executed independently of Oracle Forms constructs.

# Validate and Record Receipt Status for Tests

Type: FURTHER\_REQ

Title: Validate and Record Receipt Status for Tests  
  
Acceptance Criteria:  
1. If the agent code is '2000003085' and the test number is one of the specified values ('M583', 'M200', 'M543', 'M017', 'M755', 'm073', 'M363', 'M268', 'M106', 'M253', 'M202') and the receipt status is 'Y' and the status flag is 'Y':  
 - Retrieve the user ID and received status from the `wip\_azbj\_med\_uw` table where the contract ID and test number match.  
 - If the current user is not the same as the user who raised the request and the receipt status is 'Y', set the receipt status to 'N' and display an error message indicating that the request cannot be marked as received by another user manually.  
2. If the receipt status is 'Y', set the received date to the current date and update the received date variable. If the receipt status is not 'Y', clear the received date and set the clean flag to 'N'.  
3. If the test number is 'M393' and the receipt status is 'N':  
 - Navigate to the 'SEC\_FRAR\_REQ' block, iterate through the records, and check if the supervisor comments are 'Agree'. If found, update the corresponding fields in the 'SEC\_FRAR\_REQ' block.  
4. If the test number is 'M592' and the receipt status is 'N':  
 - Check if there are active questions for the policy and contract in the `azbj\_crm\_questions\_detail` table. If no active questions are found, navigate to the 'AZBJ\_UW\_QUESTIONS.QUESTION\_DESC' item and display an error message to enter underwriting requirement questions.  
5. If the receipt status is 'Y' and the test number is not 'M548':  
 - Check if the test number exists in the `azbj\_system\_constants` table with specific conditions. If found, check if the application number exists in the `azbj\_cv\_approval\_cases` table and if it is approved. If not approved, set the receipt status to 'N'.  
6. If the receipt status is 'Y' and the test number is 'M675' and the agent code is not '2000002995':  
 - Check if the consent information is 'NR'. If true, set the receipt status to 'N' and display an error message indicating that the request cannot be marked as received due to the consent information.  
7. If the receipt status is 'Y' and the test number is 'M675' and the agent code is not '2000002995':  
 - Set the received flag for 'M675' to 'Y'.  
  
Definition of Done:  
- The receipt status is accurately recorded and validated based on the specified conditions.  
- Appropriate error messages are displayed when conditions are not met.  
- The received date and other related fields are updated correctly based on the receipt status.  
- The system navigates to the appropriate blocks and items as required by the conditions.  
- All database queries and updates are executed successfully without errors.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Retrieve user ID and received status:  
 ```sql  
 SELECT REQ\_RASIED\_BY, NVL(RECDSTAT, 'N')  
 INTO v\_user\_id, v\_recvd\_status  
 FROM wip\_azbj\_med\_uw  
 WHERE contract\_id = :control.cn\_contract\_id  
 AND testno = :further\_req.fr\_testno;  
 ```  
  
- Check for active questions:  
 ```sql  
 SELECT COUNT()  
 INTO v\_question\_cnt  
 FROM azbj\_crm\_questions\_detail  
 WHERE policy\_no = :control.cn\_policy\_ref  
 AND contract\_id = :control.cn\_contract\_id  
 AND active\_questions = 'Y';  
 ```  
  
- Check for test number in system constants:  
 ```sql  
 SELECT COUNT(1)  
 INTO v\_count  
 FROM azbj\_system\_constants  
 WHERE sys\_type = 'CENTRALIZED\_CALLING'  
 AND sys\_desc <> 'CENTRALIZED\_CALLING'  
 AND char\_value = :further\_req.fr\_testno;  
 ```  
  
- Check for application number in approval cases:  
 ```sql  
 SELECT COUNT(1)  
 INTO v\_count  
 FROM azbj\_cv\_approval\_cases  
 WHERE application\_no = TO\_CHAR(NVL(:insured\_person.ip\_sign\_card\_no, :insured\_person.ip\_verf\_no));  
 ```  
  
- Check for approved status in approval cases:  
 ```sql  
 SELECT COUNT(1)  
 INTO v\_count  
 FROM azbj\_cv\_approval\_cases  
 WHERE approval\_status = 'APPROVED'  
 AND application\_no = TO\_CHAR(NVL(:insured\_person.ip\_sign\_card\_no, :insured\_person.ip\_verf\_no));  
 ```  
  
- Check for consent information:  
 ```sql  
 IF AZBJ\_BYPASS\_CONSENT(TO\_CHAR(NVL(:insured\_person.ip\_sign\_card\_no, :insured\_person.ip\_verf\_no)), pk\_vars.v\_consent\_partid, :agents.ag\_agent\_code, :control.cn\_product\_id) = 'N' THEN  
 ```  
  
- Set received flag for 'M675':  
 ```sql  
 IF :further\_req.fr\_res\_recd = 'Y' AND :further\_req.fr\_testno = 'M675' AND :AGENTS.AG\_AGENT\_CODE <> '2000002995' THEN  
 pk\_vars.v\_M675\_received := 'Y';  
 END IF;  
 ```

# Manage Second-Time FRAR Requirements

Type: SEC\_FRAR\_REQ

Title: Manage Second-Time FRAR Requirements  
  
Acceptance Criteria:  
1. The user should be able to view a list of second-time FRAR requirements with the following details:  
 - Serial Number (S)  
 - Request Number (Req No.)  
 - Description  
 - Life (IP Type)  
 - U/W Reason  
 - U/W Reason Display  
 - Supervisor ID  
 - Password  
 - Supervisor Comments  
  
2. The user should be able to navigate through the records using the up and down keys.  
  
3. The user should be able to input and save supervisor comments.  
  
4. The user should be able to authenticate the information by providing a supervisor ID and password.  
  
5. The user should have the option to save the entered data or exit the form.  
  
Definition of Done:  
- The form displays all required fields and allows navigation through records.  
- The form accepts and saves supervisor comments.  
- The form allows authentication using supervisor ID and password.  
- The form provides options to save the data or exit.  
- The form is tested and verified to ensure all functionalities work 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 or operations.

# Supervisor ID Field Functionality

Type: SEC\_FRAR\_REQ

Title: Supervisor ID Field Functionality  
  
Acceptance Criteria:  
- The Supervisor ID field should accept a maximum of 10 characters.  
- The input should be automatically converted to uppercase.  
- Upon exiting the Supervisor ID field, the system should apply a specific visual attribute to the field to indicate it has been processed.  
  
Definition of Done:  
- The Supervisor ID field is present and functional on the form.  
- The field accepts up to 10 characters and converts input to uppercase.  
- The visual attribute is correctly applied when the user exits 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.

# Secure Password Entry in Second time FRAR Requirements

Type: SEC\_FRAR\_REQ

Title: Secure Password Entry in Second time FRAR Requirements  
  
Acceptance Criteria:  
1. The password field should be labeled "Password".  
2. The password input should be concealed (i.e., characters should be hidden as they are typed).  
3. The password field should have a specific visual style to distinguish it from other fields.  
4. Upon exiting the password field, the visual style should be applied to ensure it remains distinct.  
  
Definition of Done:  
- The password field is implemented and labeled correctly.  
- The password input is concealed.  
- The visual style is applied to the password field.  
- The visual style remains applied after the user exits the password field.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Not applicable as the provided XML content does not include any database queries.

# Save Button Functionality

Type: SEC\_FRAR\_REQ

Title: Save Button Functionality  
  
Acceptance Criteria:  
1. When the "Save" button is pressed, the system should validate the current form data.  
2. After validation, the system should save the form data.  
3. If the field 'FURTHER\_REQ.SECND\_FRAR\_REQ' is enabled, the system should navigate to this field.  
4. If the field 'FURTHER\_REQ.SECND\_FRAR\_REQ' is not enabled, the system should navigate to the field 'FURTHER\_REQ.RCU'.  
  
Definition of Done:  
- The form data is validated and saved successfully.  
- The system navigates to the correct field based on the enabled status of 'FURTHER\_REQ.SECND\_FRAR\_REQ'.  
- The functionality is tested and verified to work as expected.

# Navigation based on item enabled status

Type: SEC\_FRAR\_REQ

Title: Navigation based on item enabled status  
  
Acceptance Criteria:  
1. When the "Exit" button is pressed, the system should check if the item 'FURTHER\_REQ.SECND\_FRAR\_REQ' is enabled.  
2. If 'FURTHER\_REQ.SECND\_FRAR\_REQ' is enabled, the system should navigate to this item.  
3. If 'FURTHER\_REQ.SECND\_FRAR\_REQ' is not enabled, the system should navigate to the item 'FURTHER\_REQ.RCU'.  
  
Definition of Done:  
- The navigation logic is implemented and tested.  
- The system correctly navigates to 'FURTHER\_REQ.SECND\_FRAR\_REQ' if it is enabled.  
- The system correctly navigates to 'FURTHER\_REQ.RCU' if 'FURTHER\_REQ.SECND\_FRAR\_REQ' is not enabled.  
- The functionality is verified through user acceptance testing.

# Supervisor Comments Input and Selection

Type: SEC\_FRAR\_REQ

Title: Supervisor Comments Input and Selection  
  
Acceptance Criteria:  
1. When the user double-clicks on the "Supervisor Comments" text field, a list of values (LOV) should be displayed.  
2. If the test number is 'M393', the LOV should display the options 'Agree' and 'Disagree'.  
3. For any other test number, the LOV should display the options 'Failure of the 1st Underwriter', 'Additional requirement', and 'Previously raised, not complied'.  
  
Definition of Done:  
1. The "Supervisor Comments" text field is enabled and allows input up to 500 characters.  
2. The LOV functionality is implemented and displays the correct options based on the test number.  
3. The user can select an option from the LOV, and the selected value is populated in the "Supervisor Comments" text field.  
4. The feature is tested and verified to work as expected.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- SELECT 'Agree ' sup\_comm FROM DUAL UNION ALL SELECT 'Disagree' sup\_comm FROM DUAL;  
- SELECT 'Failure of the 1st Underwriter' sup\_comm FROM DUAL UNION ALL SELECT 'Additional requirement' sup\_comm FROM DUAL UNION ALL SELECT 'Previously raised, not complied' sup\_comm FROM DUAL.

# Dropdown for Life Insurance Policy Type

Type: SEC\_FRAR\_REQ

Title: Dropdown for Life Insurance Policy Type  
  
Acceptance Criteria:  
1. The list of life insurance policy types should be displayed in a dropdown menu.  
2. The dropdown menu should be positioned correctly on the form.  
3. The dropdown menu should be navigable using the keyboard.  
4. The selected value should be restricted to uppercase.  
5. The dropdown menu should not display horizontal or vertical scrollbars.  
6. The prompt for the dropdown menu should be labeled "Life" and should be bold and centered.  
  
Definition of Done:  
1. The dropdown menu for selecting the life insurance policy type is implemented and functional.  
2. The dropdown menu is correctly positioned and styled as per the requirements.  
3. The dropdown menu is keyboard navigable and restricts input to uppercase.  
4. The prompt "Life" is displayed correctly and is bold and centered.  
5. The form is tested and verified to ensure all acceptance criteria are met.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the dropdown menu is not directly linked to a database item.

# Underwriting Reason Selection and Supervisor Comments Permissions

Type: SEC\_FRAR\_REQ

Title: Underwriting Reason Selection and Supervisor Comments Permissions  
  
Acceptance Criteria:  
1. When a user selects a reason for underwriting (U/W) from the list, the system should display the corresponding description:  
 - If the reason is '1', the description should be 'Document Required'.  
 - If the reason is '2', the description should be 'Internal Requirement'.  
 - If the reason is '3', the description should be 'Image not clear'.  
 - If the reason is '4', the description should be 'Suspect document'.  
 - If the reason is '5', the description should be 'Image not clear'.  
2. The system should adjust the permissions for the supervisor comments field based on the selected reason:  
 - If the reason is '2' or '3', the supervisor comments field should be read-only.  
 - If the reason is '1', the supervisor comments field should be editable.  
  
Definition of Done:  
- The user can select a reason for underwriting from a predefined list.  
- The system displays the correct description based on the selected reason.  
- The permissions for the supervisor comments field are adjusted correctly based on the selected reason.  
- All changes are saved and reflected in the system without any errors.

# Supervisor Authentication for Request Approval

Type: SEC\_FRAR\_REQ

Title: Supervisor Authentication for Request Approval  
  
Acceptance Criteria:  
1. If the "U/W Reason" field is not selected, display a warning message "U/W Reason not selected" and focus on the "U/W Reason" field.  
2. If the "U/W Reason" is set to 1 and the "Supervisor Comments" field is empty, display a warning message "Please Select Supervisor comment" and focus on the "Supervisor Comments" field.  
3. If the "Supervisor ID" field is empty, display a warning message "Please enter Supervisor ID" and focus on the "Supervisor ID" field.  
4. If the "Password" field is empty, display a warning message "Please enter Password" and focus on the "Password" field.  
5. If all fields are filled, call the approval procedure with the provided Supervisor ID and Password.  
6. If the approval is successful, display a message "Approved" and update all records in the current block with the Supervisor ID, Password, and set the "Auth Flag" to 'Y'.  
7. If the approval fails, display an error message with the provided error details.  
  
Definition of Done:  
- The authentication process should validate all required fields and display appropriate messages.  
- The approval procedure should be called with the correct parameters.  
- Upon successful approval, all records in the current block should be updated accordingly.  
- Error handling should be implemented to capture and log any unexpected errors.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include any direct database queries.

# Display appropriate LOV based on test number and navigate to Supervisor Comments

Type: SEC\_FRAR\_REQ

Title: Display appropriate LOV based on test number and navigate to Supervisor Comments  
  
Acceptance Criteria:  
1. When the user double-clicks on the "UW Reason Display" field:  
 - If the test number is 'M393', the list of values (LOV) displayed should include:  
 - 'Suspect document'  
 - 'Image not clear'  
 - If the test number is anything other than 'M393', the LOV displayed should include:  
 - 'Document Required'  
 - 'Internal Requirement'  
 - 'Image not clear'  
2. When the user presses the key to move to the next item, the focus should shift to the "Supervisor Comments" field.  
  
Definition of Done:  
- The LOV should display the correct set of reasons based on the test number.  
- The focus should correctly move to the "Supervisor Comments" field when the user navigates to the next item.  
- The functionality should be tested and verified to ensure it meets the acceptance criteria.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- SELECT 'Suspect document' req, 4 req\_int FROM DUAL UNION ALL SELECT 'Image not clear' req, 5 req\_int FROM DUAL;  
- SELECT 'Document Required' req, 1 req\_int FROM DUAL UNION ALL SELECT 'Internal Requirement' req, 2 req\_int FROM DUAL UNION ALL SELECT 'Image not clear' req, 3 req\_int FROM DUAL;

# Display and Interact with UW\_SHEET Section

Type: UW\_SHEET

Title: Display and Interact with UW\_SHEET Section  
  
Acceptance Criteria:  
1. The "Rect. No." field should be displayed but not editable.  
2. The "Agent Code" field should be displayed but not editable.  
3. The "Policy Holder" field should be displayed but not editable.  
4. The "BMI" field for the insured person should be displayed but not editable.  
5. The "BMI" field for the policy holder should be displayed but not editable.  
6. All fields should have appropriate labels and should be aligned as specified.  
7. The "UW\_SHEET" section should be displayed within a tabbed canvas named "NBTABS".  
8. The overall window should be titled "New Business" and should have a vertical scrollbar.  
  
Definition of Done:  
- The "UW\_SHEET" section is implemented with all specified fields and labels.  
- All fields are displayed as read-only where specified.  
- The section is displayed within the "NBTABS" tabbed canvas.  
- The window is titled "New Business" and includes a vertical scrollbar.  
- The implementation is tested and verified to meet all acceptance criteria.

# View and Manage Risk Declaration Details

Type: RISK\_DECLARE

Title: View and Manage Risk Declaration Details  
  
Acceptance Criteria:  
1. The system should display a list of risk declaration records with the following fields:  
 - Policy Number  
 - Premium  
 - Life Coverage Sum  
 - Year of Issue  
 - Insurance Type  
2. The fields should be displayed in a tabular format with appropriate labels and alignment.  
3. The system should allow navigation through multiple records.  
4. The system should not allow insertion or updating of records directly from this interface.  
5. The system should provide a scrollbar for navigating through the records if the number of records exceeds the display limit.  
  
Definition of Done:  
1. The user interface displays the risk declaration details as specified.  
2. The user can navigate through the records using the provided scrollbar.  
3. The system prevents any insertion or update operations on the displayed records.  
4. The user interface is tested and verified for correct display and navigation functionality.  
  
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.

# Dropdown List for Insurance Policy Type Selection

Type: RISK\_DECLARE

Detailed description: As a user, I want to select the type of insurance policy from a predefined list so that I can accurately declare the risk associated with the policy.  
  
Acceptance criteria:  
1. The user should be able to see a dropdown list with two options for the type of insurance policy.  
2. The dropdown list should be labeled "Life" and should be positioned appropriately on the form.  
3. The selected value should be stored as a number in the database.  
4. The dropdown list should be displayed on the "FIN\_UW" tab of the form.  
5. The dropdown list should have a maximum length of 2 characters.  
6. The dropdown list should be styled with a white background, black text, and use the "MS Sans Serif" font.  
  
Definition of Done:  
- The dropdown list for selecting the type of insurance policy is implemented and visible on the "FIN\_UW" tab.  
- The dropdown list contains two options and is labeled "Life".  
- The selected value is stored as a number in the database.  
- The dropdown list adheres to the specified styling and positioning requirements.  
- The feature has been tested and verified to meet the acceptance criteria.  
  
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.

# Manage Family History Records

Type: FAMILY\_HIST

Title: Manage Family History Records  
  
Acceptance Criteria:  
1. The system should allow users to add new family members with details such as family member name, age, health status, age at death, and cause of death.  
2. The system should allow users to update existing family member details.  
3. The system should allow users to delete a family member record.  
4. When navigating through records, if a specific condition is met (e.g., a policy holder's equal insurance status is 1), the system should automatically set a specific value for the family member's insurance type.  
5. The system should display appropriate warning or error messages based on specific conditions.  
  
Definition of Done:  
1. Users can successfully add, update, and delete family member records.  
2. The system correctly handles navigation and sets values based on predefined conditions.  
3. Appropriate messages are displayed to the user for warnings and errors.  
4. All functionalities are tested and verified to be working as expected.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Not applicable as the provided XML content does not include specific SQL queries for CRUD operations.

# Validate Age Input in Family History Section

Type: FAMILY\_HIST

Title: Validate Age Input in Family History Section  
  
Acceptance Criteria:  
- If the age entered in the family history section is less than zero, an error message should be displayed stating "Age Entered cannot be less than Zero."  
  
Definition of Done:  
- The system should validate the age input in the family history section.  
- An error message should be displayed if the age entered is less than zero.  
- The error message should be clear and instructive to the user.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include any specific database queries.

# Validate Age Input in FAMILY\_HIST Section

Type: FAMILY\_HIST

Title: Validate Age Input in FAMILY\_HIST Section  
  
Acceptance Criteria:  
- The age field should accept only numeric values.  
- The age entered must be greater than or equal to zero.  
- If a negative age is entered, an error message should be displayed stating, "Age Entered cannot be less than Zero."  
  
Definition of Done:  
- The age field is implemented and visible in the FAMILY\_HIST section.  
- The field accepts only numeric input.  
- Validation logic is in place to ensure the age is non-negative.  
- An appropriate error message is displayed if the validation fails.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Not applicable as the provided XML content does not include any specific database queries.

# Validate Age Died Field

Type: FAMILY\_HIST

Title: Validate Age Died Field  
  
Acceptance Criteria:  
- When the user enters a value in the "Age Died" field, the system should validate the input.  
- If the entered age is less than zero, the system should display an error message stating, "Age Entered cannot be less than Zero."  
  
Definition of Done:  
- The validation logic is implemented and tested.  
- The error message is displayed correctly when the validation fails.  
- The system prevents the user from saving invalid data.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include any specific database queries.

# Cause of Death Dropdown List

Type: FAMILY\_HIST

Title: Cause of Death Dropdown List  
  
Acceptance Criteria:  
1. The cause of death field should be a dropdown list containing 12 predefined options.  
2. The dropdown list should be displayed on the "Questionnaire" tab.  
3. The field should be labeled "Cause of Death" and the label should be centered above the dropdown list.  
4. The dropdown list should be keyboard navigable to allow users to select an option using the keyboard.  
5. The field should be enabled and visible to the user.  
6. The dropdown list should display up to 3 items at a time.  
  
Definition of Done:  
1. The cause of death dropdown list is implemented and displays 12 predefined options.  
2. The dropdown list is placed on the "Questionnaire" tab and is labeled "Cause of Death" with the label centered.  
3. The dropdown list is keyboard navigable.  
4. The field is enabled and visible to the user.  
5. The dropdown list displays up to 3 items at a time.  
6. The feature is tested and verified to meet the acceptance criteria.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include any specific database queries.

# Dropdown Menu for Family Member Selection

Type: FAMILY\_HIST

Title: Dropdown Menu for Family Member Selection  
  
Acceptance Criteria:  
1. The list of family members should be displayed in a dropdown menu.  
2. The dropdown menu should be positioned appropriately within the "Family History" section.  
3. The dropdown menu should be enabled and allow for selection.  
4. The selected family member should be saved and updated in the system.  
5. The dropdown menu should be navigable using the keyboard.  
6. The dropdown menu should display up to 3 items at a time.  
7. The dropdown menu should not have horizontal or vertical scrollbars.  
8. The prompt for the dropdown menu should be labeled "Family Member" and should be aligned and styled appropriately.  
  
Definition of Done:  
1. The dropdown menu for selecting a family member is implemented and integrated into the "Family History" section.  
2. The dropdown menu is functional and meets all acceptance criteria.  
3. The user interface is tested for proper alignment, styling, and functionality.  
4. The selected family member data is correctly saved and updated in the system.  
5. 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 specific database queries or operations.

# Delete Member from Family History

Type: FAMILY\_HIST

Title: Delete Member from Family History  
  
Acceptance Criteria:  
1. When the delete button is pressed, the system should display an alert message indicating that the member is being deleted.  
2. If the control flag `CN\_WIP\_CONTINUE` is set to 'T', the system should attempt to find the member in the `WIP\_AZBJ\_POLICY\_FAMILY\_REP` table using the current record number.  
3. If the member is not found in the extension table, an error message should be displayed.  
4. If the member is found, the system should delete the member from the `WIP\_AZBJ\_POLICY\_FAMILY\_REP` table and then delete the record from the family history block.  
5. If the control flag `CN\_WIP\_CONTINUE` is not set to 'T', the system should directly delete the record from the family history block.  
  
Definition of Done:  
- The delete button functionality is implemented and tested.  
- The system displays appropriate alert messages during the delete operation.  
- The system correctly handles the deletion of records based on the control flag.  
- Error handling is in place for cases where the member is not found in the extension table.  
- The feature is reviewed and approved by stakeholders.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- `DMI.FIND('WIP\_AZBJ\_POLICY\_FAMILY\_REP.MEMBER\_ID', :family\_hist.fm\_rownum)` - This query is used to find the member in the `WIP\_AZBJ\_POLICY\_FAMILY\_REP` table using the current record number.  
- `DMI.DELETE\_ROW('WIP\_AZBJ\_POLICY\_FAMILY\_REP', azbj\_mn\_ext\_row)` - This query is used to delete the member from the `WIP\_AZBJ\_POLICY\_FAMILY\_REP` table.

# Manage Panel Doctor Details

Type: PANEL\_DOCTOR

Title: Manage Panel Doctor Details  
  
Acceptance Criteria:  
1. The system should allow users to input and validate doctor codes and test codes.  
2. The system should display the doctor's name and other relevant details based on the inputted doctor code.  
3. The system should allow users to delete a doctor from the panel.  
4. The system should provide a button to populate the doctor panel with relevant data.  
5. The system should allow users to view images related to the doctor.  
6. The system should automatically set the IP type to 1 when navigating to the next record if a specific condition is met.  
7. The system should fetch and display medical test details based on predefined criteria.  
8. The system should ensure that only valid panel doctors are displayed, excluding those who are terminated or do not meet specific criteria.  
  
Definition of Done:  
1. The user interface for managing panel doctors is implemented and accessible.  
2. Users can input and validate doctor codes and test codes.  
3. The system displays the doctor's name and other details correctly.  
4. Users can delete doctors from the panel using the provided button.  
5. The "Populate Doctor Panel" button functions as expected.  
6. The "View Images" button allows users to view related images.  
7. The system automatically sets the IP type to 1 when navigating to the next record under specific conditions.  
8. Medical test details are fetched and displayed based on the criteria.  
9. Only valid panel doctors are displayed, excluding those who are terminated or do not meet specific criteria.  
10. All functionalities are tested and verified to be working correctly.  
  
DB queries for Table reference CRUD operations only(With Usage):  
1. Fetch Medical Test Details:  
 ```sql  
 SELECT sys\_code med\_code, UPPER(TRIM(sys\_desc)) med\_test  
 FROM azbj\_system\_constants  
 WHERE sys\_type = 'MED\_TEST'  
 AND NVL(TO\_DATE(:susac.sa\_daterecd, 'DD/MM/RRRR'), pme\_api.opus\_date)  
 BETWEEN start\_date  
 AND NVL(end\_date, TO\_DATE('01-jan-3000', 'dd/mm/rrrr'))  
 AND UPPER(TRIM(sys\_code)) = DECODE(:control.mer\_rights, 'Y', 'MER', UPPER(TRIM(sys\_code)))  
 ORDER BY med\_test;  
 ```  
  
2. Fetch Valid Panel Doctors:  
 ```sql  
 SELECT panel\_doctor  
 FROM azbj\_panel\_doctor\_ref a  
 WHERE NOT EXISTS (SELECT panel\_doctor  
 FROM azbj\_panel\_doctor\_terminated b  
 WHERE b.panel\_doctor = a.panel\_doctor)  
 AND 1 = (CASE  
 WHEN panel\_doctor NOT IN ('DO27220559', 'DO27200589')  
 AND panel\_doctor LIKE 'D%'  
 THEN 0  
 ELSE 1  
 END);  
 ```

# Validate MER No. Field Input

Type: PANEL\_DOCTOR

Title: Validate MER No. Field Input  
  
Acceptance Criteria:  
1. When a user enters a value in the "MER No." field, the system should validate if the input is a number.  
2. If the input is not a number, the system should clear the field and display an error message: "Please enter a valid MER No."  
  
Definition of Done:  
1. The "MER No." field should only accept numerical input.  
2. If the input is invalid, the field should be cleared, and an error message should be displayed.  
3. The functionality should be 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 direct database CRUD operations.

# Validate 'Date Received' Field in Panel Doctor Section

Type: PANEL\_DOCTOR

Title: Validate 'Date Received' Field in Panel Doctor Section  
  
Acceptance Criteria:  
1. The 'Date Received' field should be in the format 'dd/mm/yyyy'.  
2. The field should be disabled for editing by default.  
3. When a date is entered, it should be validated to ensure it matches a predefined date (opus date).  
4. If the entered date does not match the predefined date, an error message should be displayed: 'Received Date should be equal to opus date'.  
  
Definition of Done:  
1. The 'Date Received' field is displayed in the correct format.  
2. The field is non-editable by default.  
3. Validation logic is implemented to check the entered date against the predefined date.  
4. An appropriate error message is shown if the validation fails.  
5. All acceptance criteria are met and tested successfully.

# View Images Button Functionality

Type: PANEL\_DOCTOR

Title: View Images Button Functionality  
  
Acceptance Criteria:  
1. When the "View Images" button is pressed, the system should check if the policy reference is not null.  
2. If the policy reference is valid, the system should generate a URL using the insured person's verification number or sign card number.  
3. The system should then open the generated URL in a web browser to display the images.  
4. If there is an error in generating the URL, the system should display an appropriate error message.  
  
Definition of Done:  
- The "View Images" button functionality is implemented and tested.  
- The system correctly checks for a valid policy reference.  
- The URL is generated and opened in a web browser when the policy reference is valid.  
- Appropriate error messages are displayed in case of any issues.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided logic does not include direct CRUD operations on the database.

# Validate Doctor Code Based on Test Number, Life Type, and Application Number

Type: PANEL\_DOCTOR

Detailed description: As a user, I want the system to validate the selected doctor code based on the test number, life type, and application number, so that I can ensure the correct doctor is associated with the scrutiny module.  
  
Acceptance criteria:  
1. When the life type is changed, the system should:  
 - Retrieve the doctor code from the `azbj\_scrutiny\_test\_dtl` table where the test number matches the provided test code, the life type matches the selected life type, and the application number matches either the verification number or the sign card number of the insured person.  
 - If no matching record is found, the doctor code should be set to null.  
 - If a matching record is found, the doctor code should be compared with the current doctor code.  
 - If the retrieved doctor code is not null and different from the current doctor code, a message should be displayed to the user asking for confirmation to proceed with the selected doctor code.  
  
Definition of Done:  
- The system correctly retrieves and validates the doctor code based on the provided criteria.  
- Appropriate messages are displayed to the user for confirmation when necessary.  
- The functionality is tested and verified to work as expected.  
  
DB queries for Table reference CRUD operations only(With Usage):  
```sql  
SELECT DOCTOR\_CODE  
INTO V\_doctor\_code  
FROM azbj\_scrutiny\_test\_dtl  
WHERE test\_no = :PANEL\_DOCTOR.PD\_TEST\_CODE  
AND LIFE = TO\_CHAR(:PANEL\_DOCTOR.PD\_IP\_TYPE)  
AND APPLICATION\_NO = TO\_CHAR(NVL(:INSURED\_PERSON.IP\_VERF\_NO, :INSURED\_PERSON.IP\_SIGN\_CARD\_NO));  
```

# Manage Fee Payable Status for Doctors

Type: PANEL\_DOCTOR

Title: Manage Fee Payable Status for Doctors  
  
Acceptance Criteria:  
1. The fee payable status should be displayed as a list item with options to select 'Yes' or 'No'.  
2. The default value for the fee payable status should be 'No'.  
3. The fee payable status should be displayed on the 'MED\_UW' tab page within the New Business module.  
4. The fee payable status should be clearly labeled as "Fee Payable" and should be positioned appropriately on the user interface.  
5. The fee payable status should be non-editable by default.  
  
Definition of Done:  
1. The fee payable status is displayed as a list item with 'Yes' and 'No' options.  
2. The default value is set to 'No'.  
3. The fee payable status is visible on the 'MED\_UW' tab page.  
4. The label "Fee Payable" is correctly positioned and visible.  
5. The fee payable status is non-editable by default.  
  
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.

# Navigate to Doctor Signature Section

Type: PANEL\_DOCTOR

Title: Navigate to Doctor Signature Section  
  
User Story:  
As a user, I want to navigate to the Doctor Signature section when I press the "Doctor Signature" button, so that I can access and manage doctor signatures efficiently.  
  
Acceptance Criteria:  
1. The "Doctor Signature" button should be labeled "Doctor Signature" and be visible on the interface.  
2. The button should be located at the specified position on the interface.  
3. When the "Doctor Signature" button is pressed, the system should navigate to the Doctor Signature section.  
  
Definition of Done:  
- The "Doctor Signature" button is implemented and visible on the interface.  
- Pressing the "Doctor Signature" button successfully navigates the user to the Doctor Signature section.  
- The button's label and position match the specified requirements.  
- The functionality is tested and verified to ensure it works as expected.  
  
Block Name: PANEL\_DOCTOR

# Display Medical Examiner Name LOV on Double-Click

Type: PANEL\_DOCTOR

Title: Display Medical Examiner Name LOV on Double-Click  
  
Acceptance Criteria:  
1. When the user double-clicks on the "Doctor Name" field, a list of values (LOV) should be displayed.  
2. The LOV should fetch and display the names of medical examiners from the database.  
3. The LOV should exclude terminated medical examiners and apply specific filtering criteria:  
 - Exclude medical examiners with IDs 'DO27220559' and 'DO27200589' if their IDs start with 'D'.  
4. The user should be able to select a name from the LOV, and the selected name should populate the "Doctor Name" field.  
  
Definition of Done:  
1. The "Doctor Name" field should be non-editable directly by the user.  
2. Double-clicking on the "Doctor Name" field should trigger the LOV to appear.  
3. The LOV should correctly display the filtered list of medical examiners.  
4. Selecting a name from the LOV should populate the "Doctor Name" field with the selected name.  
5. The functionality should be tested and verified to ensure it meets the acceptance criteria.  
  
DB queries for Table reference CRUD operations only(With Usage):  
```sql  
SELECT CASE  
 WHEN partner\_type = 'I'  
 THEN institution\_name  
 ELSE before\_title || ' ' || first\_name || ' ' || surname  
 END panel\_doctor\_name,  
 panel\_doctor  
 FROM azbj\_panel\_doctor\_ref a, cp\_partners b  
 WHERE a.part\_id = b.part\_id  
 AND NOT EXISTS (SELECT panel\_doctor  
 FROM azbj\_panel\_doctor\_terminated b  
 WHERE b.panel\_doctor = a.panel\_doctor)  
 AND 1 =  
 (CASE  
 WHEN panel\_doctor NOT IN ('DO27220559', 'DO27200589')  
 AND panel\_doctor LIKE 'D%'  
 THEN 0  
 ELSE 1  
 END  
 );  
```

# Validate Test Code and Doctor Authorization

Type: PANEL\_DOCTOR

Title: Validate Test Code and Doctor Authorization  
  
Acceptance Criteria:  
1. When a test code is entered and the received date is null, the system should automatically populate the received date with the current date.  
2. The system should check if the doctor is authorized to perform the entered test code by querying the `AZBJ\_DOCTOR\_TEST\_DETAIL` table.  
3. If the doctor is not authorized to perform the test, the system should display a confirmation message asking if the user wants to continue.  
4. If the user chooses not to continue, the system should trigger a failure process.  
5. If the test code is not 'MER', the system should disable the `DOC\_MER\_NO` field and clear its value.  
6. If the test code is 'MER', the system should enable the `DOC\_MER\_NO` field.  
  
Definition of Done:  
- The system correctly validates the test code and handles the received date.  
- The system accurately checks the doctor's authorization to perform the test.  
- The confirmation message is displayed when the doctor is not authorized, and the user's choice is handled appropriately.  
- The `DOC\_MER\_NO` field is enabled or disabled based on the test code.  
  
DB queries for Table reference CRUD operations only(With Usage):  
```sql  
SELECT COUNT()  
INTO v\_cnt  
FROM AZBJ\_DOCTOR\_TEST\_DETAIL  
WHERE DOCTOR\_CODE = :PANEL\_DOCTOR.pd\_doc\_code  
AND TEST\_CODE = :PANEL\_DOCTOR.PD\_TEST\_CODE;  
```  
  
```sql  
SELECT sys\_code med\_code, UPPER(TRIM(sys\_desc)) med\_test  
FROM azbj\_system\_constants  
WHERE sys\_type = 'MED\_TEST'  
AND NVL(TO\_DATE(:susac.sa\_daterecd, 'DD/MM/RRRR'), pme\_api.opus\_date)  
BETWEEN start\_date  
AND NVL(end\_date, TO\_DATE('01-jan-3000', 'dd/mm/rrrr'))  
AND UPPER(TRIM(sys\_code)) = DECODE(:control.mer\_rights, 'Y', 'MER', UPPER(TRIM(sys\_code)))  
ORDER BY med\_test;  
```

# Delete Doctor Record with Payment Check

Type: PANEL\_DOCTOR

Detailed description: As a user, I want to be able to delete a doctor's record from the panel, but only if no payment has been made to the doctor, to ensure data integrity and prevent accidental deletions of important records.  
  
Acceptance criteria:  
1. When the delete button is pressed, the system should check if the "Continue Work In Progress" flag is set to 'T'.  
2. If the flag is set to 'T', the system should verify if any payment has been made to the doctor by checking the `azbj\_med\_payout\_extract` table using the contract ID, doctor code, and test code.  
3. If a payment has been made (indicated by a 'Y' in the `PAID\_FLAG` column), the system should display an error message: "Payment made to doctor. You cannot delete the record."  
4. If no payment has been made, the system should proceed to delete the doctor's record from the panel.  
  
Definition of Done:  
- The delete functionality should be implemented and tested.  
- The system should correctly check the payment status before allowing the deletion.  
- Appropriate error messages should be displayed when necessary.  
- The doctor's record should be deleted only if no payment has been made.  
  
DB queries for Table reference CRUD operations only(With Usage):  
```sql  
-- Check if payment has been made to the doctor  
SELECT 'Y'  
INTO v\_paid\_flag  
FROM azbj\_med\_payout\_extract  
WHERE contract\_id = :control.cn\_contract\_id  
AND doctor\_code = :PANEL\_DOCTOR.PD\_DOC\_CODE  
AND TEST\_CODE = :PANEL\_DOCTOR.PD\_TEST\_CODE  
AND PAID\_FLAG = 'Y';  
```

# Populate Doctor Panel Based on Policy Reference

Type: PANEL\_DOCTOR

Detailed description: As a user, I want to populate the doctor panel with relevant doctor information based on the policy reference, so that I can view and manage the doctor details associated with a specific policy.  
  
Acceptance criteria:  
1. When the "Populate Doctor Panel" button is pressed, an alert should prompt the user to confirm if they want to clear the existing doctor panel before populating it.  
2. If the user confirms to clear the panel:  
 - The system should clear the current doctor panel.  
 - The system should fetch doctor details from the `azbj\_cardio\_details` and `azbj\_patho\_details` tables where the `policy\_ref` matches the given policy reference and the doctor code is not null.  
 - The fetched doctor details should be displayed in the doctor panel.  
3. If the user chooses not to clear the panel:  
 - The system should fetch doctor details from the `azbj\_cardio\_details` and `azbj\_patho\_details` tables where the `policy\_ref` matches the given policy reference and the doctor code is not null.  
 - The system should check if the fetched doctor details already exist in the doctor panel.  
 - If the details do not exist, they should be added to the doctor panel.  
  
Definition of Done:  
- The doctor panel is populated with relevant doctor information based on the policy reference.  
- The user is prompted with an alert to confirm clearing the panel before populating.  
- The system correctly handles both scenarios: clearing the panel before populating and adding new details without clearing.  
- The doctor details are fetched from the appropriate tables and displayed in the doctor panel.  
  
DB queries for Table reference CRUD operations only (With Usage):  
```sql  
-- Fetching doctor details from azbj\_cardio\_details and azbj\_patho\_details based on policy\_ref  
SELECT mer\_doctor\_code AS doctor\_code, mer\_doctor\_name AS doctor\_name, 'MER' AS test\_code, mer\_no, IP\_NO  
FROM azbj\_cardio\_details  
WHERE policy\_ref = :control.cn\_policy\_ref AND mer\_doctor\_code IS NOT NULL  
UNION ALL  
SELECT ecg\_doctor\_code AS doctor\_code, ecg\_doctor\_name AS doctor\_name, 'ECG' AS test\_code, NULL AS mer\_no, IP\_NO  
FROM azbj\_cardio\_details  
WHERE policy\_ref = :control.cn\_policy\_ref AND ecg\_doctor\_code IS NOT NULL  
UNION ALL  
SELECT ctmt\_doctor\_code AS doctor\_code, ctmt\_doctor\_name AS doctor\_name, 'CTMT' AS test\_code, NULL AS mer\_no, IP\_NO  
FROM azbj\_cardio\_details  
WHERE policy\_ref = :control.cn\_policy\_ref AND ctmt\_doctor\_code IS NOT NULL  
UNION ALL  
SELECT a1\_doctor\_code AS doctor\_code, a1\_doctor\_name AS doctor\_name, 'A1' AS test\_code, NULL AS mer\_no, IP\_NO  
FROM azbj\_patho\_details  
WHERE policy\_ref = :control.cn\_policy\_ref AND a1\_doctor\_code IS NOT NULL  
UNION ALL  
SELECT b1\_doctor\_code AS doctor\_code, b1\_doctor\_name AS doctor\_name, 'B1' AS test\_code, NULL AS mer\_no, IP\_NO  
FROM azbj\_patho\_details  
WHERE policy\_ref = :control.cn\_policy\_ref AND b1\_doctor\_code IS NOT NULL  
UNION ALL  
SELECT c\_doctor\_code AS doctor\_code, c\_doctor\_name AS doctor\_name, 'C' AS test\_code, NULL AS mer\_no, IP\_NO  
FROM azbj\_patho\_details  
WHERE policy\_ref = :control.cn\_policy\_ref AND c\_doctor\_code IS NOT NULL  
UNION ALL  
SELECT d\_doctor\_code AS doctor\_code, d\_doctor\_name AS doctor\_name, 'D' AS test\_code, NULL AS mer\_no, IP\_NO  
FROM azbj\_patho\_details  
WHERE policy\_ref = :control.cn\_policy\_ref AND d\_doctor\_code IS NOT NULL  
UNION ALL  
SELECT ur\_cot\_doctor\_code AS doctor\_code, ur\_cot\_doctor\_name AS doctor\_name, 'M529' AS test\_code, NULL AS mer\_no, IP\_NO  
FROM azbj\_patho\_details  
WHERE policy\_ref = :control.cn\_policy\_ref AND ur\_cot\_doctor\_code IS NOT NULL;  
```

# Confirmation Dialog Implementation

Type: END\_MOVEMENT1

Title: Confirmation Dialog Implementation  
  
Acceptance Criteria:  
1. The confirmation dialog should have a title "Confirmation".  
2. The dialog should contain two buttons:  
 - An "Ok" button to confirm the action.  
 - A "Cancel" button to cancel the action.  
3. The dialog should also include a radio group with predefined options to select an action.  
4. The dialog should be modal, meaning it should block interaction with other windows until a decision is made.  
5. The dialog should not be resizable, minimizable, or maximizable.  
6. The dialog should be centered on the screen.  
  
Definition of Done:  
- The confirmation dialog is implemented and displays correctly with the specified title and buttons.  
- The "Ok" and "Cancel" buttons function as expected.  
- The radio group is present and initialized with a default value.  
- The dialog is modal and prevents interaction with other windows until closed.  
- The dialog cannot be resized, minimized, or maximized.  
- The dialog appears centered on the screen.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as there are no database operations specified in the provided XML content.

# Cancel Button Functionality

Type: END\_MOVEMENT1

Title: Cancel Button Functionality  
  
Acceptance Criteria:  
1. When the 'Cancel' button is pressed, the system should navigate to the 'Insured Person' section.  
2. The 'Save' window should be hidden upon pressing the 'Cancel' button.  
  
Definition of Done:  
- The 'Cancel' button is functional and navigates the user to the 'Insured Person' section.  
- The 'Save' window is hidden when the 'Cancel' button is pressed.  
- The functionality is tested and verified to ensure it works as expected.

# User Confirmation Dialog for Exit Action

Type: END\_MOVEMENT1

Title: User Confirmation Dialog for Exit Action  
  
Acceptance Criteria:  
1. The user should be presented with a dialog window titled "Confirmation" when an exit action is initiated.  
2. The dialog window should contain a radio group with predefined options for the user to select the type of exit action.  
3. The default selected option in the radio group should be "CE".  
4. The dialog window should not be resizable, maximizable, or minimizable.  
5. The dialog window should be centered on the screen with specific dimensions.  
6. The dialog window should close automatically after the user makes a selection and confirms the action.  
  
Definition of Done:  
- The dialog window appears with the title "Confirmation" when an exit action is initiated.  
- The radio group with predefined options is displayed, and the default option "CE" is selected.  
- The dialog window is non-resizable, non-maximizable, and non-minimizable.  
- The dialog window is centered on the screen with the specified dimensions.  
- The dialog window closes automatically after the user confirms their selection.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include any database-specific queries or operations.

# Implement Ok Button Logic for Validations and Actions

Type: END\_MOVEMENT1

Title: Implement Ok Button Logic for Validations and Actions  
  
Acceptance Criteria:  
1. When the "Ok" button is pressed, the system should:  
 - Check if the agent code matches specific patterns and set a flag accordingly.  
 - If the action is 'SE', navigate through different blocks and validate the status of requirements, displaying messages if any status is invalid or missing.  
 - If the action is 'SE', insert comments into the database and update the proposal status.  
 - If the action is 'CE', update user profiles and delete specific comments from the database.  
 - Suspend the current work-in-progress if necessary.  
 - Display appropriate error or warning messages based on the validations performed.  
  
Definition of Done:  
- The system correctly navigates through the blocks and performs the necessary validations and actions when the "Ok" button is pressed.  
- Appropriate messages are displayed to the user based on the validations.  
- Data is correctly inserted, updated, or deleted in the database as per the logic.  
- The system handles exceptions gracefully without crashing.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Insert into `azbj\_uw\_comments` table to log comments.  
- Update `azbj\_phub\_tracker` table to set the locking flag.  
- Delete from `azbj\_uw\_comments` and `azbj\_deqc\_reason\_comments` tables based on specific conditions.  
- Insert into `azbj\_bbu\_activity\_dtls` table to log activity details.  
- Update `azbj\_rv\_approval\_cases` table to set the policy number.

# Implement Horizontal Toolbar with Functional Buttons

Type: HORIZONTAL\_TOOLBAR

Detailed description: As a user, I want to have a horizontal toolbar with various functional buttons in the application, so that I can perform common actions quickly and efficiently.  
  
Acceptance criteria:  
1. The toolbar should include the following buttons with their respective functionalities:  
 - New Session: Initiates a new session.  
 - Criteria Query Form: Opens a form to input query criteria.  
 - Rollback: Reverts the current transaction to the last committed state.  
 - Exit: Exits the current form.  
 - Select and Exit: Selects the current item and exits the form.  
 - Invoke List of Values: Opens a list of values for selection.  
 - Cancel Query: Cancels the current query operation.  
 - Execute Query: Executes the current query.  
 - Enter Query Mode: Switches the form to query mode.  
 - Save: Commits the current changes to the database.  
 - Clear Record: Clears the current record.  
 - Invoke Text Editor: Opens a text editor for the current item.  
 - Insert Record: Inserts a new record.  
 - Delete Record: Deletes the current record.  
 - Move to Next Record: Navigates to the next record.  
 - Move to Previous Record: Navigates to the previous record.  
 - Previous Page: Navigates to the previous page.  
 - Next Page: Navigates to the next page.  
 - Help: Opens the help documentation.  
 - User Diaries: Opens the user diaries.  
 - Incomplete Limit Diaries by Me: Shows incomplete limit diaries assigned to the user.  
 - Unprogressed Limit Diaries to Me: Shows unprogressed limit diaries assigned to the user.  
 - Unopened Non-Limit Diaries to Me: Shows unopened non-limit diaries assigned to the user.  
 - User Defined Buttons: Placeholder buttons for user-defined actions.  
 - Logo: Displays the company logo.  
  
Definition of Done:  
1. The toolbar is implemented and visible in the application.  
2. Each button performs its respective function as described.  
3. The toolbar is accessible and keyboard navigable.  
4. The toolbar is visually consistent with the application's design.  
5. The toolbar does not show unnecessary scrollbars.  
6. The toolbar is tested and verified for functionality 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.

# Execute Query on Button Press

Type: HORIZONTAL\_TOOLBAR

Title: Execute Query on Button Press  
  
Acceptance Criteria:  
1. When the button is pressed, the system should check for any help context and then execute the query to fetch the data.  
2. The query execution should be triggered programmatically upon the button press event.  
  
Definition of Done:  
1. The button is visible and enabled for user interaction.  
2. Pressing the button successfully triggers the query execution.  
3. The relevant data is fetched and displayed on the screen.  
4. The system checks for any help context before executing the query.  
  
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.

# Invoke List of Values on Button Press

Type: HORIZONTAL\_TOOLBAR

Title: Invoke List of Values on Button Press  
  
Acceptance Criteria:  
- When the button is pressed, the system should check for help information.  
- After checking for help, the system should display a list of values for the user to select from.  
  
Definition of Done:  
- The button should be functional and invoke the list of values.  
- The list of values should be displayed correctly and allow the user to make a selection.  
- The help check should be performed before displaying the list of values.  
- The feature should be tested and verified to ensure it works as expected.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- No specific database queries are mentioned in the provided XML content.

# Invoke Text Editor for Editable Text Item

Type: HORIZONTAL\_TOOLBAR

Title: Invoke Text Editor for Editable Text Item  
  
Acceptance Criteria:  
1. When the user presses the button associated with the text item, the system should check for help content and then execute the text editor functionality.  
2. The text item should be editable, allowing the user to insert and update text.  
3. The text item should be visible and enabled for user interaction.  
4. The text item should be navigable using the keyboard.  
  
Definition of Done:  
1. The text editor is invoked successfully when the user interacts with the text item.  
2. The text item allows for text insertion and updates.  
3. The text item is visible and enabled for user interaction.  
4. The text item supports keyboard navigation.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include any specific database queries.

# Enter Query Mode Functionality

Type: HORIZONTAL\_TOOLBAR

Title: Enter Query Mode Functionality  
  
Acceptance Criteria:  
1. When the user presses the "Enter Query" button, the system should check for any help context and then switch to query mode.  
2. The system should allow the user to input search criteria and execute the query to retrieve matching records.  
3. The "Enter Query" button should be enabled and visible to the user.  
4. The query mode should be indicated to the user with a hint or message.  
  
Definition of Done:  
- The "Enter Query" functionality is implemented and tested.  
- The system successfully transitions to query mode upon pressing the "Enter Query" button.  
- The user can input search criteria and retrieve matching records.  
- The feature is documented and reviewed.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Not applicable as the provided XML content does not include specific database queries.

# Cancel Query Button Functionality

Type: HORIZONTAL\_TOOLBAR

Title: Cancel Query Button Functionality  
  
Acceptance Criteria:  
1. When the user presses the "Cancel Query" button, the system should immediately stop the ongoing query operation.  
2. The system should display a hint or tooltip indicating that the button cancels the query.  
3. The button should be enabled and visible only when a query operation is in progress.  
4. The button should be located at the specified position on the user interface and should have an icon representing the cancel action.  
  
Definition of Done:  
- The "Cancel Query" button is implemented and functional.  
- The button stops any ongoing query operation when pressed.  
- The button displays a hint or tooltip that indicates its function.  
- The button is only enabled and visible during an active query operation.  
- The button is positioned correctly on the user interface and has the appropriate icon.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include any specific database queries.

# Delete Current Record Functionality

Type: HORIZONTAL\_TOOLBAR

Title: Delete Current Record Functionality  
  
Acceptance Criteria:  
- When the delete button is pressed, the system should check for any help or guidance related to the action.  
- The system should then proceed to delete the current record from the dataset.  
  
Definition of Done:  
- The delete button is functional and deletes the current record upon being pressed.  
- The system checks for any help or guidance before performing the delete action.  
- The user receives appropriate feedback or confirmation that the record has been deleted.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include specific database queries.

# Button for Selecting and Exiting the Screen

Type: HORIZONTAL\_TOOLBAR

Title: Button for Selecting and Exiting the Screen  
  
Acceptance Criteria:  
- When the button is pressed, the system should check for any help context using the `utl01\_pk\_help.check\_help` function.  
- After checking for help context, the system should execute the trigger associated with the 'F2' key.  
  
Definition of Done:  
- The button should be visible and enabled for user interaction.  
- The button should be positioned correctly on the screen.  
- The button should trigger the help check and execute the 'F2' key trigger when pressed.  
- The button should be integrated within the content canvas and should not show any scrollbars.  
- The window containing the button should have the correct dimensions and title.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Not applicable as the provided XML content does not include any database-specific queries or operations.

# Clear Record Button Functionality

Type: HORIZONTAL\_TOOLBAR

Title: Clear Record Button Functionality  
  
Acceptance Criteria:  
- When the "Clear Record" button is pressed, the system should check for any help context and then execute the action to clear the current record.  
- The button should be enabled and visible on the form.  
- The button should be navigable using the keyboard.  
- The button should display a hint "Clear the current record" when hovered over.  
- The button should have an icon associated with it for visual representation.  
  
Definition of Done:  
- The "Clear Record" button is functional and clears the current record when pressed.  
- The button is accessible and follows the specified UI/UX guidelines.  
- The button's functionality 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.

# Create Record Button Functionality

Type: HORIZONTAL\_TOOLBAR

Title: Create Record Button Functionality  
  
Acceptance Criteria:  
1. When the user clicks the "Create Record" button, the system should check for any help context and then proceed to initiate the creation of a new record.  
2. The button should be enabled and visible to the user.  
3. The button should display a hint "Insert a new record" when hovered over.  
4. The button should have an icon indicating its function to insert a new record.  
  
Definition of Done:  
1. The "Create Record" button is functional and allows users to add new records.  
2. The button performs a help context check before initiating the record creation process.  
3. The button is enabled, visible, and displays the appropriate hint and icon.  
4. The feature has been tested and verified to work as expected in the user interface.

# Implement DOWN Button Functionality

Type: HORIZONTAL\_TOOLBAR

Title: Implement DOWN Button Functionality  
  
Acceptance Criteria:  
1. When the "DOWN" button is pressed, the system should check for any help-related information using the `utl01\_pk\_help.check\_help` function.  
2. After checking for help, the system should execute the action to move to the next record in the dataset.  
  
Definition of Done:  
- The "DOWN" button is visible and enabled on the interface.  
- Pressing the "DOWN" button triggers the help check function.  
- After the help check, the system successfully moves to the next record in the dataset.  
- The button should have a hint that says "Move to the next record".  
- The button should display an icon labeled "down".  
  
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.

# Exit Form Functionality

Type: HORIZONTAL\_TOOLBAR

Detailed description: As a user, I want to be able to exit the form safely and ensure that all necessary validations and conditions are checked before the form is closed.  
  
Acceptance criteria:  
1. When the exit button is pressed, the system should validate if medical values are enabled.  
2. If medical values are enabled, the system should enable the "CANC\_EXIT" action.  
3. The system should check the dispatch flag for the contract ID and enable the "CANC\_EXIT" action if the dispatch flag is 'N'.  
4. If the user flag is 'Y', the system should ensure the "SAVE\_EXIT" action is enabled.  
5. The system should handle any exceptions and display an appropriate error message if an error occurs during the exit process.  
  
Definition of Done:  
- The exit button functionality is implemented and tested.  
- All specified conditions and validations are checked before the form is closed.  
- Appropriate error messages are displayed in case of any exceptions.  
- The functionality is verified to work as expected in different scenarios.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- SELECT FNC\_PRVS\_POLICY\_POST\_FLAG (:control.cn\_contract\_id, 'PAY\_DISPATCH', :policy\_holder.pp\_bill\_type\_method\_of\_payment) INTO v\_dispatch\_flag FROM BBU\_PARAM\_MST WHERE ROWNUM = 1;  
- SELECT char\_value INTO v\_char\_value FROM azbj\_system\_constants WHERE sys\_type = 'HUB' AND sys\_code = 'SAVE\_EXIT' AND sys\_desc = 'SAVE and EXIT button enable/disable';

# Navigate to Previous Record using UP Button

Type: HORIZONTAL\_TOOLBAR

Title: Navigate to Previous Record using UP Button  
  
Acceptance Criteria:  
- When the "UP" button is pressed, the system should check for any help context and then move to the previous record in the dataset.  
- The button should be enabled and visible to the user.  
- The button should display a hint "Move to the previous record" when hovered over.  
- The button should be keyboard navigable.  
  
Definition of Done:  
- The "UP" button functionality is implemented and tested.  
- The button correctly navigates to the previous record when pressed.  
- The button displays the correct hint and is accessible via keyboard.  
- All acceptance criteria are met and verified through testing.

# Invoke Microsoft Help on Help Button Press

Type: HORIZONTAL\_TOOLBAR

Title: Invoke Microsoft Help on Help Button Press  
  
Acceptance Criteria:  
1. When the help button is pressed, the system should check if the help functionality is available.  
2. If the help functionality is available, the system should invoke the Microsoft Help.  
  
Definition of Done:  
- The help button should be functional and invoke the Microsoft Help when pressed.  
- The system should check for the availability of the help functionality before invoking it.  
- The help functionality should be accessible and provide relevant documentation to the user.  
  
SQL query for reference:  
- Not applicable as there are no direct database operations mentioned in the provided XML content.

# Next Button Functionality

Type: HORIZONTAL\_TOOLBAR

Title: Next Button Functionality  
  
Acceptance Criteria:  
1. When the "Next" button is pressed, the system should check for any help context and display it if available.  
2. After checking for help context, the system should execute the action associated with the "KEY-F4" trigger, which typically involves navigating to the next page or form.  
  
Definition of Done:  
- The "Next" button is functional and triggers the appropriate actions when pressed.  
- The system checks for help context and displays it if available.  
- The system successfully executes the "KEY-F4" trigger action, navigating to the next page or form.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Not applicable as the provided XML content does not include any database-specific operations.

# Implement 'Previous' Button Functionality

Type: HORIZONTAL\_TOOLBAR

Title: Implement "Previous" Button Functionality  
  
User Story:  
As a user, I want to navigate to the previous page using a button, so that I can easily go back to the previous content without manually navigating through the application.  
  
Acceptance Criteria:  
1. When the "Previous" button is pressed, the system should check for any help context and execute the function associated with the "KEY-F3" action.  
2. The button should be visible and enabled, allowing users to interact with it.  
3. The button should be positioned at the specified coordinates (X: 396, Y: 8) and have the specified dimensions (Width: 24, Height: 24).  
  
Definition of Done:  
1. The "Previous" button is implemented and functional.  
2. The button correctly triggers the help check and the "KEY-F3" function when pressed.  
3. The button is visible, enabled, and positioned correctly on the user interface.  
4. The feature is tested and verified to ensure it meets the acceptance criteria.

# Save Changes to Database

Type: HORIZONTAL\_TOOLBAR

User Story: Save Changes to Database  
  
Detailed description:   
As a user, I want to be able to save (commit) the changes made in the application to the database, ensuring that all necessary validations and conditions are met before the data is committed.  
  
Acceptance criteria:  
1. If the form status is 'Y', display a message indicating that saving is not allowed due to changes in rating or underwriting parameters.  
2. Log the save action with relevant contract and insured person details.  
3. Validate the risk flags and ensure that the appropriate messages are displayed if the case is high risk.  
4. Ensure that all further requirements are received before allowing the save.  
5. Validate medical tests and ensure that all required tests are received before allowing the save.  
6. If the user is not authorized, check the risk flags for both the insured person and policy holder and display appropriate messages.  
7. Ensure that the clean flag is set correctly based on the received medical tests.  
8. Validate and save probable CPs (Customer Profiles) for both insured persons and policy holders.  
9. Ensure that the save action is logged and appropriate messages are displayed based on the conditions met.  
  
Definition of Done:  
- The user can successfully save changes to the database.  
- All validations and conditions are checked and met before committing the data.  
- Appropriate messages are displayed to the user based on the conditions.  
- The save action is logged with relevant details.  
- The system handles exceptions and errors gracefully, providing meaningful feedback to the user.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Select:   
 ```sql  
 SELECT bypass\_flag INTO pk\_vars.v\_itrule\_flag  
 FROM azbj\_application\_bypass\_det  
 WHERE application\_no = TO\_CHAR(NVL(:INSURED\_PERSON.IP\_VERF\_NO, :INSURED\_PERSON.IP\_SIGN\_CARD\_NO))  
 AND BYPASS\_MODULE = 'ITRULE\_BYPASS';  
 ```  
  
- Select:   
 ```sql  
 SELECT COUNT(APPLICATION\_NO) INTO v\_cnt\_itrule  
 FROM AZBJ\_ITRULE\_ANSWER\_DTLS  
 WHERE application\_no = TO\_CHAR(NVL(:INSURED\_PERSON.IP\_VERF\_NO, :INSURED\_PERSON.IP\_SIGN\_CARD\_NO));  
 ```  
  
- Select:   
 ```sql  
 SELECT high\_risk\_flag INTO v\_ip\_risk\_flag  
 FROM azbj\_partner\_extn  
 WHERE part\_id = :insured\_person.ip\_part\_id;  
 ```  
  
- Select:   
 ```sql  
 SELECT high\_risk\_flag INTO v\_ph\_risk\_flag  
 FROM azbj\_partner\_extn  
 WHERE part\_id = :policy\_holder.ph\_part\_id;  
 ```  
  
- Select:   
 ```sql  
 SELECT FNC\_PRVS\_POLICY\_POST\_FLAG(:control.cn\_contract\_id, 'PAY\_DISPATCH', :policy\_holder.pp\_bill\_type\_method\_of\_payment)  
 INTO v\_dispatch\_flag  
 FROM BBU\_PARAM\_MST  
 WHERE ROWNUM = 1;  
 ```  
  
- Select:   
 ```sql  
 SELECT NVL(MAX(event\_no), 0) + 1 INTO v\_event\_no  
 FROM azbj\_uw\_comments  
 WHERE contract\_id = :control.cn\_contract\_id;  
 ```  
  
- Update:   
 ```sql  
 UPDATE azbj\_uw\_comments SET FLAG = 'Y'  
 WHERE contract\_id = :control.cn\_contract\_id  
 AND policy\_no = :control.cn\_policy\_ref  
 AND comments = 'SAFE DISTRICT CASE';  
 ```  
  
- Insert:   
 ```sql  
 INSERT INTO azbj\_uw\_comments (event\_no, contract\_id, policy\_no, user\_id, comment\_date, comments, flag)  
 VALUES (v\_event\_no, :control.cn\_contract\_id, :control.cn\_policy\_ref, :boiler.userid, SYSDATE, 'SAFE DISTRICT CASE', 'Y');  
 ```  
  
- Insert:   
 ```sql  
 INSERT INTO azbj\_auto\_cp\_merge\_details (appln\_no, part\_id, gpa\_status, gpa\_date, qc\_user, qc\_date, merge\_type, upd\_name\_flag, partner\_type, upd\_fath\_name\_flag, upd\_hus\_name\_flag, cp\_substatus, approved\_user)  
 VALUES (TO\_CHAR(NVL(:INSURED\_PERSON.IP\_VERF\_NO, :INSURED\_PERSON.IP\_SIGN\_CARD\_NO)), :BLK\_PROBABLE\_CP.PART\_ID, 'Y', pme\_api.opus\_date, USER, SYSDATE, 'M\_C', NULL, 'IP', NULL, NULL, 'Pending for L1 Approval', NULL);  
 ```  
  
- Insert:   
 ```sql  
 INSERT INTO azbj\_auto\_cp\_merge\_details (appln\_no, part\_id, gpa\_status, gpa\_date, qc\_user, qc\_date, merge\_type, upd\_name\_flag, partner\_type, upd\_fath\_name\_flag, upd\_hus\_name\_flag, cp\_substatus, approved\_user)  
 VALUES (TO\_CHAR(NVL(:INSURED\_PERSON.IP\_VERF\_NO, :INSURED\_PERSON.IP\_SIGN\_CARD\_NO)), :BLK\_PROBABLE\_CP\_PH.PART\_ID, 'Y', pme\_api.opus\_date, USER, SYSDATE, 'M\_C', NULL, 'PH', NULL, NULL, 'Pending for L1 Approval', NULL);  
 ```  
  
- Delete:   
 ```sql  
 DELETE FROM azbj\_uw\_comments  
 WHERE contract\_id = :control.cn\_contract\_id  
 AND policy\_no = :control.cn\_policy\_ref  
 AND comments = 'SAFE DISTRICT CASE';  
 ```  
  
- Delete:   
 ```sql  
 DELETE FROM azbj\_auto\_cp\_merge\_details  
 WHERE appln\_no = TO\_CHAR(NVL(:INSURED\_PERSON.IP\_VERF\_NO, :INSURED\_PERSON.IP\_SIGN\_CARD\_NO))  
 AND partner\_type = 'IP';  
 ```  
  
- Delete:   
 ```sql  
 DELETE FROM azbj\_auto\_cp\_merge\_details  
 WHERE appln\_no = TO\_CHAR(NVL(:INSURED\_PERSON.IP\_VERF\_NO, :INSURED\_PERSON.IP\_SIGN\_CARD\_NO))  
 AND partner\_type = 'PH';  
 ```

# Manage Counter Offers

Type: C\_OFFER

Title: Manage Counter Offers  
  
Acceptance Criteria:  
1. The user should be able to enter the package name.  
2. The user should be able to enter the benefit term, which should be a numeric value.  
3. The user should be able to enter the premium term, which should be a numeric value.  
4. The user should be able to enter the sum assured, which should be a numeric value.  
5. The user should be able to enter the premium, which should be a numeric value.  
6. The user should be able to close the form using a close button.  
  
Definition of Done:  
1. The form should display fields for package name, benefit term, premium term, sum assured, and premium.  
2. The fields for benefit term, premium term, sum assured, and premium should only accept numeric values.  
3. The form should have a close button that allows the user to exit the form.  
4. The form should be visually consistent and user-friendly.  
5. The form should be tested 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 database queries or table references.

# Close Button Functionality

Type: C\_OFFER

Title: Close Button Functionality  
  
Acceptance Criteria:  
1. When the "Close" button is pressed, the system should navigate to the covers section.  
2. The offer window should be hidden after the navigation to the covers section.  
  
Definition of Done:  
- The "Close" button successfully navigates the user to the covers section.  
- The offer window is hidden after the navigation.  
- The functionality is tested and verified to ensure it works as expected.

# View Revised Offer Details

Type: AZBJ\_FC\_REVISED\_OFFER

Title: View Revised Offer Details  
  
Acceptance Criteria:  
1. The user should be able to see the following details:  
 - Policy Reference  
 - Contract ID  
 - Activity Number  
 - Activity Date  
 - Event Type  
 - User ID  
 - Partner ID  
 - Partner Name  
2. The details should be displayed in a read-only format, meaning the user cannot modify the information.  
3. The information should be presented in a clear and organized manner on the screen.  
  
Definition of Done:  
1. The user interface displays all the required details of the revised offer.  
2. The details are non-editable and presented in a read-only format.  
3. The information is organized and easy to read.  
4. The feature has been tested and verified to ensure that all details are correctly displayed and non-editable.  
  
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 CRUD operations.

# Manage Rider Information in Offer Module

Type: C\_OFFER\_RIDER

Title: Manage Rider Information in Offer Module  
  
Acceptance Criteria:  
1. The user should be able to view and input the following fields:  
 - Rider Code  
 - Rider Description  
 - Sum Assured  
2. The fields should be displayed in a structured format with appropriate labels.  
3. The user should be able to scroll through the records if there are more than four entries.  
  
Definition of Done:  
1. The user interface displays the Rider Code, Rider Description, and Sum Assured fields.  
2. The fields are aligned and labeled correctly.  
3. The scrollbar is functional and allows the user to navigate through multiple records.  
4. The changes are 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 database queries.

# Manage Proposal Actions and Statuses

Type: SAVE

Title: Manage Proposal Actions and Statuses  
  
Acceptance Criteria:  
1. When a new block instance is created, the system should:  
 - Clear any existing issue button disable messages.  
 - Save the current status of the proposal.  
 - Validate the proposal for HNI (High Net-worth Individual) values.  
 - Enable or disable specific actions based on the combo flag, RCU validation, and tele-verification status.  
 - Display appropriate messages for combo products, RCU cases, and pending branch verifications.  
  
2. The system should:  
 - Enable or disable specific actions based on the presence of validation errors.  
 - Check for incomplete inwarded applications and ensure all requirements are received before issuance.  
 - Validate individual and family shareholdings for key-man proposals.  
 - Ensure the proposal's product type selected in cashiering matches the BBU (Back-end Business Unit) type.  
 - Handle high-risk CP (Corporate Partner) partners by restricting issuance to HOD (Head of Department) IDs only.  
  
3. The system should:  
 - Validate the proposal against various conditions such as thumb impression cases, non-standard age proofs, and specific product requirements.  
 - Calculate and display the appropriate service tax, education cess, and higher education cess based on the proposal's details.  
 - Ensure the proposal's premium is sufficient to cover all charges, including COI (Cost of Insurance), admin fees, and service taxes.  
  
Definition of Done:  
- The user can save and manage actions and statuses related to a proposal.  
- The system correctly enables or disables actions based on the proposal's status and other conditions.  
- Appropriate messages are displayed for combo products, RCU cases, pending branch verifications, and validation errors.  
- The proposal's premium is validated and calculated correctly, including all applicable charges and taxes.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- The system should execute the following queries to validate and manage the proposal:  
 - Select the tele-verification status from `balic\_history\_details`.  
 - Validate RCU using `azbj\_validate\_rcu`.  
 - Check for incomplete inwarded applications in `azbj\_phub\_add\_req\_tracker` and `azbj\_phub\_req\_tracker`.  
 - Validate individual and family shareholdings from `azbj\_system\_constants`.  
 - Calculate service tax, education cess, and higher education cess using `azbj\_pk\_gst.get\_gst\_amount` and `az\_pk7\_general.get\_service\_tax\_edu\_cess`.

# User Action Selection via Radio Buttons

Type: SAVE

Title: User Action Selection via Radio Buttons  
  
Acceptance Criteria:  
- The user should be able to see a set of radio buttons labeled with different action options.  
- The default selected action should be "RW".  
- The user should be able to change the selected action to any other available option.  
  
Definition of Done:  
- The radio buttons for action selection are displayed correctly.  
- The default action is set to "RW".  
- The user can successfully change the action by selecting a different radio button.  
- The selected action is stored and can be retrieved for further processing.

# Implement Status Selection Radio Button Group

Type: SAVE

Title: Implement Status Selection Radio Button Group  
  
Acceptance Criteria:  
1. The status selection should be presented as a radio button group.  
2. The default value for the status should be set to "IM".  
3. The radio button group should be positioned appropriately on the form.  
4. The background color of the radio button group should be gray, and the foreground color should be black.  
5. The radio button group should be 60 units wide and 18 units high.  
  
Definition of Done:  
1. The radio button group for status selection is implemented and visible on the form.  
2. The default value "IM" is pre-selected when the form is loaded.  
3. The radio button group is correctly positioned and styled as specified.  
4. The functionality is tested and verified to ensure it meets the acceptance criteria.

# Dynamic Population of Senior Underwriter List

Type: SAVE

Title: Dynamic Population of Senior Underwriter List  
  
Acceptance Criteria:  
1. When the user clicks on the list item for selecting a Senior Underwriter, the system should clear any existing entries in the list.  
2. The system should then query the database to retrieve descriptions and values for all Senior Underwriters.  
3. The retrieved descriptions and values should be added to the list for user selection.  
4. If an error occurs during the database query, the system should handle it gracefully without crashing.  
  
Definition of Done:  
- The list of Senior Underwriters is dynamically populated upon user interaction.  
- The list is cleared before new entries are added.  
- The system retrieves and displays the correct descriptions and values for Senior Underwriters.  
- Error handling is implemented to manage any issues during the database query.  
  
DB queries for Table reference CRUD operations only (With Usage):  
```sql  
SELECT sys\_desc, char\_value  
FROM azbj\_system\_constants  
WHERE sys\_type = 'SU' AND sys\_code = 'SENIOR\_UW';  
```  
- This query is used to fetch the descriptions and values for Senior Underwriters from the `azbj\_system\_constants` table.

# Cancel Operation and Navigate to Insured Person

Type: SAVE

Detailed description: As a user, I want to cancel the current operation and navigate to the "Insured Person" section, so that I can review or edit the insured person's details without saving the current changes.  
  
Acceptance criteria:  
1. When the "Cancel" button is pressed, the system should set the action status to 'W'.  
2. The system should update the policy status in the control section with the current policy status from the SOFA\_INFO\_DATA.  
3. The system should navigate to the "Insured Person" tab.  
4. The system should switch to the "Insured Person" block.  
5. The system should hide the "Save" window.  
  
Definition of Done:  
- The "Cancel" button successfully cancels the current operation.  
- The user is navigated to the "Insured Person" section.  
- The "Save" window is hidden.  
- All acceptance criteria are met without any errors.  
  
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.

# Automatically Update Record Status Based on Selected Action

Type: SAVE

Title: As a user, I want the system to automatically update the status of a record based on the selected action, so that the status reflects the current state of the record.  
  
Acceptance Criteria:  
1. When the action is set to "ISSUED", the status should be updated to "IM".  
2. When the action is set to either "REJECTED" or "DECLINED", the status should be updated to "RM".  
3. If the action is set to any other value, the status should be set to NULL.  
  
Definition of Done:  
- The system correctly updates the status based on the selected action.  
- The changes are saved and reflected in the database.  
- The functionality is tested and verified to ensure it works as expected.

# Manage Control Items for Various Functionalities

Type: CONTROL

Title: Manage Control Items for Various Functionalities  
  
Acceptance Criteria:  
1. The control items should include buttons for actions like "Save," "Exit," and "Populate Auto Reinsurance."  
2. Display items should show information such as "Customer ID," "Employer," and "Policy Number."  
3. Checkboxes should allow users to select options like "Refer To HO," "Backdating Required," and "MWP ACT Required."  
4. Text items should capture user inputs like "Customer's preferred language of communication," "Doctor Code," and "Supervisor ID."  
5. List items should provide dropdown options for fields like "Home Loan Type," "Income Segment," and "Relationship to Child."  
6. The control items should be organized within different sections or tabs for better user experience.  
  
Definition of Done:  
1. All control items are implemented and functional as per the requirements.  
2. The user interface is intuitive and easy to navigate.  
3. The control items are tested for accuracy and reliability.  
4. Documentation is updated to reflect the new functionalities.  
5. The changes are reviewed and approved by the stakeholders.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- The following queries are used to populate LOVs (List of Values) for dropdowns:  
 ```sql  
 SELECT SCREEN\_VALUE, INTERNAL\_VALUE  
 FROM Azbj\_Ho\_allocation\_list  
 WHERE Active\_flag='Y';  
 ```  
  
 ```sql  
 SELECT SYS\_DESC  
 FROM azbj\_system\_constants  
 WHERE sys\_type = 'REJEC\_TYPE';  
 ```  
  
 ```sql  
 SELECT DOC\_TYPE, DOC\_DESC  
 FROM azbj\_dms\_doc\_type\_master  
 ORDER BY SR\_NO;  
 ```  
  
 ```sql  
 SELECT LANGUAGE\_NAME, LANGUAGE\_ID  
 FROM azbj\_comm\_language  
 ORDER BY language\_id;  
 ```  
  
 ```sql  
 SELECT ic\_code  
 FROM azbj\_signature  
 WHERE sub\_ic\_code = '1234';  
 ```

# View and Select Reinsurance Type

Type: CONTROL

Title: View and Select Reinsurance Type  
  
Acceptance Criteria:  
1. The reinsurance type field should display a list of three predefined options.  
2. The field should be read-only, meaning users cannot insert or update the values.  
3. The field should have a maximum length of 100 characters.  
4. The field should be positioned at coordinates (165, 15) on the user interface and have a width of 318 and height of 28.  
  
Definition of Done:  
1. The reinsurance type field is visible on the user interface.  
2. The field displays three predefined options for selection.  
3. Users are unable to modify the list of options.  
4. The field adheres to the specified dimensions and positioning on the user interface.

# Validate Proposal Sign Date and Navigate to Beneficiary Name

Type: CONTROL

Title: Validate Proposal Sign Date and Navigate to Beneficiary Name  
  
Acceptance Criteria:  
1. If the proposal sign date is not entered, the system should display an error message: "Please enter the proposal sign date".  
2. When navigating to the next item, the system should automatically move to the beneficiary's name field.  
  
Definition of Done:  
1. The system prompts the user to enter the proposal sign date if it is missing.  
2. The system successfully navigates to the beneficiary's name field when moving to the next item.  
3. All validations and navigations 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 any direct database CRUD operations.

# Implement Reject Reason Validation and Navigation

Type: CONTROL

Title: Implement Reject Reason Validation and Navigation  
  
Acceptance Criteria:  
1. When the "OK" button is pressed, the system should check if the reject reason is provided.  
2. If the reject reason is not provided, the system should display an error message: "Please Enter Reject Reason."  
3. If the reject reason is provided, the system should update the form status to 'N' and navigate to the "Save" section.  
  
Definition of Done:  
- The "OK" button functionality is implemented and tested.  
- The system correctly validates the presence of a reject reason.  
- The error message is displayed if the reject reason is missing.  
- The form status is updated to 'N' when a reject reason is provided.  
- The system navigates to the "Save" section upon successful validation.  
  
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.

# Implement Home Loan Type Dropdown

Type: CONTROL

Title: Implement Home Loan Type Dropdown  
  
Acceptance Criteria:  
1. The home loan type field should be a dropdown list with predefined options.  
2. The dropdown list should allow a maximum of 50 characters for each option.  
3. The field should be positioned at the specified coordinates on the form.  
4. The field should be editable, allowing users to select an option from the list.  
5. The field should be displayed on the "COVERS" tab of the form.  
6. The field should have a prompt labeled "Home Loan Type" with specific font and style attributes.  
  
Definition of Done:  
1. The home loan type dropdown list is implemented and visible on the "COVERS" tab.  
2. Users can select an option from the dropdown list.  
3. The field adheres to the specified design and style requirements.  
4. The field is editable and allows for user interaction.  
5. The implementation is tested and verified to meet the acceptance criteria.

# User selects preferred language of communication

Type: CONTROL

Detailed description: As a user, I want to be able to select the preferred language of communication for a customer from a predefined list, so that the system can store and use this information for future interactions.  
  
Acceptance criteria:  
1. The user should be able to select a language from a list of available languages.  
2. If the selected language is not null and the policy holder is the same as the insured person, the system should automatically set the policy holder's communication language to the selected language.  
  
Definition of Done:  
1. The user can see a list of available languages to choose from.  
2. The system correctly updates the policy holder's communication language based on the selected language and the condition that the policy holder is the same as the insured person.  
3. The changes are saved and can be retrieved for future interactions.  
  
DB queries for Table reference CRUD operations only (With Usage):  
```sql  
SELECT LANGUAGE\_NAME, LANGUAGE\_ID  
FROM azbj\_comm\_language  
ORDER BY language\_id;  
```  
This query is used to fetch the list of available languages for the user to select from.

# Display Home Visit Charge Options in Grid

Type: CONTROL

Title: Display Home Visit Charge Options in Grid  
  
Acceptance Criteria:  
1. The list of home visit charge options should be displayed in a grid format.  
2. The grid should be positioned at the specified location on the screen.  
3. The grid should have a white background and black text.  
4. The font used in the grid should be "MS Sans Serif" with a size of 8 and bold styling.  
5. The grid should display a maximum of 5 items at a time.  
6. The grid should be 135 units wide and 18 units high.  
  
Definition of Done:  
1. The list of home visit charge options is displayed correctly in the specified grid format.  
2. The grid is positioned correctly on the screen.  
3. The grid's appearance matches the specified design (white background, black text, specified font and size).  
4. The grid displays up to 5 items at a time.  
5. The functionality has been tested and verified to meet the acceptance criteria.  
  
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.

# Input and View Rejection Comments

Type: CONTROL

Detailed description: As a user, I want to be able to input and view rejection comments in a designated text area so that I can provide detailed reasons for rejection. The text area should support multiple lines and automatically convert all input to uppercase. Additionally, I should be able to double-click on the text area to open a list of predefined rejection reasons for easy selection.  
  
Acceptance criteria:  
1. The text area should allow up to 500 characters.  
2. The text area should support multiple lines of text.  
3. All input should be automatically converted to uppercase.  
4. Double-clicking on the text area should open a list of predefined rejection reasons.  
5. The list of predefined rejection reasons should be fetched from the database using the query:   
 ```sql  
 SELECT SYS\_DESC   
 FROM azbj\_system\_constants   
 WHERE sys\_type = 'REJEC\_TYPE';  
 ```  
  
Definition of Done:  
1. The text area for rejection comments is implemented and visible on the form.  
2. The text area supports multiple lines and converts all input to uppercase.  
3. Double-clicking on the text area successfully opens a list of predefined rejection reasons.  
4. The list of predefined rejection reasons is correctly fetched from the database and displayed.  
5. All acceptance criteria are met and tested.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- The query to fetch predefined rejection reasons:  
 ```sql  
 SELECT SYS\_DESC   
 FROM azbj\_system\_constants   
 WHERE sys\_type = 'REJEC\_TYPE';  
 ```

# HO Allocation List Selection and Validation

Type: CONTROL

Detailed description: As a user, I want to be able to select a value from the HO Allocation List and have the system validate and process this selection based on specific business rules.  
  
Acceptance criteria:  
1. When a user double-clicks on the HO Allocation List field:  
 - If the field is not empty and the internal value is 'POL\_EXP', the system should check if the user is authorized to select the value.  
 - If the user is not authorized, an error message should be displayed.  
 - The system should display the HO Allocation List for selection.  
 - The selected value should be stored globally for further use.  
 - If the internal value is 'POL\_EXP', the system should set the politically exposed flag to 'Y'; otherwise, it should set it to 'N'.  
  
2. When the HO Allocation List field is validated:  
 - If the field is empty, the internal value should be set to NULL.  
 - The system should set the politically exposed flag based on the internal value ('Y' for 'POL\_EXP', 'N' otherwise).  
  
3. When the user triggers the list of values (LOV):  
 - The system should display the HO Allocation List for selection.  
  
Definition of Done:  
- The system correctly validates and processes the HO Allocation List selection based on the specified business rules.  
- The error message is displayed when the user is not authorized to select a value.  
- The selected value is stored globally and the politically exposed flag is set correctly.  
- The HO Allocation List is displayed when triggered by the user.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- The system should use the following query to fetch the HO Allocation List:  
 ```sql  
 SELECT SCREEN\_VALUE, INTERNAL\_VALUE  
 FROM Azbj\_Ho\_allocation\_list  
 WHERE Active\_flag = 'Y';  
 ```

# Implement 'Exit' Button to Navigate to 'SAVE' Section

Type: CONTROL

Title: Implement 'Exit' Button to Navigate to 'SAVE' Section  
  
Acceptance Criteria:  
1. When the 'Exit' button is pressed, the system should navigate to the 'SAVE' section.  
2. The 'Exit' button should be clearly labeled as "Exit" and easily identifiable.  
3. The button should be located at the specified position on the form.  
4. The button should have a specific appearance (e.g., gray background, Arial font, bold text) to ensure it is distinguishable.  
5. The button should be placed on the 'REJECT\_REASON' canvas.  
6. The 'REJECT\_REASON' canvas should be displayed within the 'BBU\_RESULT' window.  
  
Definition of Done:  
- The 'Exit' button is implemented and functional.  
- The button navigates to the 'SAVE' section upon being pressed.  
- The button's appearance matches the specified design requirements.  
- The 'REJECT\_REASON' canvas and 'BBU\_RESULT' window are correctly configured.  
- The functionality is tested and verified to work as expected.

# Automatic Reinsurance Type Determination

Type: CONTROL

Title: Automatic Reinsurance Type Determination  
  
Acceptance Criteria:  
1. When the user initiates the process, the system should check the insured person's coverage amount.  
2. If the coverage amount is less than or equal to 25,000,000, the system should set the reinsurance type to "AUTO".  
3. If the coverage amount is greater than 25,000,000, the system should set the reinsurance type to "FACULTATIVE" and enable validation for the reinsurance code.  
4. The system should navigate to the reinsurance section and clear any existing records if the reinsurance type is "AUTO".  
5. If the solution name is less than 3, the system should execute the reinsurance process.  
6. If the solution name is greater than 0 and less than 3, the system should execute an additional reinsurance solution process.  
7. The system should navigate to the reinsurance reference number field and display the first record.  
  
Definition of Done:  
- The system correctly sets the reinsurance type based on the insured person's coverage amount.  
- The reinsurance code validation is enabled when the reinsurance type is "FACULTATIVE".  
- The reinsurance section is cleared when the reinsurance type is "AUTO".  
- The appropriate reinsurance processes are executed based on the solution name.  
- The system navigates to the reinsurance reference number field and displays the first record.

# Implement Radio Button Group for Furniture Request Status

Type: CONTROL

Title: Implement Radio Button Group for Furniture Request Status  
  
Acceptance Criteria:  
1. The radio button group should have an initial value set to "A".  
2. The radio button group should be displayed with a width of 90 units and a height of 18 units.  
3. The radio button group should be positioned at coordinates (77, 269) on the screen.  
4. The background color of the radio button group should be white, and the foreground color should be black.  
  
Definition of Done:  
1. The radio button group for selecting the furniture request status is implemented and visible on the user interface.  
2. The initial value of the radio button group is set to "A".  
3. The radio button group is correctly positioned and sized according to the specified dimensions.  
4. The radio button group has the specified background and foreground colors.  
5. The functionality is tested and verified to ensure it meets the acceptance criteria.

# Manage Backdating Requirement

Type: CONTROL

Title: Manage Backdating Requirement  
  
Acceptance Criteria:  
1. When the backdating flag is set to 'Y', the inception date field should be enabled.  
2. When the backdating flag is set to 'N', the inception date field should be disabled.  
  
Definition of Done:  
- The backdating flag can be toggled between 'Y' and 'N'.  
- The inception date field's enabled/disabled state changes accordingly based on the backdating flag.  
- The form status is updated to 'Y' when the backdating flag is changed.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided logic does not include direct database CRUD operations.

# Exit Button Functionality

Type: CONTROL

Title: Exit Button Functionality  
  
Acceptance Criteria:  
- When the "Exit" button is pressed, the current window should be hidden.  
- The focus should then move to the item labeled 'further\_req.fr\_res\_recd' in another section.  
  
Definition of Done:  
- The "Exit" button hides the current window upon being pressed.  
- The focus is successfully moved to the specified item in another section.  
- The functionality is tested and confirmed to work as expected.

# Input and Update Scrutiny Comments

Type: CONTROL

Title: Input and Update Scrutiny Comments  
  
Acceptance Criteria:  
1. The text field should allow a maximum of 500 characters.  
2. The text field should be visible and positioned appropriately on the screen.  
3. The text field should have a white background and black text for readability.  
4. The text field should display a vertical scrollbar if the content exceeds the visible area.  
5. The text field should be labeled "Comments" and the label should be bold and positioned correctly.  
6. The text field should allow both insertion and updating of text.  
7. Upon pressing the designated key to move to the next item, the focus should shift to the "Save" button.  
  
Definition of Done:  
1. The text field is implemented and meets all the acceptance criteria.  
2. The text field is tested for maximum length, visibility, and readability.  
3. The text field allows insertion and updating of text.  
4. The focus shifts to the "Save" button upon pressing the designated key.  
5. 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 specific database queries.

# Doctor Code Selection and Navigation

Type: CONTROL

Title: Doctor Code Selection and Navigation  
  
Acceptance Criteria:  
1. When the user double-clicks on the "Doctor Code" field, a list of valid doctor codes should be displayed.  
2. When the user presses a specific key (e.g., F9) while the "Doctor Code" field is focused, the same list of valid doctor codes should be displayed.  
3. The list of valid doctor codes should be fetched from the database using the query: `SELECT ic\_code FROM azbj\_signature WHERE sub\_ic\_code = '1234'`.  
4. After selecting a doctor code from the list, the focus should move to the next field, which is the "Doctor Signature" field.  
  
Definition of Done:  
- The "Doctor Code" field should display a list of valid doctor codes upon double-click or key press.  
- The list should be populated based on the specified database query.  
- The focus should automatically move to the "Doctor Signature" field after a doctor code is selected.  
- The functionality should be tested and verified to ensure it works as expected.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- `SELECT ic\_code FROM azbj\_signature WHERE sub\_ic\_code = '1234'` (Used to fetch the list of valid doctor codes).

# Input and Update Reason for Scrutiny Failure

Type: CONTROL

Title: Input and Update Reason for Scrutiny Failure  
  
Acceptance Criteria:  
1. The input field for the reason should allow a maximum of 200 characters.  
2. The input field should be displayed with a white background and black text.  
3. The input field should be positioned appropriately within the designated section.  
4. The input field should be a dropdown list with 5 predefined options.  
5. The input field should allow both insertion of new reasons and updating of existing reasons.  
6. The section should be displayed on a gray background canvas.  
7. The canvas should be part of a modal window titled "Scrutiny Failure Process".  
  
Definition of Done:  
1. The input field for the reason is implemented and meets the specified criteria.  
2. The input field is tested to ensure it allows a maximum of 200 characters.  
3. The input field is tested to ensure it displays with the correct background and text colors.  
4. The input field is tested to ensure it is positioned correctly within the section.  
5. The dropdown list is tested to ensure it contains 5 predefined options.  
6. The input field is tested to ensure it allows both insertion and updating of reasons.  
7. The section is tested to ensure it displays on a gray background canvas.  
8. The modal window is tested to ensure it displays with the title "Scrutiny Failure Process".  
9. All tests pass successfully and the feature is reviewed and approved by stakeholders.

# Generate and Open BI Report Link

Type: CONTROL

Title: Generate and Open BI Report Link  
  
Description: As a user, I want to generate a Business Intelligence (BI) report link and open it in a web browser when I press the "BI\_GENERATION" button, so that I can view the report associated with a specific agent.  
  
Acceptance Criteria:  
1. When the "BI\_GENERATION" button is pressed, the system should retrieve the BI link from the `azbj\_system\_constants` table where `sys\_TYPE` is 'BI\_LINK' and `sys\_CODE` is 'BI\_GENERATE'.  
2. If the BI link is successfully retrieved and the agent's BI number is not null, the system should concatenate the BI link with the agent's BI number to form the final URL.  
3. The system should then open the final URL in a web browser.  
4. If the final URL is null or any error occurs during the process, the system should display a message indicating that there is an issue with the URL.  
5. The system should increment the BI generation count and update the BI URL in the relevant variables.  
  
Definition of Done:  
- The "BI\_GENERATION" button functionality is implemented and tested.  
- The system retrieves the BI link from the database as specified.  
- The final URL is correctly formed and opened in a web browser.  
- Error handling is in place to display appropriate messages if issues occur.  
- The BI generation count and URL variables are updated accordingly.  
  
DB queries for Table reference CRUD operations only (With Usage):  
```sql  
-- Retrieve BI link  
SELECT sys\_desc  
INTO v\_bi\_link  
FROM azbj\_system\_constants  
WHERE sys\_TYPE = 'BI\_LINK'  
AND sys\_CODE = 'BI\_GENERATE';  
```  
  
Block Name: CONTROL

# Handle 'MWP ACT Required' Checkbox Functionality

Type: CONTROL

Detailed description: As a user, I want the system to handle the "MWP ACT Required" checkbox functionality so that it ensures the proposer meets the necessary marital status criteria and updates the "Further Requirements" block accordingly.  
  
Acceptance criteria:  
1. When the "MWP ACT Required" checkbox is checked:  
 - If the proposer's marital status is 'W', 'M', 'D', or 'N', the system should:  
 - Navigate to the "Further Requirements" block.  
 - Check if any record in the "Further Requirements" block has a test number 'M470'.  
 - If no such record exists, create a new record with test number 'M470' and description 'MWP ADDENDUM'.  
 - Set the date called to the current date if it is not already set.  
 - Set the raised by field to 'USER CALLED'.  
 - If the proposer's marital status is not 'W', 'M', 'D', or 'N', display a message indicating that the proposer must be married, divorced, or widowed for insurance with MWP Act and uncheck the checkbox.  
2. When the "MWP ACT Required" checkbox is unchecked:  
 - Navigate to the "Further Requirements" block.  
 - Delete any record with test number 'M470'.  
 - Delete related records from the `customer.azbj\_beneficiary\_trustee\_rep` table based on the contract ID.  
 - Commit the changes.  
  
Definition of Done:  
- The system correctly handles the "MWP ACT Required" checkbox functionality as per the acceptance criteria.  
- The system displays appropriate messages and updates the "Further Requirements" block and related tables accordingly.  
- All changes are committed to the database.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Delete related records from the `customer.azbj\_beneficiary\_trustee\_rep` table based on the contract ID:  
 ```sql  
 DELETE FROM customer.azbj\_beneficiary\_trustee\_rep WHERE contract\_id = :control.cn\_contract\_id;  
 COMMIT;  
 ```

# Handle ONLY BOP case, Auto-Issuance Allowed checkbox functionality

Type: CONTROL

Detailed description: As a user, I want the system to handle the "ONLY BOP case, Auto-Issuance Allowed" checkbox functionality so that specific conditions are checked and appropriate actions are taken based on the checkbox state.  
  
Acceptance criteria:  
1. When the "ONLY BOP case, Auto-Issuance Allowed" checkbox is checked:  
 - If the insured person's marital status is 'W', 'M', 'D', or 'N':  
 - Navigate to the "further\_req" section.  
 - Check if any record in "further\_req" has a test number 'M470'.  
 - If no such record exists, create a new record in "further\_req" with test number 'M470' and description 'MWP ADDENDUM'.  
 - Set the date called to the current date if it is not already set.  
 - Mark the record as raised by 'USER CALLED'.  
 - If the insured person's marital status is not 'W', 'M', 'D', or 'N':  
 - Display a message indicating that the proposer must be married, divorced, or widowed for insurance with the MWP Act.  
 - Uncheck the "ONLY BOP case, Auto-Issuance Allowed" checkbox.  
2. When the "ONLY BOP case, Auto-Issuance Allowed" checkbox is unchecked:  
 - Navigate to the "further\_req" section.  
 - Delete any record in "further\_req" with test number 'M470'.  
 - Delete related records from the "azbj\_beneficiary\_trustee\_rep" table based on the contract ID.  
 - Commit the transaction.  
  
Definition of Done:  
- The system correctly handles the checkbox state changes and performs the necessary actions as described in the acceptance criteria.  
- The user receives appropriate messages and feedback based on the conditions checked.  
- The related records are created, updated, or deleted as required.  
- The functionality is tested and verified to ensure it works as expected.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Delete from `customer.azbj\_beneficiary\_trustee\_rep` where `contract\_id` matches the control contract ID.  
- Commit the transaction.

# Navigate to Comments Section on Exit Button Press

Type: CONTROL

Title: Navigate to Comments Section on Exit Button Press  
  
User Story:  
As a user, I want to be able to exit the current screen and navigate to the comments section when I press the "Exit" button, so that I can quickly access and review comments.  
  
Acceptance Criteria:  
1. When the "Exit" button is pressed, the system should automatically navigate to the comments section.  
2. The comments section should be highlighted or focused upon navigation.  
3. The navigation should be seamless and without errors.  
  
Definition of Done:  
- The "Exit" button is functional and correctly navigates to the comments section.  
- The comments section is displayed and ready for user interaction upon navigation.  
- The navigation should be seamless and without errors.  
- All relevant tests are passed.  
- The feature is reviewed and approved by the product owner.  
  
Block Name: CONTROL

# Clear Button Functionality in Beneficial Ownership Screen

Type: CONTROL

Title: Clear Button Functionality in Beneficial Ownership Screen  
  
Acceptance Criteria:  
1. When the user presses the "Clear" button, the system should navigate to the CFT section.  
2. The system should then clear all the data in the CFT section, resetting it to its initial state.  
  
Definition of Done:  
- The "Clear" button is visible and accessible on the Beneficial Ownership screen.  
- Pressing the "Clear" button successfully navigates to the CFT section.  
- All data in the CFT section is cleared and reset to its initial state upon pressing the "Clear" button.  
- The functionality is tested and verified to work as expected.

# Populate Questions Functionality

Type: CONTROL

Title: Populate Questions Functionality  
  
Acceptance Criteria:  
1. When the "Populate question" button is pressed, the system should:  
 - Enable the fields for question description and question ID for insertion and updates.  
 - Navigate to the questions block and clear any existing data.  
 - Fetch the list of questions from the database.  
 - Populate the questions block with the fetched data.  
 - If the question ID is '1', '2', '3', or '4', set the check flag to 'Y'.  
 - Handle any errors that occur during the process and display an appropriate error message.  
 - After populating the data, disable the fields for question description and question ID for insertion and updates.  
  
Definition of Done:  
- The "Populate question" button functionality is implemented and tested.  
- The questions block is populated with data from the database when the button is pressed.  
- The check flag is set correctly for specific question IDs.  
- Error handling is in place and appropriate error messages are displayed.  
- The fields for question description and question ID are enabled and disabled as specified.  
  
DB queries for Table reference CRUD operations only(With Usage):  
```sql  
-- Fetching questions from the database  
SELECT question\_id, question\_desc   
FROM azbj\_crm\_questions;  
```

# Save Responses to Questions

Type: CONTROL

Title: Save Responses to Questions  
  
Acceptance Criteria:  
1. When the "Save" button is pressed, the system should check if there are any active questions for the given policy and contract.  
2. If active questions exist, they should be marked as inactive.  
3. The system should navigate through the questions and ensure that specific questions (IDs 1, 2, 3, 4) are answered.  
4. If any of these specific questions are not answered, an error message should be displayed.  
5. The system should then navigate through the questions again and insert the answered questions into the database.  
6. If an error occurs during the save process, it should be logged, and an appropriate error message should be displayed.  
7. Upon successful save, a confirmation message should be displayed.  
  
Definition of Done:  
- The system correctly identifies and deactivates any existing active questions for the given policy and contract.  
- The system validates that specific questions are answered before proceeding with the save.  
- The system successfully inserts the answered questions into the database.  
- Error handling and logging are implemented as specified.  
- A confirmation message is displayed upon successful save.  
  
DB queries for Table reference CRUD operations only(With Usage):  
```sql  
-- Check for existing active questions  
SELECT count() INTO v\_cnt   
FROM azbj\_crm\_questions\_detail  
WHERE policy\_no = :control.cn\_policy\_ref  
AND contract\_id = :control.cn\_contract\_id  
AND active\_questions ='Y';  
  
-- Deactivate existing active questions  
UPDATE azbj\_crm\_questions\_detail  
SET active\_questions ='N'  
WHERE policy\_no = :control.cn\_policy\_ref  
AND contract\_id = :control.cn\_contract\_id  
AND active\_questions ='Y';  
  
-- Insert new questions  
INSERT INTO azbj\_crm\_questions\_detail (policy\_no, contract\_id, question\_desc, question\_id, active\_questions, check\_flg, insert\_date, insert\_user)   
VALUES (:control.cn\_policy\_ref, :control.cn\_contract\_id, :azbj\_uw\_questions.question\_desc, :azbj\_uw\_questions.question\_id, 'Y', :azbj\_uw\_questions.check\_flg, SYSDATE, USER);  
```

# Manage Relationship with Staff in New Business Module

Type: CONTROL

Title: Manage Relationship with Staff in New Business Module  
  
Acceptance Criteria:  
1. The "Relation with Staff" field should be displayed on the "Policy Holder" tab.  
2. The field should be enabled for both insertion and updates.  
3. The field should be visually distinct with a white background and bold, plain text in Tahoma font.  
4. The field should be positioned at coordinates (621, 600) on the tab.  
5. The tab should be part of a window titled "New Business" with specific dimensions and scroll bar settings.  
  
Definition of Done:  
- The "Relation with Staff" field is visible and functional on the "Policy Holder" tab.  
- Users can insert and update information in the "Relation with Staff" field.  
- The field adheres to the specified visual and positional requirements.  
- The "New Business" window and "Policy Holder" tab are correctly configured and displayed.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Not applicable as the provided XML content does not include specific database queries.

# Manage CRM Comments for Specific Contract

Type: CONTROL

Title: Manage CRM Comments for Specific Contract  
  
Acceptance Criteria:  
1. When the "CRM comments" button is pressed, the system should:  
 - Enable the insertion and updating of CRM comments.  
 - Retrieve and display CRM questions and comments from the database for the specified contract.  
 - Populate the CRM comments section with the retrieved data.  
 - Populate specific control flags if the question description matches a predefined text.  
 - Disable the insertion and updating of CRM comments after the data is displayed.  
  
Definition of Done:  
- The "CRM comments" button should trigger the retrieval of CRM questions and comments from the database.  
- The CRM comments section should be populated with the retrieved data.  
- Specific control flags should be populated based on the question description.  
- The insertion and updating of CRM comments should be disabled after the data is displayed.  
- Error handling should be in place to display a message if any issues occur during the process.  
  
DB queries for Table reference CRUD operations only(With Usage):  
```sql  
SELECT question\_desc, description, crm\_user, crm\_date, flg1, flg2, flg3, flg4, flg5, flg6, flg7, flg8, flg9, flg10, flg11, flg12, flg13  
FROM azbj\_crm\_questions\_detail  
WHERE contract\_id = :control.cn\_contract\_id  
 AND check\_flg = 'Y'  
ORDER BY active\_questions DESC, TO\_NUMBER(question\_id);  
```

# Exit Button Functionality

Type: CONTROL

Title: As a user, I want to be able to exit the current view and return to the main tab view when I press the "Exit" button, so that I can navigate away from the current screen easily.  
  
Acceptance Criteria:  
1. When the "Exit" button is pressed, the system should:  
 - Move the focus to the "BEN\_OWNER" field in the "CONTROL" section.  
 - Hide the current view associated with the "CAN\_AML\_CFT" canvas.  
 - Hide the window titled "Beneficial Ownership".  
 - Display the main tab view associated with the "NBTABS" canvas.  
  
Definition of Done:  
- The "Exit" button should be functional and perform the specified actions when pressed.  
- The user should be able to see the main tab view after pressing the "Exit" button.  
- The "Beneficial Ownership" window should be hidden after the "Exit" button is pressed.  
- The focus should be set to the "BEN\_OWNER" field in the "CONTROL" section after the "Exit" button is pressed.

# Display PH Signature on Button Click

Type: CONTROL

Title: Display PH Signature on Button Click  
  
Acceptance Criteria:  
1. When the "Show PH Signature" button is clicked, the system should set a parameter `ph\_image` to 'Y'.  
2. The system should then execute the function to display the PH signature.  
  
Definition of Done:  
1. The "Show PH Signature" button is present on the IMAGE\_DETAILS tab.  
2. Clicking the button sets the `ph\_image` parameter to 'Y'.  
3. The PH signature is displayed upon clicking the button.  
  
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.

# Cancel Operation and Navigate to Insured Person Details

Type: CONTROL

Title: Cancel Operation and Navigate to Insured Person Details  
  
Acceptance Criteria:  
1. When the "Cancel" button is pressed, the system should set a flag to indicate that the save operation is not required.  
2. The system should hide the current view related to duplicate contact information.  
3. The system should navigate to the insured person's details screen.  
4. The focus should be set to the insured person's name field.  
  
Definition of Done:  
- The "Cancel" button is functional and performs the described actions.  
- The insured person's details screen is displayed after the cancel operation.  
- The insured person's name field is focused after navigation.  
- No errors or exceptions occur during the operation.  
  
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.

# Validate 'Proof of Relationship' Field for Special Characters

Type: CONTROL

Detailed description: As a user, I want to ensure that the 'Proof of Relationship' field does not accept any special characters so that the data entered is clean and valid.  
  
Acceptance criteria:  
1. The 'Proof of Relationship' field should not accept any special characters.  
2. If a user attempts to enter special characters, an error message should be displayed stating, 'Special characters are not allowed in Proof of Relationship.'  
  
Definition of Done:  
1. The 'Proof of Relationship' field is validated to ensure no special characters are allowed.  
2. An appropriate error message is displayed when special characters are entered.  
3. The feature is tested and confirmed to work 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.

# Save Button Functionality for Duplicate Contact Handling

Type: CONTROL

Title: Save Button Functionality for Duplicate Contact Handling  
  
Acceptance Criteria:  
- When the "Save" button is pressed, if a duplicate contact is detected, the system should:  
 - Set a flag indicating that the save button was pressed.  
 - Hide the duplicate contact notification view.  
 - Navigate to the insured person's details section.  
 - Focus on the insured person's name field.  
  
Definition of Done:  
- The "Save" button functionality is implemented and tested.  
- The system correctly identifies and handles duplicate contacts.  
- The user is navigated to the insured person's details section upon saving.  
- The insured person's name field is focused after navigation.  
- All acceptance criteria are met and verified through testing.

# View and Manage Enrichment Details

Type: CONTROL

Detailed description: As a user, I want to be able to view and manage enrichment details for an insured person by clicking a button, so that I can easily access and update relevant information.  
  
Acceptance criteria:  
1. When the user clicks the "Enrichment Details" button, the system should check if a parameter list named 'Param1' already exists.  
2. If 'Param1' exists, it should be destroyed and recreated.  
3. The system should add the following parameters to the newly created 'Param1':  
 - 'APPLICATION\_NO' with the value of either the insured person's sign card number or verification number.  
 - 'CALL\_FORM\_NAME' with the value 'NB'.  
 - 'PROPOSAL\_NO' with the value of the policy reference number.  
 - 'LA\_NAME' with the value of the insured person's name.  
4. The system should then call another form named 'AZBJ\_DATA\_ENRICH\_FORM' with the specified parameters.  
5. If any error occurs during this process, an error message should be displayed to the user.  
  
Definition of Done:  
- The "Enrichment Details" button is functional and performs the described actions.  
- The parameters are correctly passed to the 'AZBJ\_DATA\_ENRICH\_FORM'.  
- Error handling is implemented, and appropriate 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):  
- Not applicable as the provided XML content does not include any direct database queries.

# Confirm Relationship Status for Duplicate Number

Type: CONTROL

Title: Confirm Relationship Status for Duplicate Number  
  
Acceptance Criteria:  
1. When the user selects the option indicating that the number is correct:  
 - A message should be displayed prompting the user to approve the case from a supervisor.  
 - The "Approve" button should be enabled if it is currently disabled.  
 - The "Supervisor Check" checkbox should be made visible and enabled if it is currently hidden.  
 - The "Save" button should be disabled if it is currently enabled.  
2. When the user selects any other option:  
 - The "Approve" button should be disabled if it is currently enabled.  
 - The "Supervisor Check" checkbox should be hidden if it is currently visible.  
 - The "Save" button should be enabled if it is currently disabled.  
  
Definition of Done:  
- The user is able to select the relationship status of the person with the duplicate number.  
- The system displays appropriate messages and enables/disables buttons and checkboxes based on the selected relationship status.  
- All changes are saved and reflected correctly in the system.

# Handle Partner Type Selection and Validations

Type: CONTROL

Title: Handle Partner Type Selection and Validations  
  
Acceptance Criteria:  
1. When the partner type is selected, the system should:  
 - Validate the partner type and perform necessary checks.  
 - Retrieve and display relevant data from the database based on the selected partner type.  
 - Enable or disable certain fields based on the partner type.  
 - Populate fields with data from related tables if specific conditions are met.  
  
2. The system should execute the following database operations:  
 - Retrieve the permanent receipt date from the `azbj\_batch\_items` table based on the receipt number.  
 - Retrieve the mandate date from the `azbj\_system\_constants` table based on specific system types and codes.  
 - Retrieve the old contract ID from the `azbj\_policy\_contract\_ext` and `ocp\_interested\_parties` tables based on the policy holder's partner ID.  
 - Count records in the `azbj\_aml\_nb\_records\_NEW` table based on the contract ID.  
 - Retrieve the old policy reference from the `Ocp\_policy\_bases` table based on the old contract ID.  
 - Retrieve and loop through records from the `azbj\_aml\_constants` table based on the partner type and other conditions.  
 - Populate fields in the `aml` block with data from the `azbj\_aml\_nb\_records\_NEW` and `azbj\_aml\_proof\_master` tables based on the document type and partner type.  
  
Definition of Done:  
- The system correctly handles the selection of a partner type and performs all necessary validations and data retrievals.  
- All relevant fields are populated or updated based on the selected partner type and related data.  
- The system enables or disables fields as required based on the partner type.  
- All database operations are executed correctly, and the retrieved data is displayed appropriately.  
- The user can navigate to the next item in the form without any errors.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Retrieve permanent receipt date:  
 ```sql  
 SELECT perm\_receipt\_date  
 INTO vr\_receipt\_date  
 FROM azbj\_batch\_items  
 WHERE perm\_receipt\_no = :control.cn\_permrcpt\_no;  
 ```  
  
- Retrieve mandate date:  
 ```sql  
 SELECT date\_value  
 INTO v\_date\_mandate  
 FROM azbj\_system\_constants  
 WHERE sys\_type = 'KYC\_AGENCY'  
 AND sys\_code = 'KYC\_AGENCY\_CHANNEL'  
 AND char\_value = 'Y';  
 ```  
  
- Retrieve old contract ID:  
 ```sql  
 SELECT x.contract\_id  
 INTO v\_old\_contract\_id  
 FROM ( SELECT a.contract\_id, appln\_sign\_date  
 FROM azbj\_policy\_contract\_ext a, ocp\_interested\_parties b  
 WHERE a.contract\_id = b.contract\_id  
 AND partner\_id = :POLICY\_HOLDER.PH\_PART\_ID  
 AND top\_indicator = 'Y'  
 AND a.action\_code <> 'D'  
 AND b.action\_code <> 'D'  
 AND ip\_no = 1  
 ORDER BY APPLN\_SIGN\_DATE DESC) x  
 WHERE ROWNUM < 2;  
 ```  
  
- Count records in `azbj\_aml\_nb\_records\_NEW`:  
 ```sql  
 SELECT COUNT ()  
 INTO v\_cnt  
 FROM azbj\_aml\_nb\_records\_NEW  
 WHERE contract\_id = :control.cn\_contract\_id;  
 ```  
  
- Retrieve old policy reference:  
 ```sql  
 SELECT policy\_ref  
 INTO v\_old\_policy\_ref  
 FROM Ocp\_policy\_bases  
 WHERE contract\_id = v\_old\_contract\_id  
 AND top\_indicator = 'Y'  
 AND action\_code <> 'D';  
 ```  
  
- Retrieve and loop through records from `azbj\_aml\_constants`:  
 ```sql  
 FOR i IN  
 (SELECT DISTINCT a.  
 FROM azbj\_aml\_constants a  
 WHERE partner\_type = :control.cn\_partner\_type  
 AND (0 = (CASE  
 WHEN v\_total\_prem <= 10000  
 AND document\_type = 'LEGAL\_NAME'  
 AND :control.cn\_product\_id NOT IN (167, 215)  
 THEN 0  
 WHEN v\_total\_prem > 10000  
 AND v\_total\_prem < 100000  
 AND document\_type NOT IN ('SOURCE\_OF\_FUNDS1', 'SOURCE\_OF\_FUNDS2', 'SOURCE\_OF\_FUNDS3')  
 AND azbj\_pk\_nb\_general.product\_defn(:control.cn\_product\_id) NOT IN ('NEW\_RISK\_CARE', 'FAMILY\_CARE\_FIRST')  
 AND :control.cn\_product\_id NOT IN (9, 20, 51, 10, 83, 57, 167, 215)  
 THEN 0  
 WHEN v\_total\_prem >= 100000  
 AND azbj\_pk\_nb\_general.product\_defn(:control.cn\_product\_id) NOT IN ('NEW\_RISK\_CARE', 'FAMILY\_CARE\_FIRST')  
 AND :control.cn\_product\_id NOT IN (9, 20, 51, 10, 83, 57, 167, 215)  
 THEN 0  
 END)  
 OR (1 = (CASE  
 WHEN (:agents.ag\_agent\_code LIKE '100%' OR :agents.ag\_agent\_code LIKE '110%')  
 AND document\_type IN ('LEGAL\_NAME', 'PERMANENT\_ADDRESS', 'CURRENT\_ADDRESS')  
 THEN 1  
 ELSE 0  
 END)  
 AND vr\_receipt\_date >= v\_date\_mandate)))  
 LOOP  
 -- Populate fields in the `aml` block  
 END LOOP;  
 ```  
  
- Populate fields in the `aml` block:  
 ```sql  
 SELECT PROOF\_TYPE, OTHERS, VALUE, Expiry\_date, document\_remarks  
 INTO :AML.proof\_type, :aml.others, :aml.VALUE, :aml.Expiry\_date, :aml.document\_remarks  
 FROM azbj\_aml\_nb\_records\_NEW  
 WHERE CONTRACT\_ID = :control.cn\_contract\_id  
 AND DOCUMENT\_TYPE = i.document\_type;  
 ```  
  
- Retrieve proof description:  
 ```sql  
 SELECT proof\_desc  
 INTO :aml.proof\_desc  
 FROM azbj\_aml\_proof\_master  
 WHERE partner\_type = :control.cn\_partner\_type  
 AND document\_type = i.document\_type  
 AND proof\_type = :aml.proof\_type;  
 ```

# Manage Beneficial Ownership Details

Type: CONTROL

Title: Manage Beneficial Ownership Details  
  
Acceptance Criteria:  
1. When the "Beneficial Ownership" button is pressed, the system should display the "Beneficial Ownership" window and the "AML/CFT" view.  
2. The system should disable validation for the fields: identity document, identity proof, address ID, and address proof.  
3. The system should navigate to the "CFT" block and display the first record.  
4. The system should retrieve and display the following details for the selected contract from the database:  
 - Beneficial owner  
 - Owner share  
 - Color photo  
 - Identity proof description  
 - Identity document ID  
 - Identity document date  
 - Address proof description  
 - Address document ID  
 - Address document date  
 - Address  
 - Individual shareholding  
 - Family shareholdings  
 - Date of birth  
5. The system should handle any exceptions that occur during data retrieval gracefully.  
6. The system should re-enable validation for the fields: identity document, identity proof, address ID, and address proof.  
7. The system should navigate to the "CFT Exit" item.  
  
Definition of Done:  
- The "Beneficial Ownership" button correctly triggers the display of the relevant window and view.  
- The specified fields have their validation properties correctly toggled.  
- The "CFT" block is navigated to, and the first record is displayed.  
- Data is accurately retrieved from the database and displayed in the appropriate fields.  
- Any exceptions during data retrieval are handled without crashing the system.  
- Validation properties for the specified fields are re-enabled.  
- The system navigates to the "CFT Exit" item after completing the process.  
  
DB queries for Table reference CRUD operations only (With Usage):  
```sql  
SELECT beneficial\_owner, beb\_owner\_share, color\_photo,  
 identity\_proof\_desc, identity\_doc\_id,  
 identity\_doc\_date, address\_proof\_desc, address\_doc\_id,  
 address\_doc\_date, address, insert\_date, insert\_user,  
 individual\_shareholding, family\_shareholdings, date\_of\_birth  
FROM azbj\_beneficial\_ownership  
WHERE contract\_id = :control.cn\_contract\_id;  
```  
- This query retrieves beneficial ownership details for a specific contract from the `azbj\_beneficial\_ownership` table.

# View Suspected CP Details

Type: CONTROL

Title: View Suspected CP Details  
  
Acceptance Criteria:  
1. When the "Suspected CP Details" button is pressed, the system should set a flag indicating that the suspected screen is active.  
2. The system should display the view containing the suspected CP details.  
3. The system should navigate to the block containing the suspected CP details.  
4. If any error occurs during this process, an error message should be displayed to the user.  
  
Definition of Done:  
- The "Suspected CP Details" button is functional and triggers the appropriate actions.  
- The suspected CP details view is displayed correctly.  
- Navigation to the suspected CP details block is successful.  
- Error handling is implemented, and appropriate error messages are shown if any issues arise.  
  
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.

# Manage HO Allocation Value Removal

Type: CONTROL

Title: Manage HO Allocation Value Removal  
  
Acceptance Criteria:  
1. If the internal HO allocation value is null, the system should automatically set the removal flag to 'N'.  
2. The system should check if the user has the necessary authorization by querying the `user\_uw\_limits` table:  
 - If the user is authorized, and the removal flag is set to 'Y', the system should:  
 - Set the internal HO allocation value to null.  
 - Set the screen HO allocation value to null.  
 - Set the politically exposed person flag to 'N'.  
 - If the user is not authorized, the system should:  
 - Set the removal flag to 'N'.  
 - Display an error message indicating the user is not authorized to remove the HO allocation value.  
3. If the removal flag is set to 'Y', the system should set the politically exposed person flag to 'N'.  
  
Definition of Done:  
- The system correctly handles the removal of HO allocation values based on the specified conditions.  
- The system displays appropriate error messages when the user is not authorized.  
- The system updates the relevant fields (internal value, screen value, politically exposed person flag) as per the conditions.  
- All changes are saved and reflected in the database.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Query to check user authorization:  
 ```sql  
 SELECT COUNT(1)  
 INTO v\_count  
 FROM user\_uw\_limits  
 WHERE uw\_code = :v\_internal\_value  
 AND medical\_allowed = 'Y'  
 AND user\_id = :boiler.userid;  
 ```  
  
- Query to check user authorization with global value:  
 ```sql  
 SELECT COUNT(1)  
 INTO v\_count  
 FROM user\_uw\_limits  
 WHERE uw\_code = :GLOBAL.ho\_allocation\_internal\_value  
 AND medical\_allowed = 'Y'  
 AND user\_id = :boiler.userid;  
 ```

# Exit Button Functionality

Type: CONTROL

Title: Exit Button Functionality  
  
Acceptance Criteria:  
- When the "Exit" button is pressed, the current view should be hidden.  
- The system should navigate to the AML block.  
- The first record in the AML block should be displayed.  
- If any error occurs during this process, it should be handled gracefully without any visible error message to the user.  
  
Definition of Done:  
- The "Exit" button functionality is implemented and tested.  
- The current view is hidden upon pressing the "Exit" button.  
- Navigation to the AML block is successful.  
- The first record in the AML block is displayed.  
- Error handling is in place to ensure no visible error messages are shown to the user.  
  
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 Dropdown List for Matching Criteria

Type: CONTROL

Title: Implement Dropdown List for Matching Criteria  
  
Acceptance Criteria:  
- The dropdown list should have 5 predefined options.  
- The dropdown list should be labeled "Match Only By".  
- The dropdown list should be editable, allowing users to select an option.  
- The dropdown list should be positioned at coordinates (165, 75) on the form.  
- The dropdown list should have a width of 136 and a height of 21.  
- The dropdown list should be displayed on the "Details" canvas.  
  
Definition of Done:  
- The dropdown list is implemented and visible in the "Control" section of the form.  
- The dropdown list contains 5 predefined options.  
- The dropdown list is labeled "Match Only By".  
- The dropdown list is editable and allows users to select an option.  
- The dropdown list is correctly positioned and sized on the form.  
- The dropdown list is displayed on the "Details" canvas.

# Save Beneficial Ownership Details

Type: CONTROL

Title: Save Beneficial Ownership Details  
  
Acceptance Criteria:  
1. If the sum of shares of the controlling beneficiary exceeds 100%, an error message should be displayed.  
2. The system should retrieve the maximum allowed values for family and individual shareholdings from the system constants table.  
3. If the individual or family shareholding exceeds the retrieved maximum values and supervisor approval is not granted, an error message should be displayed.  
4. The system should check if there are existing records for the given contract ID in the beneficial ownership table. If records exist, they should be deleted.  
5. The system should iterate through the records in the CFT block and insert new records into the beneficial ownership table for each beneficial owner.  
6. A success message should be displayed upon successful data saving.  
7. If any errors occur during the process, an appropriate error message should be displayed.  
  
Definition of Done:  
- The user can save beneficial ownership details without exceeding the allowed shareholding limits.  
- The system retrieves and validates shareholding limits from the system constants table.  
- Existing records for the contract ID are deleted before inserting new records.  
- New records are inserted into the beneficial ownership table for each beneficial owner.  
- Appropriate success or error messages are displayed based on the outcome of the save operation.  
  
DB queries for Table reference CRUD operations only(With Usage):  
1. Retrieve family shareholding limit:  
 ```sql  
 SELECT NUM\_VALUE  
 INTO V\_FAMILY\_SHARE\_HOLDING  
 FROM azbj\_system\_constants  
 WHERE SYS\_TYPE = 'SHARE\_HOLDING' AND SYS\_CODE = 'FAMILY\_SHARE\_HOLDING';  
 ```  
  
2. Retrieve individual shareholding limit:  
 ```sql  
 SELECT NUM\_VALUE  
 INTO V\_INDIVIDUAL\_SHARE\_HOLDING  
 FROM azbj\_system\_constants  
 WHERE SYS\_TYPE = 'SHARE\_HOLDING' AND SYS\_CODE = 'INDIVIDUAL\_SHARE\_HOLDING';  
 ```  
  
3. Check for existing records:  
 ```sql  
 SELECT COUNT (ROWNUM)  
 INTO v\_cnt  
 FROM azbj\_beneficial\_ownership  
 WHERE contract\_id = :control.cn\_contract\_id;  
 ```  
  
4. Delete existing records:  
 ```sql  
 DELETE FROM azbj\_beneficial\_ownership  
 WHERE contract\_id = :control.cn\_contract\_id;  
 ```  
  
5. Insert new records:  
 ```sql  
 INSERT INTO azbj\_beneficial\_ownership  
 (contract\_id, beneficial\_owner, beb\_owner\_share, color\_photo, identity\_proof\_desc, identity\_doc\_id, identity\_doc\_date, address\_proof\_desc, address\_doc\_id, address\_doc\_date, address, insert\_date, insert\_user, INDIVIDUAL\_SHAREHOLDING, FAMILY\_SHAREHOLDINGS, approval\_id, date\_of\_birth)  
 VALUES (:control.cn\_contract\_id, :cft.ben\_owner, :cft.shares, :cft.photo, :cft.iden\_proof, :cft.iden\_doc, :cft.iden\_date, :cft.address\_proof, :cft.add\_id, :cft.add\_date, :cft.add\_address, SYSDATE, USER, :CONTROL.INDIVIDUAL\_SHAREHOLDING, :CONTROL.FAMILY\_SHAREHOLDINGS, :control.login, :CFT.DOB);  
 ```

# Exit Button Functionality in Risk Score Details

Type: CONTROL

Title: Exit Button Functionality in Risk Score Details  
  
Acceptance Criteria:  
1. When the "Exit" button is pressed, the system should enable the ability to insert and update records in the risk score details section.  
2. The system should then navigate the user to the control section.  
  
Definition of Done:  
- The "Exit" button should be functional and perform the described actions.  
- The user should be able to insert and update records in the risk score details section after pressing the "Exit" button.  
- The user should be navigated to the control section upon pressing the "Exit" button.

# Update Minor Life Details on Button Press

Type: CONTROL

Title: Update Minor Life Details on Button Press  
  
Acceptance Criteria:  
1. When the "Minor Life Details" button is pressed, the system should navigate to the "MLQ\_CLASS\_STD" section.  
2. The system should update the `IP\_MINOR\_QST\_FLAG` field with the value from the variable `v\_IP\_MINOR\_QST\_FLAG`.  
  
Definition of Done:  
1. The "Minor Life Details" button is visible and functional.  
2. Pressing the button navigates to the correct section.  
3. The `IP\_MINOR\_QST\_FLAG` field is updated with the correct value.  
4. All changes are saved and reflected in the system.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Not applicable as the provided XML content does not include any direct database queries.

# Cancel Operation and Navigate to Covers

Type: CONTROL

Title: Cancel Operation and Navigate to Covers  
  
Acceptance Criteria:  
1. When the "Cancel" button is pressed, the current alert window should be hidden.  
2. The view associated with the alert should also be hidden.  
3. The system should set the action status to 'W'.  
4. The policy status should be updated to match the current policy status from the SOFA\_INFO\_DATA.  
5. The system should navigate to the "Covers" tab.  
6. The "Covers" block should be activated and displayed.  
  
Definition of Done:  
- The "Cancel" button hides the alert window and view.  
- The action status is set to 'W'.  
- The policy status is updated correctly.  
- The system navigates to the "Covers" tab and activates the "Covers" block.

# Secure Password Entry in Beneficial Ownership Form

Type: CONTROL

Title: Secure Password Entry in Beneficial Ownership Form  
  
Acceptance Criteria:  
1. The password field should conceal the entered data to ensure security.  
2. The password field should have a distinct visual attribute to differentiate it from other text fields.  
3. Upon exiting the password field, the visual attribute should be applied to the field to maintain its distinct appearance.  
  
Definition of Done:  
- The password field is implemented and conceals the entered data.  
- The visual attribute is applied to the password field.  
- The visual attribute is maintained when the user exits the password field.  
- The functionality is tested and verified to meet the acceptance criteria.

# Supervisor ID Input Field

Type: CONTROL

Title: Supervisor ID Input Field  
  
Acceptance Criteria:  
- The input field should accept a maximum of 10 characters.  
- The input should be automatically converted to uppercase.  
- The input field should be visually distinct with a specific visual attribute applied after the user enters the text.  
  
Definition of Done:  
- The input field for Supervisor ID is implemented and visible on the user interface.  
- The input field accepts up to 10 characters and converts all input to uppercase.  
- The visual attribute is applied to the input field after the user enters the text.  
- The feature is tested and verified to ensure it meets the acceptance criteria.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Not applicable as the provided XML content does not include any database-specific queries.

# View and Manage Doctor Mobile Application Information

Type: CONTROL

Title: View and Manage Doctor Mobile Application Information  
  
Acceptance Criteria:  
1. When the "Doctor Mob App Info" button is pressed, the system should:  
 - Navigate to the mobile application information section.  
 - Clear any existing data in the section.  
 - Retrieve and display the doctor information based on the application number, ordered by the update date and client name.  
 - Populate the fields with the retrieved data, including doctor code, partner name, gender, age, test number, timestamp, longitude, and latitude.  
 - Ensure the first record is displayed initially.  
  
Definition of Done:  
- The button should be disabled by default.  
- The button should be labeled "Doctor Mob App Info".  
- The button should be positioned correctly on the interface.  
- The system should correctly retrieve and display the doctor information when the button is pressed.  
- The data should be displayed in the correct fields and in the correct order.  
- The first record should be displayed initially after data retrieval.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Retrieve doctor information based on the application number:  
 ```sql  
 SELECT   
 FROM AZBJ\_DOCT\_CUST\_DETAILS   
 WHERE application\_no = TO\_CHAR(NVL(:insured\_person.ip\_verf\_no, :insured\_person.ip\_sign\_card\_no))   
 ORDER BY upd\_date DESC, CLIENT\_NAME;  
 ```

# View Risk Score Details

Type: CONTROL

Detailed description: As a user, I want to view detailed risk scores for a specific contract or application number, so that I can assess the risk associated with it.  
  
Acceptance criteria:  
1. When the "Risk Score details" button is pressed, the system should navigate to the risk score details section.  
2. The system should clear any existing data in the risk score details section.  
3. The system should retrieve and display the following information for each relevant transaction:  
 - Application number  
 - Parameter name (rule description)  
 - Branch risk score (calculated based on the module)  
 - DE-QC risk score (calculated based on the module)  
 - Total score (sum of branch risk score, DE-QC risk score, and a default property value of 0)  
4. The system should ensure that the risk score details section is read-only (no insert or update allowed).  
5. If an error occurs during data retrieval, an appropriate error message should be displayed.  
  
Definition of Done:  
- The risk score details are displayed correctly when the button is pressed.  
- The data is retrieved and displayed according to the specified criteria.  
- The risk score details section is read-only.  
- Error handling is implemented and tested.  
  
DB queries for Table reference CRUD operations only(With Usage):  
```sql  
SELECT b.application\_no, rule\_desc AS param\_name, b.module,  
 (CASE  
 WHEN b.module IN ('SCRUTINY', 'BOTH') THEN a.risk\_value  
 ELSE 0  
 END) branch\_risk\_score,  
 (CASE  
 WHEN b.module IN ('DE-QC', 'BOTH') THEN a.risk\_value  
 ELSE 0  
 END) deqc\_risk\_score  
FROM azbj\_claim\_uw\_rule\_transaction a,  
 azbj\_claim\_uw\_transaction b,  
 azbj\_claim\_uw\_rules d  
WHERE rule\_action = 2  
 AND a.version\_no = b.version\_no  
 AND a.rule\_id = d.rule\_id  
 AND b.param\_id = d.param\_id  
 AND b.version\_no = (SELECT MAX(version\_no)  
 FROM azbj\_claim\_uw\_transaction c  
 WHERE c.application\_no = TO\_CHAR(NVL(:insured\_person.ip\_verf\_no, :insured\_person.ip\_sign\_card\_no))  
 AND c.module IN ('SCRUTINY', 'DE-QC', 'BOTH'))  
 AND a.trans\_id = b.trans\_id  
 AND b.application\_no = TO\_CHAR(NVL(:insured\_person.ip\_verf\_no, :insured\_person.ip\_sign\_card\_no))  
 AND b.module IN ('SCRUTINY', 'DE-QC', 'BOTH');  
```

# User Authentication

Type: CONTROL

Detailed description: As a user, I want to authenticate my login credentials to access the system, ensuring that only authorized personnel can proceed.  
  
Acceptance criteria:  
1. If the login ID is not in the list of authorized supervisors ('P00O3687', 'P00O4361'), display a warning message: "Please enter proper Supervisor ID" and prompt the user to re-enter the login ID.  
2. If the login ID is empty, display a warning message: "Please enter Supervisor ID" and prompt the user to enter the login ID.  
3. If the password is empty, display a warning message: "Please enter Password" and prompt the user to enter the password.  
4. If the login ID and password are provided, call the approval procedure to validate the credentials.  
5. If the credentials are approved, display a message: "Approved" and set the approval flag to 'Y'.  
6. If the credentials are not approved, set the approval flag to 'N' and display the error message returned by the approval procedure.  
7. Handle any unexpected errors by logging the error message and ensuring the system remains stable.  
  
Definition of Done:  
- The user can enter their login ID and password.  
- The system validates the login ID and password according to the specified criteria.  
- Appropriate messages are displayed based on the validation results.  
- The approval procedure is called to validate the credentials.  
- The system handles unexpected errors gracefully.  
  
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.

# Supervisor Approval for Duplicate Contact Numbers

Type: CONTROL

Title: Supervisor Approval for Duplicate Contact Numbers  
  
Acceptance Criteria:  
1. When the "Supervisor Approval" button is pressed, the system should check if at least one policy is selected.  
2. If no policy is selected, an error message should be displayed indicating that it is mandatory to select at least one policy.  
3. If a policy is selected, the system should prompt for supervisor authentication.  
4. If authentication fails, an error message should be displayed.  
5. If authentication is successful, the "Save" button should be enabled if it was previously disabled.  
  
Definition of Done:  
- The "Supervisor Approval" button functionality is implemented and tested.  
- Error messages are displayed correctly when no policy is selected or authentication fails.  
- The "Save" button is enabled upon successful authentication.  
- The feature is tested and verified to work as expected in the user interface.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Not applicable as the provided XML content does not include any direct database queries.

# Enable/Disable SPW Percentage Field Based on SPW Flag Checkbox

Type: CONTROL

Title: Enable/Disable SPW Percentage Field Based on SPW Flag Checkbox  
  
Acceptance Criteria:  
1. When the SPW Flag checkbox is checked (value 'Y'), the SPW Percentage field should be enabled.  
2. When the SPW Flag checkbox is unchecked (value 'N'), the SPW Percentage field should be disabled.  
  
Definition of Done:  
- The SPW Percentage field is enabled when the SPW Flag checkbox is checked.  
- The SPW Percentage field is disabled when the SPW Flag checkbox is unchecked.  
- The changes are tested and verified to work as expected in the user interface.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include any database-specific queries.

# Calculate PPC on Button Press

Type: CONTROL

Title: Calculate PPC on Button Press  
  
Acceptance Criteria:  
1. When the "Calculate" button is pressed, the system should execute a predefined calculation process.  
2. The calculation process should initialize two variables, `v\_ip\_other\_sa` and `v\_ph\_other\_sa`, both set to 0.  
3. The calculation process should call a function named `azbj\_calculate\_ppc` to perform the actual PPC calculation.  
  
Definition of Done:  
- The "Calculate" button is present on the user interface.  
- Pressing the "Calculate" button triggers the calculation process.  
- The variables `v\_ip\_other\_sa` and `v\_ph\_other\_sa` are initialized to 0.  
- The function `azbj\_calculate\_ppc` is called and executed successfully.  
- The user is able to see the results of the PPC calculation.  
  
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 CRUD operations.

# Implement SPW Freq Dropdown Field

Type: CONTROL

Title: Implement SPW Freq Dropdown Field  
  
Acceptance Criteria:  
1. The "SPW Freq" field should display a dropdown list with four predefined frequency options.  
2. The field should be positioned correctly within the form, ensuring it is easily accessible to the user.  
3. The field should accept only uppercase characters.  
4. The field should allow both insertion and updating of data.  
5. The field should be visually distinct with a white background and black text.  
  
Definition of Done:  
1. The "SPW Freq" field is implemented and displays a dropdown list with the predefined options.  
2. The field is correctly positioned and styled as per the requirements.  
3. The field accepts only uppercase characters.  
4. The field allows data to be inserted and updated.  
5. The implementation is tested and verified to meet all acceptance criteria.  
  
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 table references.

# Exit Button Functionality

Type: CONTROL

Title: Exit Button Functionality  
  
Acceptance Criteria:  
1. When the "Exit" button is pressed, the current view should be hidden.  
2. The window associated with the current view should also be hidden.  
3. The focus should be moved to a specific item on the previous screen, ensuring a smooth transition.  
  
Definition of Done:  
- The "Exit" button hides the current view and window.  
- The focus is successfully moved to the specified item on the previous screen.  
- The functionality is tested and confirmed to work as expected.  
  
SQL query for reference:  
- Not applicable as the provided XML content does not include any database CRUD operations.

# Dynamic Update of Proof Types and Financial Details Based on Premium Payer Selection

Type: CONTROL

Detailed description: As a user, I want the system to dynamically update the list of proof types and financial details based on the selected premium payer, so that I can ensure the correct financial information is displayed and processed for the selected payer.  
  
Acceptance criteria:  
1. When the premium payer is selected, the system should:  
 - Default the premium payer to 1 if no value is selected.  
 - Clear the existing list of proof types.  
 - Populate the list of proof types based on the selected premium payer's occupation status and predefined categories.  
 - Include additional proof types such as 'Not Received', 'Proposal Form', and 'Income Estimator'.  
  
2. The system should retrieve and display financial details for the selected premium payer from the financial underwriting table, including:  
 - Years  
 - Income tax returns  
 - Computation  
 - Proof type  
 - Gross income  
 - Exempted income  
 - One-time income  
 - Deductions  
 - Tax  
 - Income tax return filing date  
  
3. If the premium payer is 3, the system should:  
 - Retrieve and display spouse financial details from the spouse financial underwriting table.  
 - Calculate and display the net profit difference between the average net profit and the spouse's net income.  
  
4. The system should also retrieve and display liquid investment details for the selected premium payer, including:  
 - Fixed term deposit  
 - Mutual fund  
 - Equity share  
 - Fund value of unit-linked policies  
 - Bank balance  
 - One-time income  
 - Total liquid investment  
  
5. The system should retrieve and display beneficial ownership details for the contract, including:  
 - Beneficial owner  
 - Shares owned by the beneficial owner  
  
6. The visibility and enablement of certain fields should be dynamically adjusted based on the selected premium payer.  
  
Definition of Done:  
- The system correctly updates the list of proof types and financial details based on the selected premium payer.  
- All relevant financial details are retrieved and displayed accurately.  
- The system handles exceptions gracefully without crashing.  
- The visibility and enablement of fields are dynamically adjusted as per the premium payer selection.  
- The functionality is tested and verified to ensure it meets the acceptance criteria.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Retrieve proof types based on occupation status and predefined categories.  
- Retrieve financial details for the selected premium payer.  
- Retrieve liquid investment details for the selected premium payer.  
- Retrieve spouse financial details if the premium payer is 3.  
- Retrieve beneficial ownership details for the contract.

# Refresh Suspected Partner Details

Type: CONTROL

Detailed description: As a user, I want to refresh the details of suspected partners based on specific matching criteria (Name, City, Date of Birth, or a combination of these) so that I can view the most relevant and updated information in the system.  
  
Acceptance criteria:  
1. When the refresh button is pressed, the system should check if the "MATCH\_ONLY\_BY" field is not null.  
2. If the "MATCH\_ONLY\_BY" field is not null, the system should retrieve terror details and perform the following:  
 - If the partner type is 'P', fetch the first name, middle name, surname, and date of birth from the `cp\_partners` table for the insured person.  
 - If the insured person ID is different from the policy holder ID, verify matching terror details for both the insured person and policy holder.  
 - If the partner type is not 'P', fetch the first name, middle name, surname, and date of birth for the insured person and the institution name for the policy holder.  
 - Verify matching terror details for both the insured person and policy holder.  
3. The system should then fetch the city from the `cp\_addresses` table based on the address ID.  
4. The system should combine the matching lists for the insured person and policy holder into a final list.  
5. Based on the "MATCH\_ONLY\_BY" field value, the system should:  
 - If 'NAME' or 'ALL', display the details of all matched records.  
 - If 'CITY', display the details of matched records where the city matches.  
 - If 'DOB', display the details of matched records where the date of birth matches.  
 - If 'DOB\_CITY', display the details of matched records where both the date of birth and city match.  
6. The system should update the total count of matched records and store it in a global variable.  
  
Definition of Done:  
- The refresh button functionality is implemented and tested.  
- The system retrieves and displays the correct details based on the matching criteria.  
- The total count of matched records is updated and stored correctly.  
- The functionality is verified to work without any Oracle Forms-specific terminology or dependencies.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Fetching partner details:  
 ```sql  
 SELECT TRIM(UPPER(first\_name)), TRIM(UPPER(middle\_name)), TRIM(UPPER(surname)), DATE\_OF\_BIRTH  
 INTO v\_fname\_t\_ip, v\_mname\_t\_ip, v\_lname\_t\_ip, v\_dob  
 FROM cp\_partners  
 WHERE part\_id = :insured\_person.ip\_part\_id;  
 ```  
  
- Fetching city details:  
 ```sql  
 SELECT ADDRESS\_LINE4  
 INTO v\_city  
 FROM cp\_addresses  
 WHERE add\_id = v\_add\_id;  
 ```  
  
- Fetching institution name:  
 ```sql  
 SELECT TRIM(UPPER(institution\_name))  
 INTO v\_fname\_t\_ph  
 FROM cp\_partners  
 WHERE part\_id = :policy\_holder.ph\_part\_id;  
 ```  
  
- Fetching policy holder details:  
 ```sql  
 SELECT TRIM(UPPER(first\_name)), TRIM(UPPER(middle\_name)), TRIM(UPPER(surname)), DATE\_OF\_BIRTH  
 INTO v\_fname\_t\_ph, v\_mname\_t\_ph, v\_lname\_t\_ph, v\_dob  
 FROM cp\_partners  
 WHERE part\_id = :policy\_holder.ph\_part\_id;  
 ```

# Double-click to Edit DRC Category

Type: CONTROL

Title: Double-click to Edit DRC Category  
  
Acceptance Criteria:  
1. When the user double-clicks on the "DRC Category" field, an editor should open at a specified position on the screen.  
2. The editor should display the current content of the "DRC Category" field.  
3. The user should be able to edit the content within the editor.  
4. Upon closing the editor, if the user confirms the changes, the updated content should be saved back to the "DRC Category" field.  
  
Definition of Done:  
- The editor opens correctly when the "DRC Category" field is double-clicked.  
- The current content of the "DRC Category" field is displayed in the editor.  
- The user can edit the content and confirm the changes.  
- The updated content is saved back to the "DRC Category" field upon confirmation.  
- The feature is tested and verified to work 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.

# Display RCU Reasons Button and Trigger Form

Type: CONTROL

User Story: Display RCU Reasons Button and Trigger Form  
  
Detailed description:   
As a user, I want to have a button labeled "RCU Reasons" on the interface, which when pressed, will open a form to display and manage RCU comments related to a specific policy. This button should be visible and enabled based on certain conditions, and it should pass relevant parameters to the new form.  
  
Acceptance criteria:  
1. The button labeled "RCU Reasons" should be visible on the interface.  
2. When the button is pressed, it should check if a parameter list named 'Param1' exists. If it does, it should destroy the existing parameter list and create a new one.  
3. The new parameter list should include the following parameters:  
 - 'POLICY\_NO' with the value of the current policy reference.  
 - 'MODULE\_NAME' with the value 'BBU'.  
4. The form 'AZBJ\_RCU\_COMMENTS' should be called with the new parameter list, ensuring the form is not hidden, replaces the current form, and is not query-only.  
  
Definition of Done:  
- The "RCU Reasons" button is displayed on the interface.  
- Pressing the button successfully opens the 'AZBJ\_RCU\_COMMENTS' form with the correct parameters.  
- The form behavior adheres to the specified conditions (no hide, do replace, no query only).  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Not applicable as the provided XML content does not include direct CRUD operations on the database.  
  
Explanation of Oracle Form Logic:  
- The logic involves checking for an existing parameter list and destroying it if found.  
- A new parameter list is created and populated with specific parameters.  
- The form 'AZBJ\_RCU\_COMMENTS' is called with the new parameter list, ensuring it adheres to the specified display and behavior conditions.

# Agent Code Validation and Retrieval

Type: CONTROL

Title: Agent Code Validation and Retrieval  
  
Acceptance Criteria:  
1. When an agent code is entered, the system should check if the code is not null.  
2. The system should retrieve the unique code and internal ID of the agent from the `dmt\_agents` table based on the entered agent code.  
3. The system should fetch the agent's details from the `CP\_PARTNERS` table using the retrieved unique code.  
4. The agent's name should be constructed by concatenating various name fields from the `CP\_PARTNERS` table.  
5. If the agent's name is null, it should be set to the name field from the `CP\_PARTNERS` table.  
6. The system should retrieve the agent's license number, branch code, and license expiration date from the `azbj\_agents\_ext` table using the internal ID.  
7. If no data is found during any of the retrievals, an appropriate message should be displayed, and the agent code should be cleared.  
8. The system should determine the unit manager for the agent based on specific conditions and retrieve the unit manager's name from the `DM\_V\_AGENT\_ASSIGNMENTS` table.  
9. If the agent code starts with '100%', the specified person sub IC code should be set to the agent code.  
10. If the agent code starts with '511%' or '5100000002%', the customer ID field should be enabled; otherwise, it should be disabled and cleared.  
  
Definition of Done:  
- The agent code validation and retrieval process is implemented and tested.  
- The agent's details are correctly displayed based on the entered agent code.  
- Appropriate error messages are shown when no data is found.  
- The unit manager's details are correctly retrieved and displayed.  
- The specified person sub IC code and customer ID field are correctly set based on the agent code.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Retrieve unique code and internal ID from `dmt\_agents`:  
 ```sql  
 SELECT CUST\_PART\_UNIQUE\_CODE, int\_id  
 INTO AZBJ\_MC\_PART\_ID, azbj\_mn\_int\_id  
 FROM dmt\_agents  
 WHERE reference\_code = :ag\_agent\_code;  
 ```  
  
- Fetch agent's details from `CP\_PARTNERS`:  
 ```sql  
 SELECT   
 INTO AZBJ\_MC\_PARTNER  
 FROM CP\_PARTNERS  
 WHERE CP\_PARTNERS.PARTNER\_REF = AZBJ\_MC\_PART\_ID;  
 ```  
  
- Retrieve agent's license details from `azbj\_agents\_ext`:  
 ```sql  
 SELECT licence\_no, branch\_code, license\_end\_date  
 INTO :ag\_agent\_lic\_no, :ag\_branch\_code, :ag\_lic\_exp\_date  
 FROM azbj\_agents\_ext  
 WHERE int\_id = azbj\_mn\_int\_id;  
 ```  
  
- Retrieve unit manager's name from `DM\_V\_AGENT\_ASSIGNMENTS`:  
 ```sql  
 SELECT PARTNER\_NAME  
 INTO :agents.ag\_unit\_mgr\_name  
 FROM DM\_V\_AGENT\_ASSIGNMENTS  
 WHERE REFERENCE\_CODE = :agents.ag\_unit\_mgr  
 AND :CONTROL.CN\_EFFECTIVE\_DATE >= START\_DATE  
 AND :CONTROL.CN\_EFFECTIVE\_DATE <= NVL(END\_DATE, :CONTROL.CN\_EFFECTIVE\_DATE);  
 ```  
  
- Determine the unit manager using the `GET\_MANAGER` function:  
 ```sql  
 FUNCTION GET\_MANAGER(p\_agent\_code IN VARCHAR2, p\_activity\_date IN DATE)  
 RETURN VARCHAR2 IS  
 azbj\_mn\_parent\_int\_id NUMBER;  
 azbj\_mv\_parent\_code VARCHAR2(30);  
 v\_azbj\_status VARCHAR2(30);  
 BEGIN  
 SELECT b.recruited\_by, agents\_status  
 INTO azbj\_mv\_parent\_code, v\_azbj\_status  
 FROM dmt\_agents a, azbj\_agents\_ext b  
 WHERE a.reference\_code = p\_agent\_code  
 AND a.int\_id = b.int\_id;  
  
 IF v\_azbj\_status LIKE 'T%' THEN  
 SELECT stm\_code  
 INTO azbj\_mv\_parent\_code  
 FROM azbj\_cashier\_receipt  
 WHERE perm\_receipt\_no = :control.cn\_permrcpt\_no;  
 END IF;  
  
 RETURN(azbj\_mv\_parent\_code);  
 EXCEPTION  
 WHEN others THEN  
 RETURN(' ');  
 END;  
 ```  
  
- Display appropriate messages using the `AZBJ\_MESSAGE` procedure:  
 ```sql  
 PROCEDURE azbj\_message(al\_type IN VARCHAR2, al\_message IN VARCHAR2) IS  
 mv\_alert\_name VARCHAR2(30);  
 mn\_dummy NUMBER;  
 BEGIN  
 IF al\_type = 'W' THEN  
 mv\_alert\_name := 'ALERT\_WARNING';  
 ELSIF al\_type = 'E' THEN  
 mv\_alert\_name := 'ALERT\_ERROR';  
 ELSIF al\_type = 'Q' THEN  
 mv\_alert\_name := 'ALERT\_QUESTION';  
 END IF;  
  
 SET\_ALERT\_PROPERTY(mv\_alert\_name, ALERT\_MESSAGE\_TEXT, al\_message);  
 mn\_dummy := SHOW\_ALERT(mv\_alert\_name);  
 IF al\_type = 'E' THEN  
 RAISE FORM\_TRIGGER\_FAILURE;  
 END IF;  
 IF al\_type = 'Q' THEN  
 IF mn\_dummy = ALERT\_BUTTON2 THEN  
 RAISE FORM\_TRIGGER\_FAILURE;  
 END IF;  
 END IF;  
 END;  
 ```

# Input Field for Percentage Increase in SA

Type: CONTROL

Title: Input Field for Percentage Increase in SA  
  
Acceptance Criteria:  
1. The input field for the percentage increase in SA should be visible and allow users to enter numeric values.  
2. The input should be restricted to uppercase characters.  
3. The input field should have a maximum length of 30 characters.  
4. The input field should be located at the specified position on the form.  
5. The input field should be part of a tabbed interface and should be displayed on the "COVERS" tab.  
6. The input field should have a prompt labeled "% Increasing in SA" aligned to the right.  
7. The input field should be enabled for both insertion and updates.  
  
Definition of Done:  
1. The input field for the percentage increase in SA is implemented and visible on the form.  
2. Users can enter numeric values up to 30 characters in length.  
3. The input field is positioned correctly on the form and is part of the "COVERS" tab.  
4. The prompt "% Increasing in SA" is displayed and aligned to the right.  
5. The input field allows for both insertion and updates.  
6. The input field is tested and verified to meet all acceptance criteria.  
  
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.

# Call CRIF Score Functionality

Type: CONTROL

Title: Call CRIF Score Functionality  
  
Acceptance Criteria:  
1. The system should prompt the user with a confirmation message if the CIBIL Score is null or zero.  
2. Upon confirmation, the system should clear any existing CRIF Score list.  
3. The system should retrieve the CRIF Score list from the database where the system type is 'CIBIL' and the system code is 'CRIF\_LIST'.  
4. The retrieved CRIF Score list should be added to the list element in the system.  
5. The CRIF Score list should be made visible and enabled if it is not already.  
6. The focus should be set to the CRIF Score list item after it is populated.  
  
Definition of Done:  
- The user is prompted with a confirmation message when the CIBIL Score is null or zero.  
- The CRIF Score list is cleared and repopulated with data from the database.  
- The CRIF Score list is made visible and enabled.  
- The focus is set to the CRIF Score list item.  
- All functionalities are tested and verified to be working as expected.  
  
DB queries for Table reference CRUD operations only (With Usage):  
```sql  
SELECT sys\_desc, char\_value  
FROM azbj\_system\_constants  
WHERE sys\_type = 'CIBIL' AND sys\_code = 'CRIF\_LIST';  
```  
- This query is used to retrieve the CRIF Score list from the database.

# Implement and Manage 'Cash Bonus' Field

Type: CONTROL

Title: Implement and Manage 'Cash Bonus' Field  
  
Acceptance Criteria:  
1. The 'Cash Bonus' field should allow users to input and update values.  
2. The input value should be automatically converted to uppercase.  
3. The field should be hidden by default and only become visible under certain conditions.  
4. The field should be displayed on the 'COVERS' tab of the user interface.  
5. The prompt for the 'Cash Bonus' field should be aligned to the right and displayed in bold font.  
  
Definition of Done:  
1. The 'Cash Bonus' field is implemented and allows for input and updates.  
2. The input value is automatically converted to uppercase.  
3. The field visibility is controlled based on specific conditions.  
4. The field is correctly placed on the 'COVERS' tab.  
5. The prompt for the field is right-aligned and displayed in bold font.  
  
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 Relationship to Child Dropdown

Type: CONTROL

Detailed description: As a user, I want to select the relationship of a child from a predefined list so that I can accurately specify the relationship in the system.  
  
Acceptance criteria:  
1. The user should be able to see a dropdown list labeled "Relationship to Child" on the form.  
2. The dropdown list should contain 14 predefined relationship options.  
3. The dropdown list should be positioned appropriately on the form for easy access.  
4. The form should display the "Nominee/Child Cover Details" window when this section is accessed.  
5. The window should be modal, preventing interaction with other windows until it is closed.  
6. The window should have a fixed size and position, and should not allow resizing, minimizing, or maximizing.  
  
Definition of Done:  
1. The dropdown list for "Relationship to Child" is implemented and displays 14 predefined options.  
2. The "Nominee/Child Cover Details" window is modal and has a fixed size and position.  
3. The form is tested to ensure that the dropdown list and window behave as expected.  
4. 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 specific database operations.

# Control Visibility of Supervisor List Based on Checkbox

Type: CONTROL

Title: Control Visibility of Supervisor List Based on Checkbox  
  
Acceptance Criteria:  
1. When the "Refer To Supervisor" checkbox is unchecked:  
 - The supervisor list should be hidden.  
2. When the "Refer To Supervisor" checkbox is checked:  
 - The supervisor list should be visible.  
  
Definition of Done:  
- The checkbox labeled "Refer To Supervisor" is present and functional.  
- The visibility of the supervisor list changes dynamically based on the checkbox state.  
- The changes are tested and verified to ensure they work as expected.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided logic does not involve direct CRUD operations on the database.

# Add Senior Underwriter Button Functionality

Type: CONTROL

Title: Add Senior Underwriter Button Functionality  
  
Acceptance Criteria:  
1. When the "Add Senior Underwriter" button is pressed, a new view should be displayed to the user.  
2. The focus should automatically move to the input field for the senior underwriter's user ID.  
3. If any error occurs while displaying the new view, an appropriate error message should be shown to the user.  
  
Definition of Done:  
- The "Add Senior Underwriter" button is present on the user interface.  
- Pressing the button displays a new view for adding a senior underwriter.  
- The input field for the senior underwriter's user ID is focused automatically.  
- Error handling is implemented to show a message if an error occurs while displaying the new view.  
  
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 NRI GST Waiver Checkbox Functionality

Type: CONTROL

Title: Implement NRI GST Waiver Checkbox Functionality  
  
Acceptance Criteria:  
1. When the NRI GST Waiver checkbox is checked:  
 - The form status should be set to 'Y'.  
 - The variable `v\_nri\_gst\_waiver` should be updated with the checkbox value.  
 - If the `BKD\_FLG` field is 'Y', the `CH\_INCEPTION\_DATE` field should be enabled.  
 - If the `BKD\_FLG` field is not 'Y', the `CH\_INCEPTION\_DATE` field should be disabled.  
 - If the `v\_nri\_gst\_waiver` variable is 'NGST', the `NRI\_GST\_WAIVER` field in the agents section should be made visible and set to 'NRI GST WAIVER'.  
  
2. When the NRI GST Waiver checkbox is unchecked:  
 - The form status should be set to 'Y'.  
 - The variable `v\_nri\_gst\_waiver` should be updated with the checkbox value.  
 - If the `BKD\_FLG` field is 'Y', the `CH\_INCEPTION\_DATE` field should be enabled.  
 - If the `BKD\_FLG` field is not 'Y', the `CH\_INCEPTION\_DATE` field should be disabled.  
 - The `NRI\_GST\_WAIVER` field in the agents section should remain hidden.  
  
Definition of Done:  
- The NRI GST Waiver checkbox functionality is implemented as per the acceptance criteria.  
- The form status, variable updates, and field properties are correctly handled based on the checkbox status.  
- The `CH\_INCEPTION\_DATE` field's enabled/disabled state is correctly toggled based on the `BKD\_FLG` field value.  
- The `NRI\_GST\_WAIVER` field in the agents section is correctly shown or hidden based on the `v\_nri\_gst\_waiver` variable value.  
- All changes are tested and verified to ensure they work as expected.

# Enable/Disable 'OK' Button Based on Underwriting Decision

Type: CONTROL

User Story:  
As a user, I want the system to enable or disable the "OK" button based on the selected underwriting decision so that I can proceed with the appropriate action.  
  
Acceptance Criteria:  
1. When the user selects an underwriting decision of 'TIF', 'CC', or 'CNC', the "OK" button should be enabled if it is currently disabled.  
2. When the user selects any other underwriting decision, the "OK" button should be disabled if it is currently enabled.  
  
Definition of Done:  
- The "OK" button's state (enabled/disabled) should change dynamically based on the selected underwriting decision.  
- The system should handle exceptions gracefully without causing any errors or crashes.  
  
Block Name: CONTROL

# Add Senior Underwriter

Type: CONTROL

Detailed description: As a user, I want to add a new Senior Underwriter (SU) to the system, so that the SU can be recognized and utilized within the application.  
  
Acceptance criteria:  
1. When the "Add" button is pressed, the system should check if the SU User ID is not null.  
2. The system should verify if the SU User ID already exists in the Senior Underwriter list.  
 - If the SU User ID exists, display a warning message: "User ID is already available in Senior UnderWriter List".  
 - If the SU User ID does not exist, insert the new SU User ID into the system with the following details:  
 - `sys\_type` as 'SU'  
 - `sys\_code` as 'SENIOR\_UW'  
 - `sys\_desc` as the username associated with the SU User ID  
 - `char\_value` as the SU User ID  
 - `low\_range` as 1 if a specific condition is met, otherwise NULL  
 - `start\_date` as the current date  
3. Commit the transaction after the insertion.  
4. Hide the current view and navigate to the "Add SU User" item.  
  
Definition of Done:  
- The "Add" button functionality is implemented and tested.  
- The system correctly checks for the existence of the SU User ID.  
- Appropriate messages are displayed based on the existence check.  
- New SU User IDs are successfully inserted into the system.  
- The transaction is committed, and the view is updated as specified.  
  
DB queries for Table reference CRUD operations only(With Usage):  
```sql  
-- Check if SU User ID exists  
SELECT COUNT()  
INTO V\_EXIST\_CNT  
FROM AZBJ\_SYSTEM\_CONSTANTS  
WHERE SYS\_TYPE = 'SU' AND CHAR\_VALUE = :CONTROL.SU\_USER\_ID;  
  
-- Insert new SU User ID  
INSERT INTO AZBJ\_SYSTEM\_CONSTANTS (sys\_type, sys\_code, sys\_desc, char\_value, low\_range, start\_date)  
VALUES ('SU', 'SENIOR\_UW', az\_pk2\_general.getusername(:CONTROL.SU\_USER\_ID), :CONTROL.SU\_USER\_ID, CASE WHEN :control.CHK\_SU = 'Y' THEN 1 ELSE NULL END, SYSDATE);  
```

# Enable/Disable 'Remove Split Reason' based on 'Remove Split Policy' checkbox

Type: CONTROL

Title: Enable/Disable 'Remove Split Reason' based on 'Remove Split Policy' checkbox  
  
Acceptance Criteria:  
1. When the 'Remove Split Policy' checkbox is checked, the 'Remove Split Reason' field should be enabled.  
2. When the 'Remove Split Policy' checkbox is unchecked, the 'Remove Split Reason' field should be disabled.  
3. The system should handle any exceptions gracefully without causing a crash.  
  
Definition of Done:  
- The 'Remove Split Reason' field is enabled when the 'Remove Split Policy' checkbox is checked.  
- The 'Remove Split Reason' field is disabled when the 'Remove Split Policy' checkbox is unchecked.  
- The system handles exceptions without crashing.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include any specific database queries.

# Automatic Retrieval and Update of CRIF Score and Income Segment

Type: CONTROL

Detailed description: As a user, I want the system to automatically retrieve and update the CRIF score and income segment for an insured person when the CRIF list item is changed, so that I can have the most up-to-date information without manual intervention.  
  
Acceptance criteria:  
1. When the CRIF list item is changed, the system should set a flag `v\_crif\_flag` to 'N'.  
2. The system should call a function to get the CRIF CB score using the insured person's verification number or sign card number, policy reference, and user information.  
3. If the `cibil\_score` is zero, the system should:  
 - Attempt to retrieve the `cibil\_score` from the `azbj\_crif\_cbil\_score\_dtls` table where the application number matches the insured person's verification number or sign card number, the top indicator is 'Y', and the element name is 'CB\_SCORE\_SCORE\_VALUE'.  
 - If no data is found, attempt to retrieve the `cibil\_score` from the `azbj\_annuity\_prod\_det` table where the application number matches the insured person's verification number or sign card number, the top indicator is 'Y', and the product ID matches the control's product ID.  
 - If no data is found in both cases, set the `cibil\_score` to NULL.  
4. The system should retrieve the `income\_segment` from the `azbj\_crif\_cbil\_score\_dtls` table where the application number matches the insured person's verification number or sign card number, the top indicator is 'Y', and the element name is 'INCOME\_SEGMENT\_SCORE\_DESCRIPTION'. If no data is found, set the `income\_segment` to 0.  
5. Finally, set the flag `v\_crif\_flag` to 'Y'.  
  
Definition of Done:  
- The system correctly retrieves and updates the CRIF score and income segment based on the specified conditions.  
- The flag `v\_crif\_flag` is correctly set to 'N' at the beginning and 'Y' at the end of the process.  
- The process handles cases where no data is found and sets appropriate default values.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Retrieve `cibil\_score`:  
 ```sql  
 SELECT NVL(element\_value, :control.cibil\_score)  
 INTO :control.cibil\_score  
 FROM azbj\_crif\_cbil\_score\_dtls  
 WHERE appln\_no = TO\_CHAR(NVL(:insured\_person.ip\_verf\_no, :insured\_person.ip\_sign\_card\_no))  
 AND top\_indicator = 'Y'  
 AND element\_name = 'CB\_SCORE\_SCORE\_VALUE'  
 AND ROWNUM = 1;  
 ```  
  
- Retrieve `cibil\_score` if no data found in the first query:  
 ```sql  
 SELECT cibil\_score  
 INTO :control.cibil\_score  
 FROM customer.azbj\_annuity\_prod\_det  
 WHERE appln\_no = TO\_CHAR(NVL(:insured\_person.ip\_verf\_no, :insured\_person.ip\_sign\_card\_no))  
 AND top\_indicator = 'Y'  
 AND product\_id = :control.cn\_product\_id;  
 ```  
  
- Retrieve `income\_segment`:  
 ```sql  
 SELECT NVL(element\_value, 0)  
 INTO :control.income\_segment  
 FROM azbj\_crif\_cbil\_score\_dtls  
 WHERE appln\_no = TO\_CHAR(NVL(:insured\_person.ip\_verf\_no, :insured\_person.ip\_sign\_card\_no))  
 AND top\_indicator = 'Y'  
 AND element\_name = 'INCOME\_SEGMENT\_SCORE\_DESCRIPTION'  
 AND ROWNUM = 1;  
 ```

# Save Insured Person Details and Associated Documents

Type: CONTROL

Title: Save Insured Person Details and Associated Documents  
  
Acceptance Criteria:  
1. When the "Save" button is pressed, the system should check if there are existing records for the given application number with specific conditions (e.g., MODULE\_FLAG = 'BBU' and top\_indicator = 'Y').  
2. If such records exist, the system should update the top\_indicator to 'N' for those records.  
3. The system should then navigate to the 'SIP\_CONTROL' section and determine the number of records.  
4. For each record in 'SIP\_CONTROL', if the SIP\_PROOF\_TYPE is not null, the system should insert a new record into the details table with the provided information.  
5. The system should handle any exceptions during the insert operation and display appropriate messages.  
6. After processing all records, the system should commit the transaction and display a success message.  
  
Definition of Done:  
- The "Save" button functionality is implemented as per the acceptance criteria.  
- The system correctly updates existing records and inserts new records based on the conditions.  
- Appropriate messages are displayed for any exceptions encountered during the process.  
- The transaction is committed successfully, and a success message is displayed to the user.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- The following SQL operations are performed:  
 - `SELECT COUNT () FROM azbj\_aml\_sip\_dtls WHERE application\_no = ... AND MODULE\_FLAG = 'BBU' AND top\_indicator = 'Y'`  
 - `UPDATE azbj\_aml\_sip\_dtls SET top\_indicator = 'N' WHERE application\_no = ... AND MODULE\_FLAG = 'BBU' AND top\_indicator = 'Y'`  
 - `INSERT INTO azbj\_aml\_sip\_dtls (application\_no, proposal\_no, top\_indicator, action\_code, proof\_type, doc\_value, doc\_expiry\_date, module\_flag, sg\_premium, sg\_tasa) VALUES (...)`

# Ensure Policies Review and Mandatory Fields Before Exiting

Type: CONTROL

Title: Ensure Policies Review and Mandatory Fields Before Exiting  
  
Acceptance Criteria:  
1. When the user attempts to exit the current view, the system should check if all policies have been reviewed.  
 - If any policy has not been reviewed, the system should display an error message: "Please select the Checkbox against all the policies!".  
2. The system should verify that the "UW Decision" field is not empty.  
 - If the "UW Decision" field is empty, the system should display an error message: "UW Decision Selection is Mandatory!" and focus on the "UW Decision" field.  
3. The system should verify that the "UW Comments" field is not empty.  
 - If the "UW Comments" field is empty, the system should display an error message: "UW Comments Selection is Mandatory!" and focus on the "UW Comments" field.  
4. Upon successful validation, the system should hide the current view and navigate to the "Previous Policy Details" section.  
  
Definition of Done:  
- All policies are reviewed, and necessary decisions and comments are provided.  
- Appropriate error messages are displayed for missing information.  
- The current view is hidden, and the user is navigated to the "Previous Policy Details" section upon successful validation.

# Implement 'Remove Split Reason' Field in New Business Module

Type: CONTROL

Title: Implement 'Remove Split Reason' Field in New Business Module  
  
Acceptance Criteria:  
1. The 'Remove Split Reason' field should be displayed on the 'AML' tab of the 'New Business' module.  
2. The field should be disabled by default.  
3. The field should be a dropdown list with a maximum length of 500 characters.  
4. The dropdown list should contain 5 predefined options.  
5. The field should have a gray background and black text.  
6. The field should be positioned at coordinates (711, 394) on the 'AML' tab.  
7. The field should have a width of 153 pixels and a height of 20 pixels.  
8. The font used for the field should be 'MS Sans Serif' with a size of 8, bold weight, and plain style.  
  
Definition of Done:  
- The 'Remove Split Reason' field is implemented as per the acceptance criteria.  
- The field is tested to ensure it is disabled by default and displays the correct dropdown options.  
- The field's appearance and position are verified to match the specified attributes.  
- The implementation is reviewed and approved by the stakeholders.

# Retention Flag Confirmation and Reason Entry

Type: CONTROL

Title: Retention Flag Confirmation and Reason Entry  
  
Acceptance Criteria:  
1. When the retention flag is changed to a value other than 'NO', a confirmation message should be displayed asking, "Do you really want to select the Reinsurance flag for this case?"  
2. If the user confirms the message:  
 - The reason field for the retention flag should become visible and enabled.  
 - A message should be displayed prompting the user to enter the reason for the retention flag.  
3. If the user does not confirm the message:  
 - The retention flag should be reset to 'NO'.  
 - The reason field for the retention flag should be hidden and disabled.  
4. If the retention flag is set to 'NO':  
 - The reason field for the retention flag should be hidden and disabled.  
  
Definition of Done:  
- The confirmation message is displayed correctly when the retention flag is changed.  
- The reason field visibility and enabled state are correctly toggled based on user confirmation.  
- The retention flag and reason field are reset correctly when the user does not confirm or sets the retention flag to 'NO'.  
- All changes are tested and verified to work as expected without any Oracle Forms-specific terminology or dependencies.

# User Confirmation for Reinsurance Selection

Type: CONTROL

Title: User Confirmation for Reinsurance Selection  
  
Acceptance Criteria:  
1. When the "Reinsurance" option is selected and it is not set to 'NO':  
 - A confirmation alert should be displayed asking, "Do you really want to select the Reinsurance flag for this case?"  
 - If the user confirms (clicks the alert button):  
 - The "Reinsurance Flag Reason" field should become visible and enabled.  
 - A message should be displayed prompting the user to enter the reason for selecting the "Reinsurance" flag.  
 - If the user cancels (does not click the alert button):  
 - The "Reinsurance Flag Reason" field should be set to NULL.  
 - The "Reinsurance" option should be reset to 'NO'.  
 - The "Reinsurance Flag Reason" field should be hidden and disabled.  
2. If the "Reinsurance" option is set to 'NO':  
 - The "Reinsurance Flag Reason" field should be set to NULL.  
 - The "Reinsurance Flag Reason" field should be hidden and disabled.  
  
Definition of Done:  
- The user is prompted with a confirmation message when selecting the "Reinsurance" option.  
- The "Reinsurance Flag Reason" field's visibility and enabled state are correctly toggled based on the user's confirmation.  
- Appropriate messages are displayed to guide the user in providing a reason for their selection.  
- The "Reinsurance" option and "Reinsurance Flag Reason" field are reset and hidden if the user cancels the selection or if the option is set to 'NO'.

# View Previous Policy Details

Type: CONTROL

Title: View Previous Policy Details  
  
Acceptance Criteria:  
1. When the "Previous Policy" button is pressed, the system should display the "Previous Policy" view.  
2. The system should allow inserting and updating records in the "Previous Details" section temporarily while fetching data.  
3. The system should fetch and display the following details for each previous policy:  
 - Serial number  
 - User ID  
 - Underwriting decision  
 - Activity date  
 - Underwriting comments  
4. The system should then disable inserting and updating records in the "Previous Details" section.  
5. The system should fetch and display the following details for each previous policy in the "Previous Policy" section:  
 - Policy reference number  
 - Customer name  
 - Policy status  
6. If the policy status is "CI" (Cancelled Inception), the system should fetch and display the request reason and user comments from the complaints table.  
7. If the policy status is "Declined", the system should fetch and display the reason and underwriting comments from the declined policies table.  
8. The system should handle any exceptions during data fetching and display appropriate error messages.  
  
Definition of Done:  
- The "Previous Policy" button should trigger the display of previous policy details.  
- The system should correctly fetch and display all required details for previous policies.  
- The system should handle exceptions gracefully and display error messages when necessary.  
- The functionality should be tested and verified to ensure it meets the acceptance criteria.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Fetch distinct part IDs from `azbj\_auto\_cp\_merge\_details` where `appln\_no` matches the insured person's verification or sign card number and `top\_indicator` is 'Y'.  
- Fetch all records from `AZBJ\_PREV\_POLICY\_TRACKER` where `application\_no` matches the insured person's verification or sign card number or `proposal\_no` matches the control policy reference, ordered by `activity\_date` in descending order.  
- Fetch distinct policy details from multiple tables (`OCP\_INTERESTED\_PARTIES`, `CP\_PARTNERS`, `OCP\_POLICY\_VERSIONS`, `OCP\_POLICY\_COVERS`, `OCP\_POLICY\_BASES`, `AZBJ\_POLICY\_CONTRACT\_EXT`, `CUSTOMER.AZBJ\_AUTO\_CP\_MERGE\_DETAILS`) where various conditions are met, including matching `contract\_id`, `partner\_id`, and other criteria.  
- Fetch the count of policy versions with specific change descriptions from `OCP\_POLICY\_VERSIONS` where `contract\_id` matches the previous policy's contract ID.  
- Fetch percentage values from `azbj\_policy\_covers\_ext` or `wip\_azbj\_policy\_covers\_ext` where `contract\_id` matches the previous policy's contract ID and certain conditions are met.  
- Fetch request reason and user comments from `AZBJ\_COMPLAINTS` where `policy\_ref` matches the previous policy's reference and `REQ\_DESC` is 'MO\_CANC\_INCEP'.  
- Fetch reason and underwriting comments from `azbj\_declined` and `azbj\_system\_constants` where `sys\_type` is 'DECLN\_TYPE', `sys\_code` matches the declined policy's reason, and `policy\_ref` matches the previous policy's reference.

# Validate and Process Previous Policy Information

Type: CONTROL

Title: Validate and Process Previous Policy Information  
  
Acceptance Criteria:  
1. When the "OK" button is pressed, the system should check if the "Underwriting Review" checkbox is selected for all policies. If any policy does not have the checkbox selected, an error message should be displayed: "Please select the Checkbox against all the policies!".  
2. The system should verify that the "Underwriting Decision" and "Underwriting Comments" fields are not empty. If either field is empty, an error message should be displayed indicating the mandatory field.  
3. The system should check if a record already exists in the `AZBJ\_PREV\_POLICY\_TRACKER` table for the given application or proposal number. If a record exists, it should update the `top\_indicator` to 'N' and insert a new record with the provided details.  
4. If no record exists, the system should insert a new record into the `AZBJ\_PREV\_POLICY\_TRACKER` table with the provided details.  
5. The system should commit the transaction and update relevant variables to indicate the successful processing of the previous policy.  
6. The view should be hidden, and the focus should move to the "Previous Policy Details" field.  
  
Definition of Done:  
- The "OK" button functionality is implemented as described.  
- Error messages are displayed correctly when validation fails.  
- Database operations (insert/update) are performed as per the acceptance criteria.  
- The transaction is committed, and relevant variables are updated.  
- The view is hidden, and the focus is moved to the appropriate field.  
  
DB queries for Table reference CRUD operations only(With Usage):  
```sql  
-- Check if a record exists  
SELECT COUNT()  
INTO v\_exist\_cnt  
FROM AZBJ\_PREV\_POLICY\_TRACKER  
WHERE application\_no = TO\_CHAR(NVL(:insured\_person.ip\_verf\_no, :insured\_person.ip\_sign\_card\_no))  
 OR proposal\_no = :control.cn\_policy\_ref;  
  
-- Update existing record  
UPDATE AZBJ\_PREV\_POLICY\_TRACKER  
SET top\_indicator = 'N'  
WHERE application\_no = TO\_CHAR(NVL(:insured\_person.ip\_verf\_no, :insured\_person.ip\_sign\_card\_no))  
 OR proposal\_no = :control.cn\_policy\_ref;  
  
-- Insert new record  
INSERT INTO AZBJ\_PREV\_POLICY\_TRACKER (application\_no, proposal\_no, USER\_ID, UW\_DECISION, UW\_COMMENTS, ACTIVITY\_DATE, TOP\_INDICATOR)  
VALUES (TO\_CHAR(NVL(:insured\_person.ip\_verf\_no, :insured\_person.ip\_sign\_card\_no)), :control.cn\_policy\_ref, USER, :CONTROL.PREV\_UW\_DECISION, :CONTROL.PREV\_UW\_COMMENTS, SYSDATE, 'Y');  
```

# Implement DOB Match List Item

Type: CONTROL

Title: Implement DOB Match List Item  
  
Acceptance Criteria:  
1. The DOB match option should be presented as a list item.  
2. The list should contain at least two options for the user to select from.  
3. The selected option should be clearly visible and distinguishable.  
4. The list item should be placed at a specific position on the user interface for easy access.  
5. The list item should have a specific width and height to ensure it fits well within the designated area.  
6. The list item should have a specific background color, font style, and font size to ensure readability and consistency with the overall design.  
  
Definition of Done:  
1. The DOB match list item is implemented and integrated into the user interface.  
2. The list item meets all the specified acceptance criteria.  
3. The functionality is tested and verified to ensure it works as expected.  
4. The user interface is reviewed and approved by the stakeholders.  
5. The feature is documented and ready for deployment.

# Initiate Video Call with OK Button

Type: CONTROL

Title: Initiate Video Call with OK Button  
  
Acceptance Criteria:  
1. When the "OK" button is pressed, the system should navigate to the video calling control item.  
2. The current view should be hidden upon pressing the "OK" button.  
  
Definition of Done:  
- The "OK" button is visible and enabled on the interface.  
- Pressing the "OK" button successfully initiates the video calling process by navigating to the appropriate control item.  
- The current view is hidden when the "OK" button is pressed.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- The LOV (List of Values) for document types should be populated using the following query:  
 ```sql  
 SELECT DOC\_TYPE, DOC\_DESC  
 FROM azbj\_dms\_doc\_type\_master  
 ORDER BY SR\_NO;  
 ```

# Video Calling Status Retrieval

Type: CONTROL

Title: Video Calling Status Retrieval  
  
Acceptance Criteria:  
1. When the "Video Calling Status" button is pressed, the system should retrieve the approval status, approval substatus, and remarks from the `azbj\_vdo\_cv\_approved\_cases` table based on the customer's application number.  
2. If the video calling data is not available, the system should display a warning message: "Video calling data not available."  
3. The system should display the "PIVC Status" view and navigate to the "Video Calling OK" control item.  
4. If there is an error during the process, the system should display a warning message with the error details.  
  
Definition of Done:  
- The "Video Calling Status" button should be functional and trigger the retrieval of video calling data.  
- Appropriate messages should be displayed based on the availability of data and any errors encountered.  
- The "PIVC Status" view should be displayed, and the system should navigate to the "Video Calling OK" control item.  
  
DB queries for Table reference CRUD operations only(With Usage):  
```sql  
SELECT approval\_status, approval\_substatus, remarks  
FROM CUSTOMER.azbj\_vdo\_cv\_approved\_cases b  
WHERE b.application\_no = TO\_CHAR(NVL(:insured\_person.ip\_sign\_card\_no, :insured\_person.ip\_verf\_no));  
```  
  
This query is used to fetch the approval status, substatus, and remarks for the video calling session based on the customer's application number.

# Implement Name Match Dropdown

Type: CONTROL

Title: Implement Name Match Dropdown  
  
Acceptance Criteria:  
1. The "Name Match" field should display a dropdown list with at least two options.  
2. The dropdown list should be easily accessible and visible on the screen.  
3. The selected name should be highlighted and stored for further actions.  
  
Definition of Done:  
1. The "Name Match" field is implemented and displays a dropdown list with the required options.  
2. The dropdown list is functional and allows users to select an option.  
3. The selected option is stored and can be used in subsequent processes.  
4. The user interface is consistent with the overall design and user experience guidelines.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include any specific database queries.

# Save Financial Underwriting Details

Type: CONTROL

Title: Save Financial Underwriting Details  
  
Acceptance Criteria:  
1. If the premium payer is not specified, the system should default it to 1 and display an error message indicating that regular income should not be null.  
2. Depending on the premium payer type (1, 2, or 3), the system should calculate and store the net income, gross income, and total investment for the insured person, policyholder, or spouse.  
3. If a delete flag is set, the system should reset the flag and delete the corresponding financial underwriting record.  
4. The system should iterate through the financial underwriting records and update or insert them based on the contract ID, premium payer, and year.  
5. If the gross income or total income is below a certain threshold, the system should display a warning message.  
6. The system should update or insert liquid investment details based on the contract ID and premium payer.  
7. The system should update or insert spouse financial details based on the contract ID and premium payer.  
8. The system should delete and reinsert beneficial ownership details based on the contract ID.  
9. The system should update or insert policy risk details based on the contract ID and policy number.  
10. The system should update the total sum assured for the insured person and policyholder based on the calculated values.  
  
Definition of Done:  
- The financial underwriting details are saved successfully.  
- The system updates or inserts the relevant records in the database.  
- Appropriate error and warning messages are displayed as per the acceptance criteria.  
- The total sum assured for the insured person and policyholder is updated correctly.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- DELETE from azbj\_financial\_uw WHERE CONTRACT\_ID = :control.cn\_contract\_id AND IP\_NO = :control.PREMIUM\_PAYER;  
- UPDATE azbj\_financial\_uw SET POLICY\_REF = :control.cn\_policy\_ref, ITRS = :financial\_uw.itrs, COMPUTATION = :financial\_uw.computation, GROSS\_INCOME = :financial\_uw.gross\_income, NET\_PROFIT = :financial\_uw.net\_profit, EXEMPTED\_INCOME = :financial\_uw.exempted\_income, ONETIME\_INCOME = :financial\_uw.one\_time\_income, TOTAL\_INCOME = :financial\_uw.total\_INCOME, DEDUCTION = :financial\_uw.deduction, TAX = :financial\_uw.tax, PROOF\_TYPE = :financial\_uw.PROOF\_TYPE WHERE CONTRACT\_ID = :control.cn\_contract\_id AND IP\_NO = :control.PREMIUM\_PAYER AND YRS = :financial\_uw.yrs;  
- INSERT INTO azbj\_financial\_uw (POLICY\_REF, CONTRACT\_ID, IP\_NO, YRS, ITRS, COMPUTATION, GROSS\_INCOME, NET\_PROFIT, EXEMPTED\_INCOME, ONETIME\_INCOME, TOTAL\_INCOME, DEDUCTION, TAX, PROOF\_TYPE) VALUES (:control.cn\_policy\_ref, :control.cn\_contract\_id, :control.PREMIUM\_PAYER, :financial\_uw.yrs, :financial\_uw.itrs, :financial\_uw.computation, :financial\_uw.gross\_income, :financial\_uw.net\_profit, :financial\_uw.exempted\_income, :financial\_uw.one\_time\_income, :financial\_uw.total\_INCOME, :financial\_uw.deduction, :financial\_uw.tax, :financial\_uw.PROOF\_TYPE);  
- UPDATE azbj\_liquid\_investment\_dtls SET FIXED\_TERM\_DEPOSIT = :LIQUID\_INVESTMENT.FIXED\_TERM\_DEP, MUTUAL\_FUND = :LIQUID\_INVESTMENT.MUTUAL\_FUND, EQUITY\_SHARE = :LIQUID\_INVESTMENT.EQUITY\_SHARE, FUND\_VAL\_UL\_POL = :LIQUID\_INVESTMENT.FUND\_VAL\_UL\_POL, BANK\_BALANCE = :LIQUID\_INVESTMENT.BANK\_BALANCE, ONE\_TIME\_INCOME = :LIQUID\_INVESTMENT.ONE\_TIME\_INCOME, LIQUID\_INVESTMENT = :LIQUID\_INVESTMENT.LIQUID\_INV, TOTAL\_INVESTMENT = :LIQUID\_INVESTMENT.TOTAL\_INV WHERE CONTRACT\_ID = :control.cn\_contract\_id AND IP\_NO = :control.PREMIUM\_PAYER;  
- INSERT INTO azbj\_liquid\_investment\_dtls (Contract\_id, APPLICATION\_NO, PROPOSAL\_NO, FIXED\_TERM\_DEPOSIT, MUTUAL\_FUND, EQUITY\_SHARE, FUND\_VAL\_UL\_POL, BANK\_BALANCE, ONE\_TIME\_INCOME, LIQUID\_INVESTMENT, TOTAL\_INVESTMENT, IP\_NO, create\_user, create\_date) VALUES (:control.cn\_contract\_id, TO\_CHAR(NVL(:insured\_person.ip\_verf\_no, :insured\_person.ip\_sign\_card\_no)), :control.cn\_policy\_ref, :LIQUID\_INVESTMENT.FIXED\_TERM\_DEP, :LIQUID\_INVESTMENT.MUTUAL\_FUND, :LIQUID\_INVESTMENT.EQUITY\_SHARE, :LIQUID\_INVESTMENT.FUND\_VAL\_UL\_POL, :LIQUID\_INVESTMENT.BANK\_BALANCE, :LIQUID\_INVESTMENT.ONE\_TIME\_INCOME, :LIQUID\_INVESTMENT.LIQUID\_INV, :LIQUID\_INVESTMENT.TOTAL\_INV, :control.PREMIUM\_PAYER, USER, SYSDATE);  
- UPDATE AZBJ\_SPOUSE\_FIN\_UW\_DETAILS SET SP\_INC\_PROOF\_TYPE = :SPOUSE\_FIN\_DTLS.PROOF\_TYPE, SP\_GROSS\_INCOME = :SPOUSE\_FIN\_DTLS.GROSS\_INCOME, SP\_EXEMPTED\_INCOME = :SPOUSE\_FIN\_DTLS.EXEMPTED\_INCOME, SP\_ONETIME\_INCOME = :SPOUSE\_FIN\_DTLS.ONE\_TIME\_INCOME, SP\_TOTAL\_INCOME = :SPOUSE\_FIN\_DTLS.TOTAL\_INCOME, SP\_DEDUCTION = :SPOUSE\_FIN\_DTLS.DEDUCTION, SP\_TAX = :SPOUSE\_FIN\_DTLS.TAX, SP\_NET\_INCOME = :SPOUSE\_FIN\_DTLS.NET\_INCOME, SP\_PL\_PROOF\_TYPE = :SPOUSE\_FIN\_DTLS.PROOF\_TYPE\_S, SP\_GROSS\_PL = :SPOUSE\_FIN\_DTLS.GROSS\_INCOME\_S, SP\_EXEMPTED\_PL = :SPOUSE\_FIN\_DTLS.EXEMPTED\_INCOME\_S, SP\_ONETIME\_PL = :SPOUSE\_FIN\_DTLS.ONE\_TIME\_INCOME\_S, SP\_TOTAL\_PL = :SPOUSE\_FIN\_DTLS.TOTAL\_INCOME\_S, SP\_DEDUCTION\_PL = :SPOUSE\_FIN\_DTLS.DEDUCTION\_S, SP\_TAX\_PL = :SPOUSE\_FIN\_DTLS.TAX\_S, SP\_NET\_PL = :SPOUSE\_FIN\_DTLS.NET\_INCOME\_S WHERE Contract\_ID = :control.cn\_contract\_id AND IP\_NO = :control.PREMIUM\_PAYER;  
- INSERT INTO AZBJ\_SPOUSE\_FIN\_UW\_DETAILS (Contract\_ID, APPLICATION\_NO, PROPOSAL\_NO, SP\_INC\_PROOF\_TYPE, SP\_GROSS\_INCOME, SP\_EXEMPTED\_INCOME, SP\_ONETIME\_INCOME, SP\_TOTAL\_INCOME, SP\_DEDUCTION, SP\_TAX, SP\_NET\_INCOME, SP\_PL\_PROOF\_TYPE, SP\_GROSS\_PL, SP\_EXEMPTED\_PL, SP\_ONETIME\_PL, SP\_TOTAL\_PL, SP\_DEDUCTION\_PL, SP\_TAX\_PL, SP\_NET\_PL, IP\_NO) VALUES (:control.cn\_contract\_id, TO\_CHAR(NVL(:insured\_person.ip\_verf\_no, :insured\_person.ip\_sign\_card\_no)), :control.cn\_policy\_ref, :SPOUSE\_FIN\_DTLS.PROOF\_TYPE, :SPOUSE\_FIN\_DTLS.GROSS\_INCOME, :SPOUSE\_FIN\_DTLS.EXEMPTED\_INCOME, :SPOUSE\_FIN\_DTLS.ONE\_TIME\_INCOME, :SPOUSE\_FIN\_DTLS.TOTAL\_INCOME, :SPOUSE\_FIN\_DTLS.DEDUCTION, :SPOUSE\_FIN\_DTLS.TAX, :SPOUSE\_FIN\_DTLS.NET\_INCOME, :SPOUSE\_FIN\_DTLS.PROOF\_TYPE\_S, :SPOUSE\_FIN\_DTLS.GROSS\_INCOME\_S, :SPOUSE\_FIN\_DTLS.EXEMPTED\_INCOME\_S, :SPOUSE\_FIN\_DTLS.ONE\_TIME\_INCOME\_S, :SPOUSE\_FIN\_DTLS.TOTAL\_INCOME\_S, :SPOUSE\_FIN\_DTLS.DEDUCTION\_S, :SPOUSE\_FIN\_DTLS.TAX\_S, :SPOUSE\_FIN\_DTLS.NET\_INCOME\_S, :control.PREMIUM\_PAYER);  
- DELETE AZBJ\_BENEFICIAL\_OWNERSHIP\_FW WHERE contract\_id = :control.cn\_contract\_id;  
- INSERT INTO AZBJ\_BENEFICIAL\_OWNERSHIP\_FW (contract\_id, BENEFICIAL\_OWNER, BEB\_OWNER\_SHARE) VALUES (:control.cn\_contract\_id, :CFT\_FIN.BEN\_OWNER\_FIN, :CFT\_FIN.SHARES\_FIN);  
- UPDATE WIP\_AZBJ\_POLICY\_RISK\_REP SET LIFE\_COVERAGE\_SUM = :RISK\_DECLARE.RS\_LIFE\_COVERAGE\_SUM, YR\_OF\_ISSUE = :RISK\_DECLARE.RS\_YR\_OF\_ISSUE, IP\_TYPE = NVL(:RISK\_DECLARE.RS\_IP\_TYPE, 1), Premium = NVL(:RISK\_DECLARE.RS\_PREMIUM, 0) WHERE contract\_id = :control.cn\_contract\_id AND POLICY\_NO = :RISK\_DECLARE.RS\_POLICY\_NO;  
- INSERT INTO WIP\_AZBJ\_POLICY\_RISK\_REP (CONTRACT\_ID, POLICY\_NO, ACTION\_CODE, LIFE\_COVERAGE\_SUM, YR\_OF\_ISSUE, IP\_TYPE, Premium) VALUES (:control.cn\_contract\_id, :RISK\_DECLARE.RS\_POLICY\_NO, 'A', :RISK\_DECLARE.RS\_LIFE\_COVERAGE\_SUM, :RISK\_DECLARE.RS\_YR\_OF\_ISSUE, NVL(:RISK\_DECLARE.RS\_IP\_TYPE, 1), :RISK\_DECLARE.RS\_PREMIUM);

# Manage Boiler Insurance Policy Information

Type: SOFA\_INFO\_BOILER

Detailed description: As a user, I want to manage boiler insurance policy information, including policy number, agent number, policy holder, policy status, and effective date, so that I can efficiently track and update policy details.  
  
Acceptance criteria:  
1. The system should allow the user to input and display the following fields:  
 - Policy Number  
 - Agent Number  
 - Policy Holder  
 - Policy Status  
 - Effective Date  
2. The system should initialize each field with default values corresponding to their respective data types.  
3. The system should display the fields in a vertical orientation without scrollbars.  
4. The system should ensure that the effective date is a valid date format.  
  
Definition of Done:  
1. The user can successfully input and save the policy number, agent number, policy holder, policy status, and effective date.  
2. The fields are displayed correctly without any horizontal or vertical scrollbars.  
3. The effective date is validated to ensure it is in the correct date format.  
4. The user interface is intuitive and easy to navigate.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Not applicable as the provided XML content does not include specific database queries or table references.

# View and interact with New Business module sections

Type: BOILER

Detailed description: As a user, I want to view and interact with various sections of the "New Business" module, including user information, effective dates, and different tabs for insured persons, policy holders, premium payers, covers, agents, and other related entities. This will help me manage and review new business entries efficiently.  
  
Acceptance criteria:  
1. The user should be able to see the following sections:  
 - User ID  
 - Effective Date  
 - Reason Code  
 - Abort Movement  
2. The user should be able to navigate through different tabs, each representing a specific category such as:  
 - Insured Person  
 - Policy Holder  
 - Premium Payer/Beneficiaries  
 - Covers  
 - Agent  
3. The user should be able to see the toolbar with user information and the date.  
4. The user should not be able to insert, update, or delete records in this section.  
5. The user should not see any scrollbars in the sections.  
  
Definition of Done:  
- The user interface displays all the specified sections and tabs.  
- The user can navigate through the tabs without any issues.  
- The toolbar displays the user information and date correctly.  
- The sections are read-only, and no modifications can be made by the user.  
- The interface is free of scrollbars as specified.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the block is not linked to any database operations.

# Implement Tabbed Interface for Structured Information Display

Type: TABS

Title: Implement Tabbed Interface for Structured Information Display  
  
Acceptance Criteria:  
1. The tabbed interface should display a horizontal and vertical layout with distinct sections.  
2. Each tab should be clearly labeled and should not be editable.  
3. The interface should include visual elements such as lines and blocks to separate different sections.  
4. The tabbed interface should be visually appealing with appropriate colors and fonts.  
5. The interface should not allow any data manipulation (insert, update, delete) directly within the tabs.  
6. The tabbed interface should be displayed within a window titled "New Business" with specified dimensions.  
  
Definition of Done:  
1. The tabbed interface is implemented and displays correctly within the application.  
2. All tabs and visual elements are non-editable and properly styled.  
3. The interface meets the specified visual and functional requirements.  
4. The window containing the tabbed interface is titled "New Business" and matches the specified dimensions.  
5. The implementation is tested and verified to ensure it meets the acceptance criteria.

# Display Validation Error List

Type: AZBJ\_VALIDATION\_LIST

Title: Display Validation Error List  
  
Acceptance Criteria:  
1. The list should display the following columns:  
 - Block Name  
 - Item Name  
 - Error Code  
 - Error Message  
 - Error Type  
 - Error Section  
2. Each column should have a corresponding prompt or label for clarity.  
3. The list should be scrollable if the number of records exceeds the display limit.  
4. There should be an "Ok" button to close the error list window.  
  
Definition of Done:  
1. The error list displays all required columns with appropriate labels.  
2. The list is scrollable when there are more records than can be displayed at once.  
3. The "Ok" button is functional and closes the error list window.  
4. The error list window is modal, preventing interaction with other windows until it is closed.  
5. The error list window has a fixed size and cannot be resized or maximized.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include any specific database queries.

# Navigate to Validation List Item on Double-Click

Type: AZBJ\_VALIDATION\_LIST

Title: Navigate to Validation List Item on Double-Click  
  
Acceptance Criteria:  
1. When a user double-clicks on an error section, the system should navigate to the corresponding item in the validation list.  
2. The navigation should be based on the block name and item name associated with the error section.  
  
Definition of Done:  
1. The user can double-click on any error section in the error list.  
2. Upon double-clicking, the system successfully navigates to the corresponding item in the validation list.  
3. The navigation logic 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 database-specific queries.

# Navigate to Validation List Item on Double-Click

Type: AZBJ\_VALIDATION\_LIST

Title: Navigate to Validation List Item on Double-Click  
  
Acceptance Criteria:  
1. When a user double-clicks on the error type field, the system should navigate to the corresponding item in the validation list.  
2. The navigation should be seamless and should not require any additional user input.  
  
Definition of Done:  
1. The functionality to navigate to the corresponding item in the validation list upon double-clicking the error type field is implemented.  
2. The feature is tested and verified to ensure it works as expected.  
3. The feature is documented for future reference and user training.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include any database-specific queries.

# Navigate to Validation List Item on Double-Clicking Error Code

Type: AZBJ\_VALIDATION\_LIST

Title: Navigate to Validation List Item on Double-Clicking Error Code  
  
Acceptance Criteria:  
1. When a user double-clicks on an error code in the error list, the system should navigate to the corresponding item in the validation list.  
2. The navigation should be seamless and should not require any additional user input.  
  
Definition of Done:  
1. The functionality to navigate to the corresponding item in the validation list upon double-clicking an error code is implemented.  
2. The feature is tested and verified to ensure that it works as expected.  
3. The user documentation is updated to reflect this new functionality.  
4. The code is reviewed and approved by the team.  
  
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 Ok Button Functionalities and Validations

Type: AZBJ\_VALIDATION\_LIST

Title: Implement Ok Button Functionalities and Validations  
  
Acceptance Criteria:  
1. When the "Ok" button is pressed, the system should:  
 - Enable the "Commit Form" button on the horizontal toolbar if it is currently disabled.  
 - Set a global flag to 'N'.  
 - Update a control variable to 'Z'.  
 - Find and store the ID of a specific tab page.  
 - If a global error status is 'Y', disable certain form elements and enable others.  
 - Navigate to the 'covers' block and move to the first record.  
 - If a specific form question number is not null, disable a particular form element.  
 - If certain control variables meet specific conditions, disable another form element.  
 - Set a coverhead variable to 'N'.  
 - If a control product ID is within a specified range and a global error status is not 'Y', navigate through the 'covers' block records and perform specific checks and updates based on cover codes and entry ages.  
 - If a specific function returns 'Y' or a control product ID is within another specified range, reset certain coverhead variables and navigate through the 'covers' block records again, performing similar checks and updates.  
 - Find and store the ID of another tab page.  
 - If certain coverhead variables are 'Y', make the tab page visible and enable specific form elements.  
 - If a control product ID is a specific value, enable certain form elements.  
 - Otherwise, hide the tab page and disable specific form elements.  
 - If certain coverhead variables are 'Y' and a minor flag is 'Y', make the tab page visible.  
 - If a global error status is not 'Y', call a specific procedure.  
 - Select and store specific values from a database table into variables.  
 - Update certain agent variables with the stored values.  
 - Navigate to the 'AML' block and move to the first record.  
 - If a specific AML document description is null, update a control variable and execute a trigger.  
 - If certain agent variables meet specific conditions, navigate to a specific item.  
 - Otherwise, navigate to another item.  
 - If a control product ID is within a specified range and a global error status is not 'Y', call a specific procedure.  
  
Definition of Done:  
- The "Ok" button triggers all specified actions and updates correctly.  
- All conditions and validations are met as described.  
- The system navigates through records and updates form elements as required.  
- All specified procedures are called and executed without errors.  
- The form behaves as expected based on the current state and input values.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Not applicable as the provided XML content does not include direct CRUD operations on database tables.

# View and Manage Underwriting Limits

Type: BLK\_UW\_LIMITS

Title: View and Manage Underwriting Limits  
  
Acceptance Criteria:  
1. The system should display the following fields:  
 - User ID  
 - User Name  
 - Underwriting Code  
 - Underwriting Description  
 - Minimum SUC  
 - Maximum SUC  
 - Minimum Age  
 - Maximum Age  
 - Minimum Code Value  
 - Maximum Code Value  
 - Start Date  
 - Expiry Date  
2. Each field should be displayed with appropriate labels and alignments.  
3. The fields should be read-only and not editable by the user.  
4. The system should provide an "Exit" button to close the view.  
  
Definition of Done:  
1. The user interface displays all the specified fields with correct labels and alignments.  
2. The fields are read-only and cannot be edited by the user.  
3. The "Exit" button is functional and closes the view when clicked.  
4. The implementation is tested and verified to meet the acceptance criteria.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- No specific database queries are provided in the XML content.

# Exit Button Functionality and Navigation to Agents Section

Type: BLK\_UW\_LIMITS

Detailed description: As a user, I want to be able to exit the current form and navigate to the 'Agents' section, ensuring that all necessary conditions and validations are met before the transition.  
  
Acceptance criteria:  
1. When the exit button is pressed, the current window and canvas should be hidden.  
2. The system should navigate to the 'Agents' section and focus on the 'Agents' button.  
3. If the 'HORIZONTAL\_TOOLBAR.COMMIT\_FORM' item is disabled, it should be enabled.  
4. The global flag should be set to 'N'.  
5. The 'control.v\_uw\_rcvd' should be set to 'Z'.  
6. If there is an error status, certain items should be disabled, otherwise, they should be enabled.  
7. The system should navigate to the 'covers' section and process records to check for specific conditions related to cover codes and entry ages.  
8. If certain conditions are met, specific items should be enabled or disabled accordingly.  
9. If the insured person's or policy holder's contact telephone number is null, it should be set to 'Not Available'.  
10. The system should navigate to the 'AML' section and process records.  
11. If the 'agents.subcode\_enabled\_flag' is 'Y' and the 'agents.ag\_sub\_code' is null, the system should focus on the 'AGENTS.ag\_SUB\_CODE' item.  
12. If certain product IDs are present and there is no error status, the system should check for riders.  
13. If the user flag is 'Y' and the 'HORIZONTAL\_TOOLBAR.COMMIT\_FORM' item is enabled, it should be disabled.  
  
Definition of Done:  
- The exit button functionality is implemented and tested.  
- All conditions and validations are met as per the acceptance criteria.  
- The system navigates correctly to the 'Agents' section and focuses on the appropriate item.  
- All necessary items are enabled or disabled based on the conditions.  
- The insured person's and policy holder's contact telephone numbers are set correctly if null.  
- The system processes records in the 'covers' and 'AML' sections as required.  
- The functionality is verified through unit tests and user acceptance tests.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Not applicable as the provided XML content does not include specific CRUD operations that can be directly executed in the database.

# Display Policy Information

Type: SOFA\_INFO\_DATA

Title: Display Policy Information  
  
Acceptance Criteria:  
1. The policy information should be displayed in a structured format with the following fields:  
 - Policy Number  
 - Product ID  
 - Agent Number  
 - Contract ID  
 - Policyholder  
 - Branch  
 - Receipt Number  
 - Policy Status  
 - Effective Date  
 - Amount  
2. The fields should be pre-populated with data from the respective sources:  
 - Policy Number from CONTROL.CN\_POLICY\_REF  
 - Product ID from CONTROL.CN\_PRODUCT\_ID  
 - Agent Number from AGENTS.AG\_AGENT\_CODE  
 - Receipt Number from CONTROL.CN\_PERMRCPT\_NO  
 - Effective Date from CONTROL.CN\_EFFECTIVE\_DATE  
3. All fields should be read-only and not editable by the user.  
4. The information should be displayed in a visually appealing manner with appropriate font styles and colors.  
  
Definition of Done:  
- The policy information is displayed correctly with all the required fields.  
- The data is accurately fetched from the respective sources and displayed in the fields.  
- The fields are read-only and cannot be edited by the user.  
- The display format is visually appealing and consistent with the design specifications.  
  
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.

# Manage Questionnaire Form

Type: AZBJ\_BBU\_QUESTIONS\_PH

Title: Manage Questionnaire Form  
  
Acceptance Criteria:  
1. The form should display a list of questions with the following details:  
 - Question Description  
 - Answer  
 - Version Date  
 - Version Number  
 - End Date  
 - Answer Type  
 - Validation  
 - Process  
 - Sub Question Number  
 - Question ID  
 - Form Question Number  
 - Details if Marked Yes  
  
2. The form should allow navigation between records using keyboard keys:  
 - Pressing the down arrow key should move to the next record and focus on the answer field.  
 - Pressing the up arrow key should move to the previous record and focus on the answer field, unless it is the first record.  
  
3. The form should handle new record instances by checking if the form question number and question description are null. If both are null and it is not the first record, it should navigate to the previous record.  
  
Definition of Done:  
- The form displays all required fields for each question.  
- Users can navigate between records using the keyboard.  
- The form correctly handles new record instances as specified.  
- All functionalities are tested and verified to work as expected.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- No specific database queries are provided in the XML content.

# Validate and Process Questionnaire Answers

Type: AZBJ\_BBU\_QUESTIONS\_PH

Title: Validate and Process Questionnaire Answers  
  
Acceptance Criteria:  
1. When the answer to a question is 'Y', the system should check if the question already exists in the group. If it does not, it should add the question and its answer to the group.  
2. When the answer to a question is 'N', the system should update the existing answer in the group if the question exists.  
3. The system should delete any group rows where the answer is 'N'.  
4. The system should validate that answers for certain question types are in the correct format:  
 - For 'BY' and 'BN' types, the answer must be 'Y' or 'N'.  
 - For 'D' type, the answer must be a valid date in the format 'DD-MM-YYYY'.  
 - For 'NN' and 'NY' types, the answer must be a valid number.  
5. The system should handle specific conditions for pregnancy, alcohol, and tobacco-related questions:  
 - For pregnancy-related questions, the system should set a global status based on the answer and validate the month of pregnancy if applicable.  
 - For alcohol-related questions, the system should set a global status and validate the details of drinking habits if applicable.  
 - For tobacco-related questions, the system should set a global status based on the answer.  
6. The system should navigate to the next or previous record based on the conditions met during the validation process.  
  
Definition of Done:  
- The system correctly validates and processes answers according to the specified rules.  
- The system updates the group with the correct answers and deletes rows where necessary.  
- The system ensures that answers are in the correct format for specific question types.  
- The system handles special conditions for pregnancy, alcohol, and tobacco-related questions.  
- The system navigates to the appropriate record based on the validation results.  
- All acceptance criteria are met, and the functionality is tested and verified.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Not applicable as the provided XML content does not include direct database queries that can be executed without modification.

# Manage and View Questionnaire Details

Type: AZBJ\_BBU\_QUESTIONS

Detailed description: As a user, I want to manage and view questionnaire details, including partner information, question descriptions, and answers, so that I can efficiently handle questionnaire data.  
  
Acceptance criteria:  
1. The system should display a list of questions with the following details:  
 - Partner ID (read-only)  
 - Partner Name (read-only)  
 - Question Description (read-only)  
 - Answer (editable)  
 - Details if Marked Yes (editable)  
 - Version (read-only)  
 - Version Date (editable)  
 - End Date (editable)  
 - Answer Type (editable)  
 - Validation (editable)  
 - Process (editable)  
 - Sub Question (read-only)  
 - Question ID (read-only)  
 - Form Question Number (read-only)  
 - Displayed (read-only)  
2. The system should allow navigation between records using keyboard keys.  
3. The system should ensure that if both the form question number and question description are null, the user is navigated to the previous record unless it is the first record.  
  
Definition of Done:  
- The user can view and manage questionnaire details as specified.  
- The system correctly handles navigation between records.  
- The system enforces the condition to navigate to the previous record if both the form question number and question description are null, except for the first record.  
- All fields are displayed and editable as per the acceptance criteria.  
- 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 specific database queries.

# Validate Description Input Based on Answer and Question Conditions

Type: AZBJ\_BBU\_QUESTIONS

Title: Validate Description Input Based on Answer and Question Conditions  
  
Acceptance Criteria:  
1. If the answer to the question is not 'Y' and a description is entered, the system should display an error message stating, "Answer to the question is N. Hence cannot enter description."  
2. The validation should only apply if the question ID is not 85 and the sub-question is either 68 or 69.  
  
Definition of Done:  
- The system should validate the description input based on the specified conditions.  
- An error message should be displayed if the conditions are not met.  
- The validation logic should be thoroughly tested to ensure it works as expected.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- No specific DB queries provided in the XML content.

# Validate and Process Questionnaire Answers

Type: AZBJ\_BBU\_QUESTIONS

Title: Validate and Process Questionnaire Answers  
  
Acceptance Criteria:  
1. The system should validate the answer for each question based on its type:  
 - If the answer type is 'BY' or 'BN', the answer must be 'Y' or 'N'.  
 - If the answer type is 'D', the answer must be a valid date in the format 'DD-MM-YYYY'.  
 - If the answer type is 'NN' or 'NY', the answer must be a valid number.  
2. The system should handle specific conditions for certain questions:  
 - For questions related to pregnancy (e.g., question IDs 53, 59), the system should set a global pregnancy status based on the answer.  
 - For questions related to alcohol consumption (e.g., question ID 47), the system should set a global alcohol status based on the answer.  
 - For questions related to tobacco use (e.g., question ID 46), the system should set a global tobacco status based on the answer.  
3. The system should ensure that certain answers are not null for specific questions:  
 - For question ID 59, sub-question 5, the answer cannot be null.  
 - For question ID 6, sub-question 5, the answer cannot be null.  
4. The system should delete rows from the internal group if the answer is 'N'.  
5. The system should display appropriate error messages if the validation fails:  
 - "Answer should be in Y or N" for invalid 'BY' or 'BN' answers.  
 - "Please enter valid date in the format DD-MM-YY" for invalid 'D' answers.  
 - "Please enter valid number" for invalid 'NN' or 'NY' answers.  
 - "Husband Insurance cannot be null" for specific sub-questions.  
 - "Please check pregnancy status" for inconsistent pregnancy-related answers.  
 - "She is not pregnant... Invalid Entry" for invalid pregnancy-related answers.  
 - "The Pregnancy months can be between 1 and 12" for invalid pregnancy month answers.  
 - "Please enter the month of pregnancy" for null pregnancy month answers.  
 - "Please check Alcoholism Question" for inconsistent alcohol-related answers.  
 - "The person does not drink alcohol... Invalid answer" for invalid alcohol-related answers.  
 - "Please give in details of the drinking habits" for null alcohol-related answers.  
  
Definition of Done:  
- The system correctly validates and processes the answers based on the specified criteria.  
- Appropriate error messages are displayed for invalid answers.  
- The system updates the internal group and global statuses as required.  
- The functionality is tested and verified to ensure it meets the acceptance criteria.

# Close Button Functionality and Navigation

Type: AZBJ\_BBU\_QUESTIONS

Title: Close Button Functionality and Navigation  
  
Acceptance Criteria:  
1. When the "Close" button is pressed, the system should check the status of the control variable `cn\_bbu\_status`.  
2. If `cn\_bbu\_status` is 'T', the system should navigate to the 'BBPROCESS' block.  
3. If `cn\_bbu\_status` is not 'T', the system should navigate to the 'MED\_UW' block.  
4. The system should then navigate to the 'med\_uw' block and check if the `mu\_testno` field is null.  
5. If `mu\_testno` is null, the system should navigate to the 'azbj\_bbu\_questions.ans1' item.  
6. The system should handle any exceptions that occur during this process.  
  
Definition of Done:  
- The "Close" button functionality is implemented and tested.  
- The system correctly navigates to the appropriate block based on the control variable status.  
- The system ensures that specific questions are answered before proceeding.  
- Exception handling is in place and tested.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- No specific DB queries are mentioned in the provided XML content that are directly executable without modification.

# Automatic Navigation to Previous Record if Question Number is Empty

Type: DESCRIPTION

Title: Automatic Navigation to Previous Record if Question Number is Empty  
  
Acceptance Criteria:  
1. When navigating to a new record in the "Description" section, the system should check if the "Question Number" field is empty.  
2. If the "Question Number" field is empty, the system should automatically navigate to the previous record.  
  
Definition of Done:  
1. The system correctly identifies when the "Question Number" field is empty upon navigating to a new record.  
2. The system successfully navigates to the previous record if the "Question Number" field is found to be empty.  
3. The functionality is tested and verified to ensure it works 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 any specific database queries.

# User can enter a description in the designated text field

Type: DESCRIPTION

Title: User can enter a description in the designated text field  
  
Acceptance Criteria:  
1. The description field should be enabled and allow text input up to 1000 characters.  
2. The field should be positioned at the specified coordinates and have the specified dimensions.  
3. The field should have a white background and black text, using the "MS Sans Serif" font with a size of 8 points.  
4. The field should display a prompt "Details if Marked Yes" above it, aligned to the top with a bold font style.  
5. When the user double-clicks on the field, no action should be triggered.  
6. When the user presses the down arrow key while in the field, the cursor should remain in the description field.  
  
Definition of Done:  
- The description field is implemented with the specified properties and behaviors.  
- The field is tested to ensure it meets all acceptance criteria.  
- The field 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 database-specific queries.

# Populate and Manage Questionnaire Data for Policy Members and Policy Holders

Type: BBPROCESS

Detailed description: As a user, I want to populate and manage questionnaire data for policy members and policy holders, ensuring that all necessary questions are answered and validated before proceeding with policy issuance.  
  
Acceptance criteria:  
1. The system should allow users to populate questions for policy members and policy holders.  
2. The system should validate that all required questions are answered for each policy member and policy holder.  
3. If any questions are not answered, the system should display an error message indicating which member or holder has incomplete questions.  
4. The system should check for specific conditions such as the presence of certain flags or statuses and handle them accordingly.  
5. The system should enable or disable form elements based on the validation results and user actions.  
6. The system should handle specific business rules related to policy issuance, such as checking for incomplete applications or special conditions for certain branches or products.  
  
Definition of Done:  
1. The user can successfully populate and manage questionnaire data for all policy members and policy holders.  
2. All required questions are validated, and appropriate error messages are displayed for incomplete entries.  
3. The system correctly handles specific business rules and conditions, enabling or disabling form elements as needed.  
4. The system ensures that all necessary validations and checks are performed before allowing the user to proceed with policy issuance.  
5. The functionality is tested and verified to work as expected without any Oracle Forms-specific terminology or dependencies.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- The system should execute the following query to check for previous lapse policies:  
 ```sql  
 SELECT COUNT () INTO v\_cnt2  
 FROM balic\_history\_details  
 WHERE appl\_no = TO\_CHAR (NVL (:insured\_person.ip\_sign\_card\_no, :insured\_person.ip\_verf\_no))  
 AND UPPER(tele\_ver\_status) = 'COMPLETE'  
 AND updated\_flag = 'Y';  
 ```  
- The system should execute the following query to check for incomplete applications:  
 ```sql  
 SELECT COUNT()  
 INTO v\_incomplete\_flg  
 FROM azbj\_phub\_add\_req\_tracker a, azbj\_phub\_req\_tracker b  
 WHERE a.APPLICATION\_NO = b.APPLICATION\_NO  
 AND a.REQ\_TYPE = b.REQ\_TYPE  
 AND a.application\_no = TO\_CHAR(NVL(:INSURED\_PERSON.IP\_VERF\_NO, :INSURED\_PERSON.IP\_SIGN\_CARD\_NO))  
 AND b.RECD\_IN\_OPUS = 'N';  
 ```

# Populate Questions Based on Selected Partner ID

Type: BBPROCESS

Detailed description: As a user, I want to populate questions based on the selected partner ID when I press the "Populate Questions" button, so that I can ensure the correct questions are displayed for the selected partner.  
  
Acceptance criteria:  
1. If no partner is selected, an error message "Please select partner" should be displayed.  
2. If a partner is selected, the system should check if the selected partner ID matches any existing partner IDs in the "Policy Member" records.  
3. If a match is found, the system should set a flag to indicate a match and exit the loop.  
4. If no match is found, the system should check if the selected partner ID matches the insured person or policy holder IDs.  
5. Based on the match, the system should call the `populate\_questions` function with appropriate parameters:  
 - If the selected partner ID matches the insured person ID, call `populate\_questions` with the insured person's sex, partner ID, a constant value of 1, and the insured person's name.  
 - If the selected partner ID matches the policy holder ID, call `populate\_questions` with the policy holder's sex, partner ID, a constant value of 2, and the policy holder's name.  
 - If the selected partner ID matches a policy member ID, call `populate\_questions` with the policy member's sex, partner ID, the policy member's IP number (or 1 if null), and the policy member's name.  
  
Definition of Done:  
- The "Populate Questions" button should trigger the described logic when pressed.  
- Error messages and function calls should work as specified.  
- The system should correctly identify and handle the selected partner ID, ensuring the appropriate questions are populated.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include specific CRUD operations or direct database queries.

# Process Declaration under Income Tax Rules 2015

Type: BBPROCESS

Detailed description: As a user, I want to ensure that the system correctly processes the declaration under Income Tax Rules 2015 based on the provided policy holder and insured person details, so that the necessary parameters are set and the appropriate form is called.  
  
Acceptance criteria:  
1. When the button is pressed, the system should set the variable `pk\_vars.v\_itrule` to 'Y'.  
2. The system should determine the value of `v\_ip\_no` based on the `PH\_EQUAL\_INS` attribute of the policy holder and the age of the insured person.  
3. The system should navigate to the 'FURTHER\_REQ' block and iterate through the records to check specific test numbers ('M682' and 'M711') and set the `v\_mand\_flag` and `v\_req\_flag` accordingly.  
4. The system should check the nationality of the insured person and set the `v\_nationality` and `v\_ip\_nat` variables.  
5. The system should determine if a passport proof is required based on the age proof of the insured person and policy holder, and the `v\_req\_flag`.  
6. If passport proof is not required, the system should navigate to the 'AML' block and iterate through the records to check the proof type and set the `v\_passport\_prf` variable.  
7. If the `v\_mand\_flag` is 'Y', the system should create a parameter list with the necessary parameters and call the 'AZBJ\_INCOMETAX\_QUEST\_DTLS' form.  
8. If the `v\_mand\_flag` is not 'Y', the system should display a warning message indicating that the declaration under Income Tax Rules 2015 is not received.  
  
Definition of Done:  
- The system correctly processes the declaration under Income Tax Rules 2015 based on the provided details.  
- The necessary parameters are set and the appropriate form is called.  
- The system displays the correct warning message if the declaration is not received.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include any direct database queries.

# Select Partner and Display Associated Questions and Answers

Type: BBPROCESS

Title: Select Partner and Display Associated Questions and Answers  
  
Acceptance Criteria:  
1. When a partner is selected, the system should check if the selected partner's ID matches the policy holder's or insured person's ID.  
2. If the selected partner's ID matches the policy holder's ID, the system should display the questions and answers related to the policy holder.  
3. If the selected partner's ID matches the insured person's ID, the system should display the questions and answers related to the insured person.  
4. If the selected partner's ID does not match either, the system should check if questions are already present for the selected partner.  
5. If questions are not present for the selected partner, the system should prompt the user to populate the questions.  
  
Definition of Done:  
1. The user can select a partner from a list.  
2. The system correctly identifies and displays the questions and answers based on the selected partner's ID.  
3. The system prompts the user to populate questions if they are not present for the selected partner.  
4. The functionality is tested and verified to ensure it works as expected without any errors.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- The following queries are used to fetch questions and answers based on the selected partner's ID:  
 ```sql  
 SELECT a.question\_desc, b.answer, b.description  
 FROM azbj\_bbu\_questions a, wip\_azbj\_bbu\_answers b  
 WHERE b.question\_id = a.question\_id  
 AND b.sub\_question = a.sub\_question  
 AND b.contract\_id = :control.cn\_contract\_id  
 AND partner\_no = 1;  
  
 SELECT a.question\_desc, b.answer, b.description  
 FROM azbj\_bbu\_questions a, wip\_azbj\_bbu\_answers b  
 WHERE b.question\_id = a.question\_id  
 AND b.sub\_question = a.sub\_question  
 AND b.contract\_id = :control.cn\_contract\_id  
 AND partner\_no = 2;  
 ```

# View BBU Results Button Functionality

Type: BBPROCESS

Title: View BBU Results Button Functionality  
  
Acceptance Criteria:  
1. When the "View BBU Results" button is clicked, the system should navigate to the "Result" section.  
2. If there are no records in the "Result" section or if the "FORM\_QST" field is null, the system should navigate to the "MED\_UW" section and display the first record.  
  
Definition of Done:  
- The "View BBU Results" button is present on the questionnaire tab.  
- Clicking the button navigates to the "Result" section.  
- If the "Result" section has no records or the "FORM\_QST" field is null, the system navigates to the "MED\_UW" section and displays the first record.  
- The functionality is tested and verified to ensure it works as expected.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- No specific DB queries are mentioned in the provided XML content that are directly executable without modification.

# Validate and Populate Questionnaire Answers

Type: BBPROCESS

User Story: Validate and Populate Questionnaire Answers  
  
Detailed description:   
As a user, I want to ensure that all questions in the questionnaire are answered correctly and that the answers are validated based on specific conditions, so that the data integrity and accuracy are maintained.  
  
Acceptance criteria:  
1. When the "Enter Answers" button is pressed, the system should:  
 - Enable the description field.  
 - Navigate to the questionnaire block and count the number of records.  
 - If there is only one record and both the form question number and question description are null, display an error message prompting the user to click on the "Populate Question" button.  
 - For each record in the questionnaire block:  
 - If the answer is not 'Y', set a flag to 'N'; otherwise, set it to 'Y'.  
 - If the answer is null and the answer type contains 'N', display an error message prompting the user to enter all answers.  
 - For specific question IDs and sub-questions, set global flags based on the answer values and display appropriate error messages if conditions are not met.  
 - If the answer is null for certain questions, display an error message prompting the user to provide details.  
 - After processing all records, if the flag is 'N', clear the description block.  
 - Navigate back to the questionnaire block and reset the record pointer to the first record.  
  
2. The system should sort and populate the description block based on the questionnaire answers:  
 - Find and validate the record group and columns.  
 - Clear the description block and populate it with sorted questionnaire data.  
 - For each record in the record group, set the question ID, sub-question, and form question number in the description block.  
  
Definition of Done:  
- The "Enter Answers" button functionality is implemented and tested.  
- All specified conditions and validations are met.  
- Appropriate error messages are displayed for invalid or incomplete data.  
- The description block is correctly populated based on the questionnaire answers.  
- The system navigates between blocks and records as specified.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Not applicable as the provided XML content does not include direct database queries that can be executed independently of Oracle Forms constructs.

# Ensure All Necessary Questions for Policy Members are Entered and Validated

Type: BBPROCESS

Detailed description: As a user, I want to ensure that all necessary questions for policy members are entered and validated before proceeding with the policy issuance process. This includes checking for incomplete applications, verifying specific conditions for different products, and ensuring that all required fields are populated correctly.  
  
Acceptance criteria:  
1. When the user initiates the process, the system should set the form status to 'N' and reset the question flag.  
2. If the RV\_STOP flag is 'N', the system should check if the annuity flag is 'N' and call the CHECK\_HNI\_POLICIES procedure.  
3. The system should query the `balic\_history\_details` table to check for previous lapse policies and handle any exceptions.  
4. The system should navigate to the 'AZBJ\_BBU\_QUESTIONS' block and check if questions for all policy members are entered.  
5. If the product is 'FAMILY\_CARE\_FIRST', the system should ensure that questions are populated for all members and add an error if not.  
6. The system should navigate through the 'POLICY\_MEMBER' block and ensure that questions are populated for each member.  
7. The system should check if the product is 'YOUNGCARE' or 'YOUNGCAREPLUS' and ensure that question 36 is answered.  
8. The system should call the `azbj\_bk\_black\_box` procedure.  
9. The system should enable or disable specific buttons based on certain conditions.  
10. The system should handle specific conditions for the 'MED\_UW' block, including setting the `MU\_TEST\_CONDUCTED\_FROM` field to null if certain conditions are met.  
11. The system should handle specific conditions for the 'RKB' branch, including adding warnings if certain criteria are met.  
12. The system should handle specific conditions for handicapped cases and add appropriate warnings.  
13. The system should handle specific conditions for unit-linked products and ensure that the allocated premium is greater than the frequency charges.  
14. The system should handle specific conditions for the 'FAMILY\_CARE\_FIRST' product and call the `Azbj\_FCF\_VALIDATIONS` procedure.  
15. The system should query the `azbj\_phub\_add\_req\_tracker` and `azbj\_phub\_req\_tracker` tables to check for incomplete applications and add warnings if necessary.  
16. The system should navigate to the 'Result' block and handle specific conditions based on the presence of certain fields.  
  
Definition of Done:  
- The system correctly navigates through the necessary blocks and performs all required checks and validations.  
- All necessary questions for policy members are entered and validated.  
- Appropriate warnings and errors are added based on specific conditions.  
- The system handles all exceptions and edge cases as specified.  
- The process is completed without any unhandled errors or issues.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Query to check for previous lapse policies:  
 ```sql  
 SELECT COUNT () INTO v\_cnt2  
 FROM balic\_history\_details  
 WHERE appl\_no = TO\_CHAR(NVL(:insured\_person.ip\_sign\_card\_no, :insured\_person.ip\_verf\_no))  
 AND UPPER(tele\_ver\_status) = 'COMPLETE'  
 AND updated\_flag = 'Y';  
 ```  
- Query to check for incomplete applications:  
 ```sql  
 SELECT COUNT()  
 INTO v\_incomplete\_flg  
 FROM azbj\_phub\_add\_req\_tracker a, azbj\_phub\_req\_tracker b  
 WHERE a.APPLICATION\_NO = b.APPLICATION\_NO  
 AND a.REQ\_TYPE = b.REQ\_TYPE  
 AND a.application\_no = TO\_CHAR(NVL(:INSURED\_PERSON.IP\_VERF\_NO, :INSURED\_PERSON.IP\_SIGN\_CARD\_NO))  
 AND b.RECD\_IN\_OPUS = 'N';  
 ```

# Populate Questions Button Functionality

Type: BBPROCESS

Title: Populate Questions Button Functionality  
  
Acceptance Criteria:  
1. When the "Populate Questions" button is pressed, the system should:  
 - Disable the 'DESCRIPTION\_ph.DESCRIPTION' field.  
 - Clear the blocks 'azbj\_bbu\_questions\_ph' and 'DESCRIPTION\_ph'.  
 - If the product ID in the control block is one of 221, 259, or 297, populate the 'azbj\_bbu\_questions\_ph' block with questions from a specific cursor (C3) based on the policyholder's sex and other criteria.  
 - If the product ID is not in a specific list ('GROUP\_CREDIT\_PROTECT', 'GROUP\_LOAN\_PROTECTOR', 'GROUP\_CREDIT\_PROTECTION\_PLUS'), populate the 'azbj\_bbu\_questions\_ph' block with questions from another cursor (C1) based on the policyholder's sex and other criteria.  
 - If none of the above conditions are met, populate the 'azbj\_bbu\_questions\_ph' block with questions from cursor C1.  
 - Ensure the 'azbj\_bbu\_questions\_ph' block is populated with the question ID, sub-question, form question number, question description, and answer type.  
 - Delete any existing record group named 'QuesRg\_ph'.  
  
Definition of Done:  
- The "Populate Questions" button functionality is implemented as per the acceptance criteria.  
- The relevant questions are displayed correctly based on the policyholder's details and product ID.  
- The system handles different product IDs and conditions appropriately.  
- The 'DESCRIPTION\_ph.DESCRIPTION' field is disabled, and the blocks 'azbj\_bbu\_questions\_ph' and 'DESCRIPTION\_ph' are cleared before populating new data.  
- The record group 'QuesRg\_ph' is deleted if it exists.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- The following SQL queries are used to fetch questions based on specific criteria:  
 ```sql  
 SELECT FROM azbj\_bbu\_questions  
 WHERE displayed = 'Y'  
 AND QUESTION\_ID <> (CASE WHEN :Policy\_holder.PH\_SEX = 'F' THEN 0 ELSE 6 END)  
 AND QUESTION\_ID NOT IN (17, 18, 19)  
 AND QUESTION\_ID < 25  
 ORDER BY question\_id, SUB\_QUESTION;  
  
 SELECT FROM azbj\_bbu\_questions  
 WHERE displayed = 'Y' AND question\_id > 48 AND question\_id < 51  
 UNION ALL  
 SELECT FROM azbj\_bbu\_questions  
 WHERE displayed = 'Y'  
 AND :policy\_holder.PH\_SEX = 'F'  
 AND question\_id = 6;  
  
 SELECT DISTINCT a.question\_id, b.sub\_question, b.form\_question\_no,  
 b.question\_desc, b.answer\_type  
 FROM azbj\_strip\_questions\_mapping a, azbj\_bbu\_questions b  
 WHERE a.question\_id = b.question\_id  
 AND a.sub\_question\_id = b.sub\_question  
 AND proposal\_type = 'N'  
 AND b.question\_id <> (CASE WHEN :policy\_holder.PH\_SEX = 'F' THEN 0 ELSE 6 END)  
 ORDER BY to\_number(a.question\_id), b.sub\_question;  
 ```

# Validate Policy Information

Type: CLIENT\_ENV

User Story: Validate Policy Information  
  
Detailed description:   
As a user, I want to validate the policy information to ensure that all necessary details are correctly entered and verified before proceeding with further actions. This includes checking the completeness of required fields, ensuring that all policy members have their questions answered, and verifying that the policy meets specific criteria such as premium amounts and product definitions.  
  
Acceptance criteria:  
1. The system should check if the policy's annuity flag is set to 'N'. If it is, it should proceed to validate the policy.  
2. The system should verify if the policy is not in the list of specific products ('SWAYAM\_SHAKTI\_SURAKSHA', 'SARVE\_SHAKTI\_SURAKSHA', 'GROUP\_SEVA\_PLAN').  
3. If the annuity flag is 'N', the system should check for HNI policies and validate them.  
4. The system should count the number of records in the `balic\_history\_details` table where the application number matches the insured person's verification number or sign card number, and the tele verification status is 'COMPLETE' and updated flag is 'Y'.  
5. If the count is greater than zero, the system should check for previous lapse policies.  
6. The system should navigate to the 'AZBJ\_BBU\_QUESTIONS' block and ensure that all questions for policy members are entered.  
7. If the product definition is 'FAMILY\_CARE\_FIRST', the system should ensure that questions are populated for all members.  
8. The system should validate if the question ID 36 is answered with 'Y' or 'N' for products 'YOUNGCARE' and 'YOUNGCAREPLUS'.  
9. The system should enable or disable specific buttons based on the validation results.  
10. The system should check if the policy is logged through the RKB branch and meets certain criteria (e.g., TASA greater than 3000000 and multiplier greater than 1.25) and refer the case to HO if necessary.  
11. The system should validate if the policy is an IPP Disable Life Case and refer it to RHOBR if necessary.  
12. The system should ensure that the allocated premium is greater than the frequency charges if loadings are present.  
13. The system should perform additional validations for specific products and scenarios as described in the logic.  
  
Definition of Done:  
- The policy validation logic is implemented and tested.  
- All acceptance criteria are met.  
- The system correctly identifies and handles incomplete or invalid policies.  
- The system enables or disables buttons based on the validation results.  
- The system logs appropriate messages and errors for any validation failures.  
- The system refers cases to HO or RHOBR as required based on the validation logic.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- The system should execute the following query to count records in the `balic\_history\_details` table:  
 ```sql  
 SELECT COUNT () INTO v\_cnt2  
 FROM balic\_history\_details  
 WHERE appl\_no = TO\_CHAR (NVL (:insured\_person.ip\_sign\_card\_no, :insured\_person.ip\_verf\_no))  
 AND UPPER(tele\_ver\_status) = 'COMPLETE'  
 AND updated\_flag = 'Y';  
 ```  
  
- The system should execute the following query to check for incomplete inwarded applications:  
 ```sql  
 SELECT COUNT()  
 INTO v\_incomplete\_flg  
 FROM azbj\_phub\_add\_req\_tracker a, azbj\_phub\_req\_tracker b  
 WHERE a.APPLICATION\_NO = b.APPLICATION\_NO  
 AND a.REQ\_TYPE = b.REQ\_TYPE  
 AND a.application\_no = TO\_CHAR(NVL(:INSURED\_PERSON.IP\_VERF\_NO, :INSURED\_PERSON.IP\_SIGN\_CARD\_NO))  
 AND b.RECD\_IN\_OPUS = 'N';  
 ```

# Implement 'Rate' Button for Policy Validation

Type: CLIENT\_ENV

Title: Implement 'Rate' Button for Policy Validation  
  
Acceptance Criteria:  
- When the 'Rate' button is pressed, the system should trigger the validation process for the policy.  
- The button should be visible and enabled on the client environment screen.  
- The button should be labeled 'Rate' and should have a hint that says 'Validate Policy'.  
  
Definition of Done:  
- The 'Rate' button is implemented and visible on the client environment screen.  
- Pressing the 'Rate' button triggers the policy validation process.  
- The button is labeled 'Rate' and displays the hint 'Validate Policy' when hovered over.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Not applicable as the provided XML content does not include any specific database queries.

# Ensure Policy Cannot Be Skipped Without Partner Creation

Type: CLIENT\_ENV

Title: Ensure Policy Cannot Be Skipped Without Partner Creation  
  
Acceptance Criteria:  
1. If the insured person's partner reference is null, the system should display an error message: "Cannot Skip Policy As Partner Is Not Created. Kindly Create Partner And Proceed."  
2. If the policy holder's partner reference is null, the system should display an error message: "Cannot Skip Policy As Partner Is Not Created. Kindly Create Partner And Proceed."  
3. If both partner references are present, the system should set the action to 'SE' and proceed to the next step.  
  
Definition of Done:  
- The system correctly identifies when the partner references are null and displays the appropriate error messages.  
- The action is set to 'SE' and the system proceeds to the next step when both partner references are present.  
- The functionality is tested and verified to ensure it works as expected without any Oracle Forms-specific dependencies.

# Validate Policy Button Functionality

Type: CLIENT\_ENV

Title: Validate Policy Button Functionality  
  
Acceptance Criteria:  
1. When the "Validate Policy" button is pressed, the system should temporarily suppress any non-critical messages.  
2. The system should execute a validation procedure to check the policy information.  
3. After the validation procedure, the system should restore the message level to its original state.  
4. If the validation is successful and certain conditions are met (e.g., a specific control value is 'T' and a result field is not null), the system should disable a specific item to prevent further actions.  
5. If any errors occur during the validation process, the system should display an error message with the details of the error.  
  
Definition of Done:  
- The "Validate Policy" button triggers the validation process as described.  
- The system suppresses and restores message levels appropriately.  
- The validation procedure is executed and the system responds based on the validation results.  
- Error handling is implemented to display error messages when exceptions occur.  
- The specific item is disabled based on the conditions mentioned.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include any direct database queries.

# Save Only Button Functionality

Type: CLIENT\_ENV

Detailed description: As a user, I want to be able to save policy details and validate various conditions when I press the "Save Only" button, so that the system ensures data integrity and proper validation before committing the data.  
  
Acceptance criteria:  
1. When the "Save Only" button is pressed, the system should:  
 - Execute a series of validation and save procedures.  
 - Check for the existence of specific entries in the `azbj\_ip\_ext` and `wip\_azbj\_ip\_ext` tables based on the contract ID and IP number.  
 - If certain entries are not found, log an error message with the contract ID and policy status.  
 - If certain entries are found, log a different error message with the contract ID and policy status.  
 - If the `incmp\_prp\_dtls\_save\_flg` flag is set to 'Y', execute the `azbj\_incmpf\_save\_dtls` procedure.  
 - If the `SAVE\_BTN\_FLAG` flag is set to 'Y', execute the `azbj\_save\_dupl\_contact\_no\_dtls` procedure.  
 - For specific product IDs, update or insert records in the `azbj\_policy\_covers\_risk\_sa` table based on cover codes and other conditions.  
 - Check for the existence of preferred IC entries and update or insert records in the `AZBJ\_MEDUW\_PREFERED\_IC` table accordingly.  
 - Commit the transaction.  
 - Log the user action as 'UPDATE' with the description 'SAVE-ONLY'.  
 - Handle any exceptions by logging an appropriate error message.  
  
Definition of Done:  
- The "Save Only" button functionality is implemented and tested.  
- All specified validations and procedures are executed correctly.  
- Error messages are logged appropriately when conditions are not met.  
- Data is committed to the database only after all validations pass.  
- User actions are logged as specified.  
- The functionality is verified through unit tests and integration tests.  
  
DB queries for Table reference CRUD operations only(With Usage):  
```sql  
-- Check for existence of entries in azbj\_ip\_ext and wip\_azbj\_ip\_ext tables  
SELECT COUNT(1) INTO v\_cnt\_ip\_ext1\_1 FROM azbj\_ip\_ext WHERE contract\_id = :control.cn\_contract\_id AND ip\_no = 1;  
SELECT COUNT(1) INTO v\_cnt\_ip\_ext2\_1 FROM wip\_azbj\_ip\_ext WHERE contract\_id = :control.cn\_contract\_id AND ip\_no = 1;  
SELECT COUNT(1) INTO v\_cnt\_ip\_ext1\_2 FROM azbj\_ip\_ext WHERE contract\_id = :control.cn\_contract\_id AND ip\_no = 2;  
SELECT COUNT(1) INTO v\_cnt\_ip\_ext2\_2 FROM wip\_azbj\_ip\_ext WHERE contract\_id = :control.cn\_contract\_id AND ip\_no = 2;  
  
-- Update or insert records in azbj\_policy\_covers\_risk\_sa table  
UPDATE azbj\_policy\_covers\_risk\_sa  
SET sum\_insured\_whole\_cover = :COVERS.cv\_freq\_std\_prem TO\_NUMBER(:coverhead.ch\_booking\_frequency) 10,  
 claimable\_sa = :COVERS.cv\_freq\_std\_prem TO\_NUMBER(:coverhead.ch\_booking\_frequency) 10  
WHERE contract\_id = :control.cn\_contract\_id;  
  
IF SQL%NOTFOUND THEN  
 INSERT INTO azbj\_policy\_covers\_risk\_sa (contract\_id, action\_code, top\_indicator, product\_id, cover\_code, sum\_insured\_whole\_cover, claimable\_sa, claimable\_year, create\_date)  
 VALUES (:control.cn\_contract\_id, 'A', 'Y', :control.cn\_product\_id, :COVERS.CV\_COVER\_CODE, :COVERS.cv\_freq\_std\_prem TO\_NUMBER(:coverhead.ch\_booking\_frequency) 10, :COVERS.cv\_freq\_std\_prem TO\_NUMBER(:coverhead.ch\_booking\_frequency) 10, 0, pme\_api.opus\_date);  
END IF;  
  
-- Check for existence of preferred IC entries and update or insert records in AZBJ\_MEDUW\_PREFERED\_IC table  
SELECT COUNT() INTO v\_prefered\_ic\_cnt FROM azbj\_ic\_upload\_dtls WHERE ic\_code = :agents.ag\_agent\_code AND ic\_type = 'PREFERRED\_IC';  
  
IF v\_prefered\_ic\_cnt > 0 THEN  
 UPDATE AZBJ\_MEDUW\_PREFERED\_IC  
 SET ACTIVE\_FLAG = 'Y', CREATE\_DATE = SYSDATE, CREATE\_USER = USER  
 WHERE AGENT\_CODE = :agents.ag\_agent\_code;  
  
 IF SQL%NOTFOUND THEN  
 INSERT INTO AZBJ\_MEDUW\_PREFERED\_IC (AGENT\_CODE, ACTIVE\_FLAG, CREATE\_USER, CREATE\_DATE)  
 VALUES (:agents.ag\_agent\_code, 'Y', USER, SYSDATE);  
 END IF;  
END IF;  
```

# Handle Scrutiny Failure Button Press

Type: CLIENT\_ENV

User Story: Handle Scrutiny Failure Button Press  
  
Detailed Description:  
As a user, I want to ensure that when I press the "Scrutiny Failure" button, the system validates the application number and checks if the user is authorized. If the application number is valid and the user is authorized, the system should pass specific parameters to another form for further processing. If the application number is invalid or the user is not authorized, appropriate error messages should be displayed.  
  
Acceptance Criteria:  
1. When the "Scrutiny Failure" button is pressed:  
 - The system should retrieve the application number from the insured person's verification number or sign card number.  
 - If the application number is null, the system should prompt the user to enter the application number and display an error message: "Application no can not be null".  
 - If the user ID starts with 'UU', the system should:  
 - Create a parameter list named 'input\_params'.  
 - Add parameters 'RES\_TYPE' with value 'BBU' and 'RES\_appln' with the application number to the parameter list.  
 - Call the form 'AZBJ\_SCRUTINY\_FAILURE' with the parameter list.  
 - If the user ID does not start with 'UU', the system should display an error message: "Not an Authorized ID".  
  
Definition of Done:  
- The system correctly validates the application number and user authorization when the "Scrutiny Failure" button is pressed.  
- Appropriate error messages are displayed for invalid application numbers or unauthorized users.  
- The form 'AZBJ\_SCRUTINY\_FAILURE' is called with the correct parameters when the user is authorized.  
  
DB Queries for Table Reference CRUD Operations:  
- Not applicable as the provided logic does not include direct database CRUD operations.  
  
Explanation of Oracle Form Logic:  
- The logic involves checking the application number and user authorization.  
- If the application number is null, the system navigates to the item where the application number should be entered and displays an error message.  
- If the user ID starts with 'UU', the system creates a parameter list, adds necessary parameters, and calls another form with these parameters.  
- If the user ID does not start with 'UU', an error message is displayed indicating the user is not authorized.

# Validate Policy Member Information and Handle Specific Conditions

Type: CLIENT\_ENV

Detailed description: As a user, I want to ensure that the system validates the completeness of policy member information and handles specific conditions for different product types, so that accurate and complete data is maintained for policy issuance.  
  
Acceptance criteria:  
1. The system should set the form status to 'N' and reset the question flag when the button is pressed.  
2. If the RV\_STOP flag is 'N' and the annuity flag is 'N', the system should check for HNI policies.  
3. The system should query the `balic\_history\_details` table to check for previous lapse policies and handle exceptions.  
4. The system should navigate to the 'AZBJ\_BBU\_QUESTIONS' block and check if questions for all policy members are entered.  
5. For the 'FAMILY\_CARE\_FIRST' product, the system should ensure that questions are populated for all members and add an error if not.  
6. The system should handle specific conditions for the 'YOUNGCARE' and 'YOUNGCAREPLUS' products, ensuring that question 36 is answered.  
7. The system should enable or disable specific buttons based on user flags and other conditions.  
8. The system should handle the visibility and navigation of tab pages and blocks based on certain conditions.  
9. The system should validate and handle specific conditions for the 'RKB' branch and products with high TASA or premium values.  
10. The system should validate and handle cases for disabled life policies and unit-linked products.  
11. The system should check for incomplete inwarded applications and add appropriate errors.  
12. The system should navigate through various blocks and handle specific conditions for medical underwriting and other validations.  
  
Definition of Done:  
- The system correctly sets the form status and resets flags as specified.  
- The system accurately checks for HNI policies and handles exceptions.  
- The system navigates to the appropriate blocks and ensures questions are populated for all members.  
- The system handles specific conditions for different product types and adds errors where necessary.  
- The system enables or disables buttons based on user flags and other conditions.  
- The system validates and handles specific conditions for the 'RKB' branch and products with high TASA or premium values.  
- The system validates and handles cases for disabled life policies and unit-linked products.  
- The system checks for incomplete inwarded applications and adds appropriate errors.  
- The system navigates through various blocks and handles specific conditions for medical underwriting and other validations.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Query to check for previous lapse policies:  
 ```sql  
 SELECT COUNT () INTO v\_cnt2  
 FROM balic\_history\_details  
 WHERE appl\_no = TO\_CHAR (NVL (:insured\_person.ip\_sign\_card\_no, :insured\_person.ip\_verf\_no))  
 AND UPPER(tele\_ver\_status) = 'COMPLETE'  
 AND updated\_flag = 'Y';  
 ```  
- Query to check for incomplete inwarded applications:  
 ```sql  
 SELECT COUNT()  
 INTO v\_incomplete\_flg  
 FROM azbj\_phub\_add\_req\_tracker a, azbj\_phub\_req\_tracker b  
 WHERE a.APPLICATION\_NO = b.APPLICATION\_NO  
 AND a.REQ\_TYPE = b.REQ\_TYPE  
 AND a.application\_no = TO\_CHAR(NVL(:INSURED\_PERSON.IP\_VERF\_NO, :INSURED\_PERSON.IP\_SIGN\_CARD\_NO))  
 AND b.RECD\_IN\_OPUS = 'N';  
 ```

# Policy Validation on Validate Button Press

Type: CLIENT\_ENV

Title: Policy Validation on Validate Button Press  
  
Acceptance Criteria:  
1. When the "Validate" button is pressed, the system should reset the variable `v\_covers\_call` to 0.  
2. The system should call a procedure to perform preliminary claim validation.  
3. The system should check if the application number exists in the `azbj\_application\_bypass\_det` table with a bypass flag set to 'Y' and bypass module as 'AADHAR\_CARD\_BYPASS'. If found, it should proceed with further checks.  
4. The system should count the number of records in the `azbj\_supervisor\_appr\_det` table where the application number matches and the PAN issuance date is different from the provided date. If the count is zero or greater than zero, it should call a procedure for PAN verification.  
5. If the specified person sub IC code is not null and matches certain patterns, and the product ID is not in a specified list, the system should add an error indicating that POSP is not allowed to log in for this product.  
6. The system should check if the suspected case flag is set to 'Y' and if the suspected screen flag is 'N'. If true, it should add an error message prompting the user to select the dropdown list for name and DOB matching on the suspected CP search screen.  
7. The system should check if the agent's sub code is null and the agent code does not match certain patterns. If true, it should add an error indicating that Lead By is mandatory for the channel.  
8. The system should check if the referral ID is null and the agent code does not match certain patterns. If true, it should add an error indicating that Bank/Referral ID is mandatory for the channel.  
9. If the method of payment is 'UPI\_M' and the premium amount is less than or equal to 5000, the system should disable the method of payment field.  
10. If the agent's BI number is not null and the product ID matches certain values, the system should set the premium term and benefit term accordingly.  
11. The system should check if the product ID matches certain values and if the premium term is not null. If true, it should disable the premium term field.  
12. If the method of payment is 'S', the system should disable the method of payment field.  
  
Definition of Done:  
- The "Validate" button triggers all the specified checks and validations.  
- Appropriate error messages are displayed based on the conditions.  
- Relevant fields are enabled or disabled based on the conditions.  
- The system performs all necessary database queries and updates as specified.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- The system should execute the following queries:  
 - Count records in `azbj\_application\_bypass\_det` where application number matches and bypass flag is 'Y' and bypass module is 'AADHAR\_CARD\_BYPASS'.  
 - Count records in `azbj\_supervisor\_appr\_det` where application number matches and PAN issuance date is different from the provided date.  
 - Select suspected case from `azbj\_phub\_scrutiny\_prop` where application number matches.  
 - Count records in `azbj\_supervisor\_appr\_det` where application number matches and create date is the maximum date for the application number.

# Manual Case Push to UW Comments

Type: CLIENT\_ENV

Title: Manual Case Push to UW Comments  
  
Acceptance Criteria:  
1. When the "Reason for Manual" button is pressed, the system should check if there are any existing transactions with action\_id = 2 for the given application number.  
2. If such transactions exist, the system should:  
 - Navigate to the "RES\_MAN\_BBU" section.  
 - Clear the current block without validation.  
 - Retrieve and display specific messages based on the source\_flag and other criteria from the database.  
 - Populate the "rule\_message" and "param\_value" fields with the retrieved data.  
 - Show the "RES\_MAN\_BBU" window and view.  
 - Navigate to the "RES\_MAN\_BBU.EXIT" item.  
3. If no transactions are found, the system should display a warning message stating "No Details available for this case."  
  
Definition of Done:  
- The "Reason for Manual" button functionality is implemented and tested.  
- The system correctly identifies and processes existing transactions.  
- The appropriate messages are displayed in the UW comments section.  
- The warning message is displayed when no relevant transactions are found.  
- The feature is tested and verified to work as expected in the user interface.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- The system should execute the following query to check for existing transactions:  
 ```sql  
 SELECT COUNT(1)  
 INTO v\_count  
 FROM bbu\_trans a, bbu\_trans\_dtls b  
 WHERE a.trans\_id = b.trans\_id AND action\_id = 2  
 AND a.appl\_no = NVL(:insured\_person.ip\_sign\_card\_no, TO\_CHAR(:insured\_person.ip\_verf\_no));  
 ```  
  
- The system should execute the following query to retrieve and display specific messages:  
 ```sql  
 SELECT DISTINCT (CASE  
 WHEN source\_flag = 'RULE' THEN 'Rule No '  
 WHEN source\_flag = 'QC' THEN 'Question No '  
 WHEN source\_flag = 'MED' THEN 'Medical Requirements'  
 ELSE NULL  
 END  
 || NVL(a.rule\_config\_id, a.question\_id)  
 || '('  
 || NVL(fixed\_msg, question\_desc)  
 || ')'  
 || ' Raised Requirement '  
 || sys\_code  
 || '( '  
 || sys\_desc  
 || ' )'  
 || (CASE  
 WHEN req\_received\_date IS NOT NULL THEN ' And Received at DEQC'  
 ELSE ' And Not received at DEQC'  
 END  
 ) rule\_message,  
 (CASE  
 WHEN source\_flag = 'RULE' THEN 'Rule Configurator'  
 WHEN source\_flag = 'MED' THEN 'Medical Requirements'  
 WHEN source\_flag = 'QC' THEN 'QC REQUIREMENTS'  
 ELSE NULL  
 END  
 ) param\_type  
 FROM bbu\_trans\_rule\_req\_dtls a, azbj\_system\_constants b, bbu\_rule\_config c, azbj\_qc\_questions d, bbu\_trans e  
 WHERE a.trans\_id = e.trans\_id  
 AND e.appl\_no = NVL(:insured\_person.ip\_sign\_card\_no, TO\_CHAR(:insured\_person.ip\_verf\_no))  
 AND a.req\_code = b.sys\_code  
 AND a.rule\_config\_id = c.rule\_config\_id(+)  
 AND a.question\_id = d.question\_id(+)  
 UNION ALL  
 SELECT UNIQUE 'Question No '  
 || TO\_NUMBER(SUBSTR(rule\_message, 36, LENGTH(rule\_message)))  
 || '('  
 || d.question\_desc  
 || ') Failed at DEQC',  
 'QC REQUIREMENTS'  
 FROM bbu\_trans\_dtls a, azbj\_qc\_questions d, bbu\_trans b  
 WHERE a.trans\_id = b.trans\_id  
 AND b.appl\_no = NVL(:insured\_person.ip\_sign\_card\_no, TO\_CHAR(:insured\_person.ip\_verf\_no))  
 AND action\_id = 2  
 AND rule\_message <> 'QC Requirements raised.'  
 AND TO\_NUMBER(SUBSTR(rule\_message, 36, LENGTH(rule\_message))) = d.question\_id(+)  
 AND param\_val\_string = 'QC REQUIREMENTS'  
 AND 1 = (CASE  
 WHEN rule\_message NOT LIKE 'QC Failed for question %'  
 AND rule\_message NOT LIKE 'QC Requirements raised for question %' THEN 0  
 ELSE 1  
 END)  
 AND NOT EXISTS (SELECT 1  
 FROM azbj\_qc\_ques\_req  
 WHERE question\_id = TO\_NUMBER(SUBSTR(rule\_message, 36, LENGTH(rule\_message))))  
 AND LENGTH(rule\_message) > 23  
 UNION ALL  
 SELECT UNIQUE (CASE  
 WHEN a.rule\_config\_id IS NOT NULL THEN 'Rule No ' || a.rule\_config\_id || '( ' || rule\_message || ' ) Failed at Rule Configurator '  
 ELSE rule\_message  
 END),  
 (CASE  
 WHEN a.rule\_config\_id IS NOT NULL THEN 'Rule Configurator'  
 ELSE param\_val\_string  
 END)  
 FROM bbu\_trans\_dtls a, bbu\_rule\_config d, bbu\_trans c  
 WHERE NOT EXISTS (SELECT   
 FROM bbu\_req\_rule b  
 WHERE b.rule\_config\_id = a.rule\_config\_id)  
 AND a.rule\_config\_id = d.rule\_config\_id(+)  
 AND a.trans\_id = c.trans\_id  
 AND a.action\_id = 2  
 AND c.appl\_no = NVL(:insured\_person.ip\_sign\_card\_no, TO\_CHAR(:insured\_person.ip\_verf\_no))  
 AND a.rule\_config\_id IS NOT NULL  
 UNION ALL  
 SELECT UNIQUE a.rule\_message rule\_description, param\_val\_string  
 FROM bbu\_trans\_dtls a, bbu\_trans b  
 WHERE a.action\_id = 2  
 AND a.trans\_id = b.trans\_id  
 AND b.appl\_no = NVL(:insured\_person.ip\_sign\_card\_no, TO\_CHAR(:insured\_person.ip\_verf\_no))  
 AND rule\_message NOT IN ('Medical Requirements raised in UW Rule Validations',  
 'Product Validations Failed.',  
 'QC Requirements raised.',  
 'Issuance Validations Failure...ORA-0000: normal, successful completion')  
 AND rule\_message NOT LIKE 'QC Requirements raised for question%'  
 AND param\_val\_string IN ('PRODUCT VALIDATIONS', 'UW RULES', 'QC REQUIREMENTS', 'ISSUANCE')  
 ORDER BY 2;  
 ```

# Automatic Navigation on New Record Instance

Type: DESCRIPTION\_PH

Title: Automatic Navigation on New Record Instance  
  
Acceptance Criteria:  
1. When navigating to a new record in the "Description" section, the system should check if the "Form Question Number 1" field is empty.  
2. If the "Form Question Number 1" field is empty, the system should automatically navigate to the previous record.  
3. The "Form Question Number 1" field should be editable and visible to the user.  
4. The "Description" field should be editable and visible to the user.  
5. The "Question ID" and "Sub Question" fields should be hidden and not editable by the user.  
  
Definition of Done:  
- The system correctly checks the "Form Question Number 1" field when navigating to a new record.  
- The system navigates to the previous record if the "Form Question Number 1" field is empty.  
- All specified fields are displayed and behave as described in the acceptance criteria.  
- The functionality is tested and verified to work as expected.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- No specific database queries are provided in the XML content.

# Input and View Detailed Descriptions

Type: DESCRIPTION\_PH

Title: Input and View Detailed Descriptions  
  
Acceptance Criteria:  
1. The text field should be enabled and allow users to input up to 1000 characters.  
2. The text field should be positioned at the specified coordinates within the form.  
3. The text field should have a white background and black text for readability.  
4. The text field should display a prompt "Details if Marked Yes" above it.  
5. The text field should be single-line and not allow multiline input.  
6. When the user presses the down arrow key, the cursor should move to the next item in the form.  
  
Definition of Done:  
1. The text field is implemented and visible in the "New Business" module.  
2. The text field meets all specified criteria for appearance and functionality.  
3. The prompt "Details if Marked Yes" is displayed correctly above the text field.  
4. The cursor navigation using the down arrow key works as expected.  
5. The feature is tested and verified to ensure it meets the acceptance criteria.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the text field is not a database item.

# Display Results for Black Box Underwriting

Type: RESULT

Title: Display Results for Black Box Underwriting  
  
Acceptance Criteria:  
1. The results interface should display a list of records with a maximum of 5 records visible at a time.  
2. Each record should include the following fields:  
 - A row number.  
 - A question number (hidden from view).  
 - A sub-question ID (hidden from view).  
 - The question text.  
 - The result status.  
 - The action required.  
3. The interface should have a scrollbar to navigate through the records if there are more than 5.  
4. The question text, result status, and action required fields should be read-only and not editable by the user.  
5. The interface should be presented in a dialog window titled "Results for Black Box Underwriting" and should not be resizable.  
  
Definition of Done:  
- The results interface is implemented and displays the required fields as specified.  
- The interface allows navigation through records using a scrollbar.  
- The fields are correctly populated and are read-only.  
- The dialog window is titled "Results for Black Box Underwriting" and is not resizable.  
- The implementation is tested and verified to meet the acceptance criteria.  
  
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 Previous Reinsurance Details

Type: AZBJ\_PREV\_RINS\_DET

Title: View Previous Reinsurance Details  
  
Acceptance Criteria:  
1. The user should be able to see the following details for each reinsurance policy:  
 - Policy Reference  
 - Cover Description  
 - Reinsurance Code  
 - Reinsurance Percentage  
 - Cover Code  
 - Cover SAR  
 - Reinsurance Type  
 - Reference Number  
 - Reinsurance SA  
 - Retain Amount  
2. The details should be displayed in a structured format with appropriate labels.  
3. The fields should be read-only and not allow any modifications.  
4. The user should have an option to exit the screen.  
  
Definition of Done:  
1. The reinsurance details screen is implemented and displays all the required fields.  
2. The fields are read-only and cannot be edited by the user.  
3. The screen has an exit button that allows the user to close the screen.  
4. The screen is tested and verified to ensure all details are displayed correctly and the exit functionality works as expected.

# Navigate to Reinsurance Section on Exit Button Press

Type: AZBJ\_PREV\_RINS\_DET

Title: Navigate to Reinsurance Section on Exit Button Press  
  
Acceptance Criteria:  
- When the 'Exit' button is pressed, the system should automatically navigate to the 'Reinsurance' section.  
  
Definition of Done:  
- The 'Exit' button is functional and correctly navigates to the 'Reinsurance' section upon being pressed.  
- The navigation should be seamless and should not require any additional user input.  
- The feature should be tested and verified to ensure it works as expected in different scenarios.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include any database CRUD operations.

# Manage Clauses

Type: CLAUSES

Title: Manage Clauses  
  
Acceptance Criteria:  
1. The system should display a list of clauses with a maximum of three records visible at a time.  
2. Each clause should have a description field that is prominently displayed.  
3. There should be a button to delete a selected clause record.  
4. The interface should be user-friendly, with clear prompts and labels for each field and button.  
  
Definition of Done:  
1. The clause management interface is implemented and accessible.  
2. Users can view a list of clauses with a maximum of three records displayed at once.  
3. Users can see the description of each clause.  
4. Users can delete a clause record by clicking the "Delete Record" button.  
5. The interface is visually appealing and easy to navigate, with clear prompts and labels.  
  
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.

# Delete Record Button Functionality

Type: CLAUSES

Title: Delete Record Button Functionality  
  
Acceptance Criteria:  
1. When the "Delete Record" button is pressed:  
 - If the insured person's national ID is either 'FOREIGN NATIONAL' or 'NRI':  
 - If the clause description is 'NRI Clause', display an error message stating "NRI Clause is mandatory for NRI/FOREIGN NATIONAL case".  
2. If the above conditions are not met, proceed to delete the record.  
  
Definition of Done:  
- The "Delete Record" button should trigger the specified conditions and actions.  
- The error message should be displayed correctly when the conditions are met.  
- The record should be deleted if the conditions are not met.  
- The functionality should be tested and verified for accuracy.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Not applicable as the provided logic does not include direct database queries.

# View and Select Clauses

Type: CLAUSES

Title: View and Select Clauses  
  
Acceptance Criteria:  
1. When the user double-clicks on the clause description field or presses a specific key, the system should gather all clause descriptions from the current record set.  
2. The system should then query the database to retrieve a list of clauses that are not already present in the current record set.  
3. The retrieved list of clauses should be displayed in a pop-up window for the user to select from.  
4. The pop-up window should be titled "Clauses" and should automatically adjust column widths for better readability.  
5. The system should handle any errors that occur during the process and display an appropriate error message to the user.  
  
Definition of Done:  
- The functionality to view and select clauses is implemented and tested.  
- The list of clauses excludes any clauses already present in the current record set.  
- The pop-up window displays the list of clauses with appropriate column widths.  
- Error handling is implemented and tested to ensure that any issues are communicated to the user.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- The query to retrieve the list of clauses not present in the current record set:  
 ```sql  
 SELECT sys\_code, sys\_desc   
 FROM azbj\_system\_constants   
 WHERE sys\_type = 'CLAUSES'   
 AND sys\_code NOT IN (clause\_string)  
 ORDER BY sys\_code;  
 ```

# View and Manage Revised Offer Details

Type: AZBJ\_REVISED\_OFFER

Title: View and Manage Revised Offer Details  
  
Acceptance Criteria:  
1. The system should display the following fields for each revised offer:  
 - Policy Reference  
 - Contract ID  
 - Activity Number  
 - Activity Date  
 - Event Type  
 - User ID  
 - Product ID  
 - Package Code  
 - Cover Code  
 - Sum Assured  
 - Benefit Term  
 - Premium Term  
 - Premium Amount  
 - Multiplier  
 - ML Percentage  
 - ML Amount  
 - OC Percentage  
 - OC Amount  
 - SR Percentage  
 - SR Amount  
 - C Offer Reason  
 - NRI Percentage  
 - NRI Amount  
2. The fields should be read-only and not allow insertions or updates.  
3. The system should support scrolling through multiple records, displaying up to 10 records at a time.  
4. The system should ensure that the data displayed is accurate and up-to-date.  
  
Definition of Done:  
- The user can view all the specified fields for revised offers.  
- The fields are read-only and cannot be modified by the user.  
- The system displays up to 10 records at a time and allows scrolling through additional records.  
- The data displayed is accurate and reflects the latest information.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Not applicable as the provided XML content does not include specific database queries or operations.

# Manage Health Cover Details

Type: COVERS\_HEALTH

Detailed description: As a user, I want to manage health cover details within a form, so that I can input, view, and validate various attributes related to health insurance covers.  
  
Acceptance criteria:  
1. The form should allow the user to input and view the following attributes for health covers:  
 - Cover Code  
 - Cover Description  
 - Sum Insured for the Whole Cover  
 - Benefit Term  
 - Premium Term  
 - Entry Age  
 - Annual and Frequency-based Monetary Amounts  
 - Standard Premiums  
 - Discount Percentages and Amounts  
 - Rounding Offsets  
 - Deletion Checkbox  
2. The form should validate that the Cover Code and Cover Description are populated from a predefined list of values.  
3. The form should ensure that numeric fields such as Sum Insured, Benefit Term, Premium Term, and others are validated for correct numeric input.  
4. The form should handle navigation between records and items efficiently, ensuring that the user can move up and down through records and items without errors.  
5. The form should display error messages if there are issues populating the list of values or if required fields are not filled.  
  
Definition of Done:  
1. The form allows the user to input and view all specified attributes.  
2. The form validates the Cover Code and Cover Description against a predefined list of values.  
3. Numeric fields are validated for correct input.  
4. Navigation between records and items works smoothly.  
5. Error messages are displayed appropriately for any issues encountered.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- The form uses the following query to populate the list of values for Cover Code and Cover Description:  
 ```sql  
 SELECT COVER\_DESCRIPTION, COVER\_CODE   
 FROM CFG\_V\_PROD\_COVERS\_API   
 WHERE PRODUCT\_ID = :PARAMETER.PRODUCT\_ID   
 AND PROD\_VERSION = 1;  
 ```

# Auto-populate and store premium term value

Type: COVERS\_HEALTH

Title: Auto-populate and store premium term value  
  
Acceptance Criteria:  
1. When the user navigates to the premium term field, the system should automatically set its value to the value of the benefit term field.  
2. If the current record is the first record in the dataset, the system should store the value of the premium term field in a global variable.  
  
Definition of Done:  
- The premium term field is automatically populated with the benefit term value upon navigation.  
- The premium term value is stored in a global variable when the current record is the first in the dataset.  
- The functionality is tested and verified to work as expected without any Oracle Forms-specific terminology or dependencies.

# Store Benefit Term Globally on Item Validation

Type: COVERS\_HEALTH

Title: Store Benefit Term Globally on Item Validation  
  
Acceptance Criteria:  
- When the item is validated and the cursor is on the first record, the benefit term value should be stored in a global variable.  
- The global variable should be updated with the current benefit term value from the item.  
  
Definition of Done:  
- The benefit term value is successfully stored in a global variable when the item is validated.  
- The global variable is updated only when the cursor is on the first 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 operations.

# Validate Entry Age and Calculate Terms for Health Coverage

Type: COVERS\_HEALTH

Title: Validate Entry Age and Calculate Terms for Health Coverage  
  
Acceptance Criteria:  
1. When the user enters the entry age for the first record, the system should store this age in a global variable and update the main cover entry age.  
2. If the product is a child care product and the user is on the second record, the system should update the child care entry age.  
3. For child care products, the system should calculate the benefit term as the difference between the package maturity date and the entry age, and the premium term as 18 minus the entry age.  
4. The calculated benefit and premium terms should be updated in the respective fields.  
  
Definition of Done:  
- The system correctly validates and stores the entry age.  
- The system updates the main cover entry age and child care entry age as per the product type.  
- The benefit and premium terms are accurately calculated and displayed for child care products.  
- All calculations and updates are performed without any errors.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided logic does not include direct database CRUD operations.

# Health Cover Code Input Field

Type: COVERS\_HEALTH

Title: Health Cover Code Input Field  
  
Acceptance Criteria:  
- The text field should accept a maximum of 10 characters.  
- The input should be automatically converted to uppercase.  
- The text field should not have horizontal or vertical scrollbars.  
- The text field should not skip to the next field automatically after input.  
- The text field should not validate the input against a list.  
- The text field should not be linked to any database item.  
- The text field should not have a LOV (List of Values) associated with it.  
- The text field should not trigger any specific action upon validation.  
  
Definition of Done:  
- The text field is implemented and meets all the acceptance criteria.  
- The text field is tested to ensure it converts input to uppercase and adheres to the specified constraints.  
- The feature is reviewed and approved by stakeholders.

# Validate and Update Sum Insured for Whole Cover

Type: COVERS\_HEALTH

Title: Validate and Update Sum Insured for Whole Cover  
  
Acceptance Criteria:  
1. When the item is validated:  
 - If the cursor is on the first record, the system should update the global sum insured and the main cover sum insured with the value from the current cover.  
 - The system should disable the commit and exit buttons on the toolbar.  
  
2. When navigating to the next item:  
 - If the cover code is 'R014A01', the system should store the current sum insured value and move to the next record.  
 - If the next record's cover code is 'R015A01', the system should calculate the sum insured as `(stored value / 30000) 1250` and update the current cover with this value.  
 - The system should then move back to the previous record.  
  
Definition of Done:  
- The system correctly updates the global and main cover sum insured values when the item is validated.  
- The commit and exit buttons are disabled as specified.  
- The system correctly navigates and updates the sum insured based on the cover codes 'R014A01' and 'R015A01'.  
- All calculations and updates are performed without errors, and the system behaves as expected during navigation and validation.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Not applicable as the provided logic involves form-specific operations and does not directly translate to standalone SQL queries.

# Manage Document Details in AML Module

Type: AML

Title: Manage Document Details in AML Module  
  
Acceptance Criteria:  
1. The system should allow users to view and edit the following document details:  
 - Document Description  
 - Document Type  
 - Mandatory Flag  
 - Value (restricted to uppercase)  
 - Proof Type (with a list of values)  
 - Document ID (restricted to uppercase)  
 - Expiry Date (formatted as DD/MM/RRRR)  
 - Old Policy Reference  
 - Proof Description (with a list of values)  
 - Document Remarks  
  
2. The system should provide a button labeled "Populate AML Details" to automatically populate AML-related information.  
  
3. The system should include a checkbox labeled "Edit AML Details" to enable or disable editing of AML details.  
  
4. The system should display a message field for any relevant information or alerts.  
  
5. The system should fetch proof type and description from the database where the active flag is 'Y'.  
  
Definition of Done:  
- The user can view and edit all specified document details.  
- The "Populate AML Details" button functions correctly and populates relevant information.  
- The "Edit AML Details" checkbox enables or disables editing as expected.  
- The message field displays relevant information or alerts.  
- Proof type and description are fetched from the database based on the active flag.  
  
DB queries for Table reference CRUD operations only (With Usage):  
```sql  
SELECT DISTINCT Proof\_type, proof\_desc, display\_message  
FROM azbj\_aml\_proof\_master  
WHERE active\_flag = 'Y';  
```

# Implement Document Remarks Field in AML Section

Type: AML

Title: Implement Document Remarks Field in AML Section  
  
Acceptance Criteria:  
1. The "Document Remarks" field should allow a maximum of 300 characters.  
2. The field should be positioned appropriately within the AML section and should be clearly labeled as "Document Remarks".  
3. When the user navigates away from the "Document Remarks" field, the focus should automatically move to the "BI Number" field in the Agents section.  
  
Definition of Done:  
1. The "Document Remarks" field is implemented and visible in the AML section.  
2. The field accepts up to 300 characters.  
3. The field is labeled correctly and positioned as specified.  
4. The navigation logic is implemented such that moving away from the "Document Remarks" field takes the user to the "BI Number" field in the Agents section.  
5. All changes are tested and verified to ensure they meet the acceptance criteria.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include any specific database queries.

# Control Editability of VALUE Field Based on Document Type and Checkbox Value

Type: AML

Title: Control Editability of VALUE Field Based on Document Type and Checkbox Value  
  
Acceptance Criteria:  
1. If the document type is one of the following: 'LEGAL\_NAME', 'PERMANENT\_ADDRESS', 'CURRENT\_ADDRESS', 'SOURCE\_OF\_FUNDS3', 'COMPANY\_NAME', 'COMPANY\_ADDRESS', 'PARTNERSHIP\_FIRM\_NAME', 'PARTNERSHIP\_FIRM\_ADDRESS', 'TRUSTEES\_NAME', 'TRUSTEES\_ADDRESSES', then the "VALUE" field should be non-editable.  
2. If the checkbox value for 'CHK\_EDIT\_AML' is 'Y', then the "VALUE" field should be editable.  
3. In all other cases, the "VALUE" field should be non-editable.  
  
Definition of Done:  
- The "VALUE" field's editability is correctly controlled based on the document type and checkbox value.  
- The system has been tested to ensure that the "VALUE" field behaves as expected in all specified scenarios.  
- The changes have been reviewed and approved by the relevant stakeholders.

# Edit AML Details via Checkbox

Type: AML

Title: Edit AML Details via Checkbox  
  
Acceptance Criteria:  
1. When the checkbox is checked, the system should enable the relevant fields for editing AML details.  
2. The system should validate AML details using the `Aml\_validate` function, which checks various conditions such as product definitions and total premium amounts.  
3. The system should populate AML details based on the selected product and partner type, including document type, description, and mandatory flag.  
4. The system should automatically fill in specific fields like name, address, and annual income based on the document type.  
5. If the total premium is less than or equal to 10,000, the system should display a message indicating that AML details are not required.  
  
Definition of Done:  
- The checkbox functionality is implemented and tested.  
- Relevant fields are enabled or disabled based on the checkbox state.  
- AML details are validated and populated correctly according to the specified conditions.  
- Automated messages and field population work as expected.  
- All acceptance criteria are met, and the feature is tested and verified by QA.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- No specific DB queries provided in the XML content for CRUD operations.

# Navigate to bi\_no field from Old Policy Ref

Type: AML

Title: Navigate to bi\_no field from Old Policy Ref  
  
Acceptance Criteria:  
- When the user inputs the old policy reference number and presses the designated key to move to the next item, the system should automatically navigate to the "bi\_no" field in the "Agents" section.  
  
Definition of Done:  
- The old policy reference number field is present and functional.  
- The system successfully navigates to the "bi\_no" field in the "Agents" section upon user action.  
- The navigation logic is tested and confirmed to work as expected.

# Populate AML Details for Selected Insured Person and Policy Holder

Type: AML

Detailed description: As a user, I want to populate AML (Anti-Money Laundering) details for a selected insured person and policy holder, so that I can view and verify the relevant AML information associated with a policy.  
  
Acceptance criteria:  
1. When the "Populate AML Details" button is pressed, the system should check if the `v\_phonepe\_flag` is set to 'N'.  
2. If the `v\_phonepe\_flag` is 'N', the system should execute the procedure to populate AML details.  
3. The procedure should:  
 - Clear any existing data in the `PROOF\_RG` record group.  
 - Populate the `PROOF\_RG` record group with active proof types and descriptions from the `azbj\_aml\_proof\_master` table.  
 - Check if the `Ip\_part\_id` of the insured person is not null.  
 - Retrieve the `contract\_id` for the insured person based on specific conditions.  
 - Navigate to the AML block and fetch the first record.  
 - Count the number of AML records associated with the retrieved `contract\_id`.  
 - Retrieve the `policy\_ref` for the `contract\_id`.  
 - If AML records are found, loop through each record and populate the AML block fields with the corresponding data.  
 - If no AML records are found, display an error message indicating that no AML details are present for the policy.  
 - If the `Ip\_part\_id` is null, display an error message indicating that the insured person and policy holder must be selected.  
  
Definition of Done:  
- The "Populate AML Details" button should trigger the procedure to populate AML details as per the acceptance criteria.  
- The system should display appropriate error messages when no AML details are found or when the insured person and policy holder are not selected.  
- The AML block fields should be populated with the correct data from the database.  
- The "Populate AML Details" button should be disabled after the AML details are populated.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- `select distinct Proof\_type, proof\_desc, display\_message from azbj\_aml\_proof\_master where active\_flag='Y'`  
- `SELECT x.contract\_id INTO v\_contract\_id FROM (SELECT a.contract\_id, appln\_sign\_date FROM azbj\_policy\_contract\_ext a, ocp\_interested\_parties b WHERE a.contract\_id = b.contract\_id AND partner\_id = CASE WHEN :control.cn\_product\_id = 315 THEN :insured\_person.ip\_part\_id ELSE :policy\_holder.ph\_part\_id END AND top\_indicator = 'Y' AND a.action\_code <> 'D' AND b.action\_code <> 'D' AND ip\_no = 1 ORDER BY appln\_sign\_date DESC) x WHERE ROWNUM < 2`  
- `select count() into v\_cnt from azbj\_aml\_nb\_records\_NEW where contract\_id=v\_contract\_id`  
- `select policy\_ref into v\_policy\_ref from Ocp\_policy\_bases where contract\_id= v\_contract\_id AND top\_indicator='Y' AND action\_code<>'D'`  
- `select B.DOCUMENT\_TYPE\_SCREEN\_VALUE, A.PROOF\_TYPE, A.PARTNER\_TYPE, OTHERS, VALUE, a.document\_type, C.PROOF\_DESC, Expiry\_date, a.document\_remarks from azbj\_aml\_nb\_records\_NEW A, azbj\_aml\_constants B, azbj\_aml\_proof\_master c WHERE A.DOCUMENT\_TYPE=B.DOCUMENT\_TYPE AND A.PARTNER\_TYPE=B.PARTNER\_TYPE AND B.DOCUMENT\_TYPE= C.DOCUMENT\_TYPE AND A.PROOF\_TYPE= C.PROOF\_TYPE AND A.PARTNER\_TYPE=C.PARTNER\_TYPE AND CONTRACT\_ID=v\_contract\_id order by sort\_no desc`

# Proof Type Selection and Validation

Type: AML

Title: Proof Type Selection and Validation  
  
Acceptance Criteria:  
1. When the user double-clicks on the proof type field, the system should display a list of valid proof types based on the document type and partner type.  
2. If the document type is not null, the system should query the `azbj\_aml\_proof\_master` table to fetch distinct proof types, descriptions, and display messages where the partner type and document type match the user's input and the active flag is 'Y'.  
3. If the proof type is 'DL' (Driving License), the system should fetch the proof details from the `azbj\_proof\_valid\_details` table based on the application number and other criteria.  
4. If the proof type is 'PP' (Passport), the system should fetch the passport details from the `azbj\_passport\_valid\_det` table based on the application number and other criteria.  
5. If the proof type is 'VI' (Voter ID), the system should fetch the age proof details from the `azbj\_proposal\_appln\_det` table based on the application number and other criteria.  
6. If the proof type is 'AC' or 'ACS' (Aadhar Card), the system should fetch the Aadhar number from the `azbj\_partner\_extn` table based on the partner ID and other criteria.  
7. If the proof type is 'PC' (PAN Card) and the PAN status is 'Valid PAN', the system should fetch the PAN card details from the `AGENTS` table.  
8. The system should handle exceptions and display appropriate error messages if any issues occur during the data fetching process.  
  
Definition of Done:  
- The proof type field should display a list of valid proof types when double-clicked.  
- The system should validate the selected proof type based on the document type and partner type.  
- Additional fields such as proof description, expiry date, and others should be populated based on the selected proof type.  
- Appropriate error messages should be displayed if any issues occur during the data fetching process.  
- The functionality should be tested and verified to ensure it meets the acceptance criteria.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Query to fetch proof types based on document type and partner type:  
 ```sql  
 SELECT DISTINCT Proof\_type, proof\_desc, display\_message  
 FROM azbj\_aml\_proof\_master  
 WHERE partner\_type = :control.cn\_partner\_type  
 AND document\_type = :AML.document\_type  
 AND active\_flag = 'Y';  
 ```  
  
- Query to fetch proof details for Driving License:  
 ```sql  
 SELECT proof\_type, proof\_id, proof\_expiry\_date  
 INTO :aml.proof\_desc, :aml.OTHERS, :aml.expiry\_date  
 FROM customer.azbj\_proof\_valid\_details  
 WHERE appln\_no = TO\_CHAR(NVL(:insured\_person.ip\_verf\_no, :insured\_person.ip\_sign\_card\_no))  
 AND proof\_type = 'DRIVING LICENSE'  
 AND top\_indicator = 'Y'  
 AND create\_date = (SELECT MAX(create\_date)  
 FROM azbj\_proof\_valid\_details  
 WHERE appln\_no = TO\_CHAR(NVL(:insured\_person.ip\_verf\_no, :insured\_person.ip\_sign\_card\_no))  
 AND proof\_type = 'DRIVING LICENSE'  
 AND top\_indicator = 'Y')  
 AND partner\_type = CASE  
 WHEN :policy\_holder.ph\_equal\_ins = 1 THEN 'IP'  
 ELSE 'PH'  
 END;  
 ```  
  
- Query to fetch passport details:  
 ```sql  
 SELECT passport\_no, pass\_expiry\_date  
 INTO :aml.OTHERS, :aml.expiry\_date  
 FROM customer.azbj\_passport\_valid\_det  
 WHERE appln\_no = TO\_CHAR(NVL(:insured\_person.ip\_verf\_no, :insured\_person.ip\_sign\_card\_no))  
 AND top\_indicator = 'Y'  
 AND create\_date = (SELECT MAX(create\_date)  
 FROM azbj\_passport\_valid\_det  
 WHERE appln\_no = TO\_CHAR(NVL(:insured\_person.ip\_verf\_no, :insured\_person.ip\_sign\_card\_no))  
 AND top\_indicator = 'Y')  
 AND partner\_type = CASE  
 WHEN :policy\_holder.ph\_equal\_ins = 1 THEN 'IP'  
 ELSE 'PH'  
 END;  
 ```  
  
- Query to fetch age proof details for Voter ID:  
 ```sql  
 SELECT NVL(ip\_age\_proof\_id, ph\_age\_proof\_id), NVL(ip\_age\_proof, ph\_age\_proof)  
 INTO :aml.OTHERS, v\_proof  
 FROM customer.azbj\_proposal\_appln\_det  
 WHERE appln\_no = TO\_CHAR(NVL(:insured\_person.ip\_verf\_no, :insured\_person.ip\_sign\_card\_no))  
 AND de\_flag = 'D2'  
 AND (ip\_age\_proof = 'VI' OR ph\_age\_proof = 'VI');  
 ```  
  
- Query to fetch Aadhar number:  
 ```sql  
 SELECT aadhar\_no  
 INTO :aml.OTHERS  
 FROM azbj\_partner\_extn  
 WHERE part\_id = NVL(:insured\_person.ip\_part\_id, :policy\_holder.ph\_part\_id)  
 AND aadhar\_no IS NOT NULL;  
 ```  
  
- Query to fetch PAN card details:  
 ```sql  
 SELECT :AGENTS.PAN\_CARD  
 INTO :AML.OTHERS  
 FROM AGENTS  
 WHERE :AGENTS.PAN\_STATUS = 'Valid PAN'  
 AND :AML.PROOF\_TYPE = 'PC';  
 ```

# Validate and Manage Document ID for AML Section

Type: AML

User Story: Validate and Manage Document ID for AML Section  
  
Detailed Description:  
As a user, I need the system to validate and manage the Document ID field in the AML section to ensure that the data entered is accurate and meets the required format and conditions. This includes enabling or disabling the field based on specific conditions and validating the input against predefined rules.  
  
Acceptance Criteria:  
1. The Document ID field should be enabled or disabled based on the value of the `CHK\_EDIT\_AML` field:  
 - If `CHK\_EDIT\_AML` is 'Y', the Document ID field should be enabled for both update and insert operations.  
 - If `CHK\_EDIT\_AML` is not 'Y', the Document ID field should be disabled for both update and insert operations.  
  
2. When the user navigates to the next item:  
 - If the `DOCUMENT\_DESC` is 'IDENTITY PROOF' and `PROOF\_TYPE` is 'PC', the system should validate the Document ID:  
 - The Document ID must be 10 characters long.  
 - The Document ID must match the PAN card format (e.g., '^[A-Z]{3}[C,P,H,F,A,T,B,L,J,G][A-Z][0-9]{4}[A-Z]').  
 - If the Document ID is invalid, an error message should be displayed, and the field should be cleared.  
 - If the Document ID is valid, it should be copied to the `PAN\_CARD` field if it is empty or prompt the user to overwrite if it differs from the existing value.  
  
Definition of Done:  
- The Document ID field is correctly enabled or disabled based on the `CHK\_EDIT\_AML` field.  
- The Document ID is validated according to the specified rules when navigating to the next item.  
- Appropriate error messages are displayed for invalid Document IDs.  
- The Document ID is correctly copied to the `PAN\_CARD` field or prompts the user to overwrite if necessary.

# Manage Policy Members

Type: POLICY\_MEMBER

Detailed description: As a user, I want to manage policy members within the system, including adding, deleting, and updating member details, so that I can maintain accurate and up-to-date information for each policy member.  
  
Acceptance criteria:  
1. The system should allow the user to add a new policy member by clicking the "Add Member" button.  
2. The system should allow the user to delete an existing policy member by clicking the "Delete Member" button.  
3. The system should enable or disable the "Calculate SA" checkbox based on specific conditions:  
 - If the premium received is 5000 and the agent is from the corporate channel, the checkbox should be enabled.  
 - Otherwise, the checkbox should be disabled.  
4. The system should automatically set the IP number and cover code for each policy member based on their relationship (e.g., spouse, wife, husband) and other criteria.  
5. The system should validate the member details when the "Validate" button is clicked.  
6. The system should allow the user to decline a member by clicking the "Decline Member" button.  
7. The system should allow the user to load member details by clicking the "Member Loading" button.  
  
Definition of Done:  
- The user can successfully add, delete, and update policy member details.  
- The "Calculate SA" checkbox functionality works as per the specified conditions.  
- The IP number and cover code are correctly assigned to each policy member.  
- The validation process for member details is functional.  
- The "Decline Member" and "Member Loading" buttons perform their respective actions correctly.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- The following query is used to fetch age proof types from the database:  
 ```sql  
 SELECT PROOF\_TYPE, PROOF\_DESC  
 FROM azbj\_aml\_requirements  
 WHERE DOCUMENT\_TYPE = 'AGE\_PROOF'  
 AND ENABLED\_FLAG = 'Y'  
 ORDER BY PROOF\_TYPE;  
 ```

# Validate Policy Member Information

Type: POLICY\_MEMBER

Title: Validate Policy Member Information  
  
Acceptance Criteria:  
1. When the "Validate" button is pressed, the system should execute the logic to populate the partner list.  
2. After the validation process, the focus should automatically move to the next item, specifically the "Annual Income" field.  
  
Definition of Done:  
- The "Validate" button is functional and triggers the validation process.  
- The partner list is populated correctly upon validation.  
- The focus moves to the "Annual Income" field after the validation is complete.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Not applicable as the provided XML content does not include any specific database queries.

# Automatic Cover Code Update and Navigation on Relation Selection

Type: POLICY\_MEMBER

Title: Automatic Cover Code Update and Navigation on Relation Selection  
  
Acceptance Criteria:  
1. When a relation is selected from the list, the system should call a function to retrieve the appropriate cover code based on the selected relation and the current record.  
2. The retrieved cover code should be assigned to the cover code field.  
3. The system should then automatically navigate to the sum assured field.  
  
Definition of Done:  
- The system correctly updates the cover code based on the selected relation.  
- The system navigates to the sum assured field after updating the cover code.  
- 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 XML content does not include any direct database queries.

# Navigation and Validation for PM\_AGE\_PRF\_ID Field

Type: POLICY\_MEMBER

Detailed description: As a user, I want to navigate to the next field, "PM\_RELATION," after entering data in the "PM\_AGE\_PRF\_ID" field, so that I can continue filling out the form without interruption.  
  
Acceptance criteria:  
1. When the user completes data entry in the "PM\_AGE\_PRF\_ID" field and presses the tab key, the cursor should automatically move to the "PM\_RELATION" field.  
2. The "PM\_AGE\_PRF\_ID" field should accept a maximum of 20 characters and convert all input to uppercase.  
3. The "PM\_AGE\_PRF\_ID" field should be validated against a predefined list to ensure the entered value is valid.  
  
Definition of Done:  
1. The navigation from "PM\_AGE\_PRF\_ID" to "PM\_RELATION" works seamlessly.  
2. The "PM\_AGE\_PRF\_ID" field enforces the maximum length and case restriction.  
3. The validation logic for the "PM\_AGE\_PRF\_ID" field is implemented and functioning correctly.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include any specific database queries.

# Validate and Set Default Weight Change Value for Policy Member

Type: POLICY\_MEMBER

Title: Validate and Set Default Weight Change Value for Policy Member  
  
Acceptance Criteria:  
1. If the weight change value for the insured person is null, it should be automatically set to 0.  
2. If the global loading status is 'F', the form status should be updated to 'Y'.  
  
Definition of Done:  
- The weight change value is validated and set to 0 if it is null.  
- The form status is updated to 'Y' when the global loading status is 'F'.  
- The functionality is tested and confirmed to work as expected.  
  
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.

# Age Proof Selection and Validation

Type: POLICY\_MEMBER

Title: Age Proof Selection and Validation  
  
Acceptance Criteria:  
1. When the user double-clicks on the age proof field, the system should display a list of valid age proofs.  
2. Upon selection, the system should fetch the corresponding age proof details from the database and populate the relevant fields.  
3. If no matching age proof is found, the system should display a message indicating "NA" and clear the age proof value field.  
4. The system should handle any other exceptions by displaying "NA" and clearing the age proof value field.  
  
Definition of Done:  
- The age proof field should allow the user to select from a list of valid age proofs.  
- The system should correctly fetch and display the age proof details based on the user's selection.  
- Appropriate messages should be displayed when no matching age proof is found or other exceptions occur.  
- The functionality should be tested and verified to ensure it meets the acceptance criteria.  
  
DB queries for Table reference CRUD operations only (With Usage):  
```sql  
SELECT PROOF\_TYPE, PROOF\_DESC  
FROM azbj\_aml\_requirements  
WHERE DOCUMENT\_TYPE = 'AGE\_PROOF'  
AND ENABLED\_FLAG = 'Y'  
ORDER BY PROOF\_TYPE;  
```  
- This query is used to fetch the list of valid age proofs for the user to select from.

# Delete Member Functionality

Type: POLICY\_MEMBER

Title: Delete Member Functionality  
  
Acceptance Criteria:  
1. When the delete member button is pressed, the system should:  
 - Identify the member to be deleted based on the cover code.  
 - Navigate to the "Covers" section and delete records matching the member's cover code.  
 - Navigate to the "Further Requirements" section and delete records matching the member's IP number and specific test number.  
 - Navigate to the "FCF Loading Details" section and delete records matching the member's part ID.  
 - If the control flag indicates continuation, navigate to the "BBU Questions" section and delete records matching the member's part ID.  
 - Finally, delete the member record from the "Policy Member" section.  
2. If the member to be deleted is a spouse, update the spouse presence flag accordingly.  
3. Display an alert message confirming the deletion of the member and related questions.  
4. If no member is selected, display a warning message prompting the user to select a member.  
5. Ensure that the system handles any errors during the deletion process gracefully and displays an appropriate error message.  
  
Definition of Done:  
- The delete member functionality is implemented and tested.  
- All related records in the "Covers," "Further Requirements," "FCF Loading Details," and "BBU Questions" sections are deleted as expected.  
- Appropriate messages are displayed to the user during the process.  
- The system handles errors gracefully and provides meaningful feedback to the user.  
- 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 contains Oracle Forms-specific constructs that cannot be executed directly in the database without modification.

# Add Coverage Details for Policy Member

Type: POLICY\_MEMBER

Title: Add Coverage Details for Policy Member  
  
Acceptance Criteria:  
1. When the user selects the "Add Covers" checkbox, the system should check if the member relation is selected.  
2. If the member relation is not selected, the system should display a warning message: "Please select the member relation...!".  
3. If the member relation is selected, the system should assign the coverage details (cover code, sum assured, age, and IP number) to the policy member.  
  
Definition of Done:  
- The "Add Covers" checkbox functionality is implemented.  
- The system checks for the member relation and displays a warning if it is not selected.  
- The system assigns the coverage details to the policy member when the checkbox is selected and the member relation is present.  
- 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 XML content does not include any direct database queries.

# Automatic Sum Assured Calculation for Policy Members

Type: POLICY\_MEMBER

Detailed description: As a user, I want the system to automatically calculate the sum assured (SA) for policy members when the "Calculate SA" checkbox is checked, so that I can ensure accurate premium calculations based on the provided family member details.  
  
Acceptance criteria:  
1. When the "Calculate SA" checkbox is checked:  
 - The system should iterate through all policy members and collect their details such as IP number, age proof, relation to insured, name, part ID, date of birth, sex, height, weight, address ID, mail ID, marital status, age, and age proof ID.  
 - The system should call the `FCF\_CALC\_SA\_FROM\_PREM` function with the collected family member details, insured person's date of birth, premium amount, sum assured, and the date received.  
 - If the sum assured is null, the system should prompt the user to enter the correct number of family members along with their relations or check the age of the insured person.  
 - If the sum assured is not null, the system should navigate to the sum assured field for further actions.  
  
2. Exception Handling:  
 - If any error occurs during the process, the system should log the error and set the sum assured to 0.  
  
Definition of Done:  
- The system correctly calculates the sum assured when the "Calculate SA" checkbox is checked.  
- The system handles exceptions gracefully and logs errors appropriately.  
- The user is prompted with appropriate messages if the sum assured is null.  
- The system navigates to the sum assured field if the calculation is successful.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Not applicable as the provided logic involves procedural operations and function calls rather than direct CRUD operations on the database.

# Decline Member Functionality

Type: POLICY\_MEMBER

Detailed description: As a user, I want to be able to decline a member from the policy member list, so that the member is removed from the relevant records and the system reflects the updated status.  
  
Acceptance criteria:  
1. When the "Decline Member" button is pressed, the system should prompt the user with a confirmation message asking, "Do You Want to Decline the Member?".  
2. If the user confirms, the system should:  
 - Disable the "Commit Form" button.  
 - Check if the selected policy member's name is not null.  
 - Display an alert message stating, "Deleting the Member. Question will be deleted for this Member."  
 - Navigate to the "FCF\_LOADING\_DETAILS" section and delete any records matching the policy member's part ID.  
 - If the control flag `CN\_WIP\_CONTINUE` is set to 'T', the system should:  
 - Navigate to the "FCF\_DECLINE\_MEMBER" section.  
 - Create a new record with the policy member's details and increment the sequence number.  
 - Navigate to the "azbj\_bbu\_questions" section and delete any records matching the policy member's part ID.  
 - Finally, delete the policy member record from the "POLICY\_MEMBER" section.  
3. If the policy member's name is null, the system should display a warning message, "Please select member".  
  
Definition of Done:  
- The "Decline Member" button functionality is implemented and tested.  
- The system correctly prompts the user for confirmation and processes the decline operation as per the acceptance criteria.  
- All related records are updated or deleted as specified.  
- Appropriate messages are displayed to the user at each step.  
- The feature is tested and verified to work without any Oracle Forms-specific dependencies.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- No direct DB queries provided in the XML content.

# Manage Lateral Shift Status of Policy Members

Type: POLICY\_MEMBER

Detailed description: As a user, I want to manage the lateral shift status of policy members so that I can ensure accurate tracking and updates of their shift information.  
  
Acceptance criteria:  
1. When the lateral shift checkbox is changed, the system should:  
 - Navigate to the lateral shift records.  
 - Check if the current policy member's partner ID matches any existing lateral shift records.  
 - If a match is found and the checkbox is checked, set a flag to 'Y'.  
 - If a match is found and the checkbox is unchecked, set a flag to 'Y' and delete the corresponding lateral shift record.  
 - If no match is found, navigate to the next record.  
2. If no matching record is found after checking all records:  
 - Navigate to the lateral shift block.  
 - If the partner ID is not null, move to the next record.  
 - Update the lateral shift record with the current policy member's partner ID, IP number, and name.  
  
Definition of Done:  
- The lateral shift status of policy members is accurately tracked and updated based on the checkbox status.  
- The system correctly navigates through records and performs necessary updates or deletions.  
- The functionality is tested and verified to ensure it meets the acceptance criteria.  
  
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.

# Transfer Member Loading Details from Policy Member to Loading Details Section

Type: POLICY\_MEMBER

Title: Transfer Member Loading Details from Policy Member to Loading Details Section  
  
Acceptance Criteria:  
1. When the "Member Loading" button is pressed, the system should navigate to the loading details section and check the number of records.  
2. The system should then navigate back to the policy member section and check the number of records there.  
3. For each policy member record, the system should:  
 - Navigate to the loading details section and check if the member's details are already present.  
 - If the member's details are not present and the member's relation is not 'Son', 'Daughter', or 'Child', the system should:  
 - Copy the member's IP number, name, part ID, and relation to the loading details section.  
 - Set the cover code to 'L066A01' and the cover description to 'Bajaj Allianz Family CareFirst Main Benefit'.  
 - Move to the next record in the loading details section.  
4. The system should ensure that all policy member records are processed and the loading details section is updated accordingly.  
  
Definition of Done:  
- The member loading details are accurately transferred from the policy member details to the loading details section.  
- The system correctly identifies and skips members with relations 'Son', 'Daughter', or 'Child'.  
- The cover code and description are correctly set for each new record in the loading details section.  
- The process is completed without errors, and the user can verify the updated details in the loading details section.  
  
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.

# Validate and Calculate Policy Member Weight

Type: POLICY\_MEMBER

User Story: Validate and Calculate Policy Member Weight  
  
Detailed Description:  
As a user, I want the system to validate and calculate the weight of a policy member to ensure that the weight entered is valid and to compute the Body Mass Index (BMI) based on the height and weight of the policy member.  
  
Acceptance Criteria:  
1. The system should validate the weight of the policy member when the weight is entered.  
2. If the weight is less than or equal to zero, the system should display an error message: "Please Select a Valid Weight for the person".  
3. The system should calculate the BMI of the policy member if the height is greater than zero. The BMI is calculated using the formula:   
 \[  
 \text{BMI} = \frac{\text{weight}}{(\text{height} / 100)^2}  
 \]  
4. If the policy member's height is greater than zero, the system should compute the BMI and store it in the appropriate field.  
5. The system should disable certain form actions if specific conditions are met, such as the presence of a premium frequency.  
  
Definition of Done:  
- The weight validation logic is implemented and tested.  
- The BMI calculation logic is implemented and tested.  
- Error messages are displayed correctly when invalid weight is entered.  
- Form actions are disabled/enabled based on the specified conditions.  
- All acceptance criteria are met and verified through testing.  
  
DB Queries for Table Reference CRUD Operations:  
- Not applicable as the provided XML content does not include direct database queries for CRUD operations.  
  
Explanation of Oracle Form Logics:  
- The `WHEN-VALIDATE-ITEM` trigger is used to validate the weight of the policy member and calculate the BMI.  
- The `KEY-NEXT-ITEM` trigger is used to navigate to the next item in the form.  
- The logic includes conditions to check the product definition and package code to determine if the weight validation and BMI calculation should be performed.  
- The system disables certain form actions (e.g., commit and exit) if specific conditions are met, such as the presence of a premium frequency.

# Validate Age Proof for Policy Members

Type: POLICY\_MEMBER

User Story: Validate Age Proof for Policy Members  
  
Detailed Description:  
As a system, I need to validate the age proof provided by policy members to ensure it meets the required criteria and to prevent the use of invalid documents.  
  
Acceptance Criteria:  
1. If the premium frequency is not null, disable the form's commit and exit buttons.  
2. If the age proof provided by the insured person is 'SYBM' and the agent code does not start with '522', display an error message stating that the Syndicate Bank BM Certificate is allowed only for Syndicate Bank.  
3. On double-clicking the age proof field, display a list of valid age proof types from the database.  
  
Definition of Done:  
- The system should disable the commit and exit buttons when the premium frequency is not null.  
- The system should display an error message if the age proof is 'SYBM' and the agent code does not start with '522'.  
- The system should display a list of valid age proof types when the age proof field is double-clicked.  
  
DB Queries for Table Reference CRUD Operations:  
```sql  
SELECT PROOF\_TYPE, PROOF\_DESC  
FROM azbj\_aml\_requirements  
WHERE DOCUMENT\_TYPE = 'AGE\_PROOF'  
AND ENABLED\_FLAG = 'Y'  
ORDER BY PROOF\_TYPE;  
```  
  
This query fetches the list of valid age proof types from the database to be displayed when the age proof field is double-clicked.

# Add a New Member to the Policy

Type: POLICY\_MEMBER

User Story: Add a New Member to the Policy  
  
Detailed Description:  
As a user, I want to add a new member to an existing policy so that the new member's details are included in the policy records. When the "Add Member" button is pressed, the system should execute a series of steps to validate and assign the appropriate details to the new member.  
  
Acceptance Criteria:  
1. When the "Add Member" button is pressed, the system should:  
 - Call a function to set the customer partner details.  
 - Assign an identification number to the new member.  
 - Validate the relationship of the new member (e.g., spouse, child).  
 - Assign the appropriate cover code based on the relationship.  
 - Mark the member as added to the policy.  
 - Ensure that the member's details are correctly updated in the system.  
  
2. The system should handle the following scenarios:  
 - If the new member is a spouse, assign a specific identification number and cover code.  
 - If the new member is not a spouse, assign a sequential identification number starting from 3.  
 - Ensure that the cover code is correctly assigned based on the member's relationship and sequence.  
  
3. The system should display appropriate messages if any errors occur during the process.  
  
Definition of Done:  
- The "Add Member" button functionality is implemented and tested.  
- The system correctly assigns identification numbers and cover codes to new members.  
- The system handles different relationships and updates the policy records accordingly.  
- Error handling and messaging are implemented and tested.  
- The functionality is reviewed and approved by stakeholders.  
  
DB Queries for Table Reference CRUD Operations:  
- Not applicable as the provided XML content does not include direct database queries.  
  
Explanation of Oracle Form Logic:  
- The logic involves setting customer partner details, assigning identification numbers, validating relationships, and updating cover codes. The system iterates through the policy members to ensure correct assignment and updates the records accordingly. Error handling is included to manage any issues that arise during the process.

# Validate BMI Input for Policy Members

Type: POLICY\_MEMBER

User Story: Validate BMI Input for Policy Members  
  
Detailed Description:  
As a user, I want to ensure that the BMI (Body Mass Index) input for policy members is validated according to specific business rules, so that only valid BMI values are accepted and processed.  
  
Acceptance Criteria:  
1. The system should validate the BMI input only if the global loading flag is set to 'F'.  
2. The form status should be updated to 'Y' if the global loading flag is 'F'.  
3. The BMI validation should check the product definition of the policy. If the product definition does not match specific pension-related products or package codes, and the pension flag is set to 0, the system should proceed with further validations.  
4. If the insured person's weight is not null and the product definition is not among specific group products, and certain flags are not set, the system should validate the weight.  
5. If the weight is less than or equal to 0, an error message should be displayed: "Please Select a Valid Weight for the person".  
6. If the premium frequency is not null, certain form buttons should be disabled to prevent further actions until the validation is complete.  
  
Definition of Done:  
- The BMI input validation logic is implemented and tested.  
- The system correctly updates the form status and displays appropriate error messages based on the validation rules.  
- The form buttons are disabled as specified when the premium frequency is not null.  
- All acceptance criteria are met, and the functionality is verified through unit and integration tests.  
  
DB Queries for Table Reference CRUD Operations:  
- Not applicable as the provided logic does not include direct CRUD operations on database tables.

# Validate Policy Member's Height

Type: POLICY\_MEMBER

User Story: Validate Policy Member's Height  
  
Detailed Description:  
As a user, I need to ensure that the height of a policy member is validated according to specific business rules, so that the data entered is accurate and consistent with the product requirements.  
  
Acceptance Criteria:  
1. The system should validate the height of the policy member when the height field is populated.  
2. The validation should check if the product definition of the policy is not one of the specified pension or special products.  
3. The validation should ensure that the package code does not match specific patterns related to 'FUTURE SECURE SIZE' or 'FUTURE SECURE II OPTION A'.  
4. The validation should confirm that the pension flag is set to 0.  
5. If the height is not null and the product definition is not one of the specified group or special products, and certain flags are not set, the system should check if the height is greater than 0.  
6. If the height is less than or equal to 0, an error message should be displayed: "Please Select a Valid Height for the Person".  
7. The system should update the form status to 'Y' if the global loading flag is set to 'F'.  
8. If the frequency premium is not null, the system should disable the commit and exit form buttons.  
  
Definition of Done:  
- The height validation logic is implemented and tested.  
- The system displays appropriate error messages when the height is invalid.  
- The form status and button properties are updated based on the validation results.  
- All specified business rules are adhered to during the validation process.  
  
DB Queries for Table Reference CRUD Operations:  
- Not applicable as the provided XML content does not include direct database queries for CRUD operations.  
  
Explanation of Oracle Form Logics:  
- The `WHEN-VALIDATE-ITEM` trigger is used to validate the height of the policy member based on various conditions.  
- The `KEY-NEXT-ITEM` trigger is used to navigate to the next item (weight) after the height is validated.  
- The form status and button properties are updated based on the validation results to ensure data integrity and proper user guidance.  
  
Please review the user story and provide feedback. Once approved, it will be saved.

# Manage Further Requirements for Tests

Type: TFURTHER\_REQ

Title: Manage Further Requirements for Tests  
  
Acceptance Criteria:  
1. The system should allow users to add new test requirements.  
2. The system should allow users to delete existing test requirements.  
3. The system should display a list of test requirements with the following details:  
 - Test Number  
 - Description  
 - Received Status (Y/N)  
 - Date Received  
 - IP Type  
 - Date Called  
4. The system should provide a checkbox to mark a test as received.  
5. The system should allow navigation through the list of test requirements using up and down keys.  
6. The system should validate the test number from a predefined list of values.  
7. The system should ensure that the received status is either 'Y' or 'N'.  
8. The system should display the received date in a specific format.  
9. The system should provide buttons for confirming actions (Ok) and navigating back (Back).  
  
Definition of Done:  
1. The functionality to add, delete, and view test requirements is implemented and tested.  
2. The list of test requirements displays all necessary details as specified.  
3. The received status checkbox works correctly and updates the status.  
4. Navigation through the list using up and down keys is functional.  
5. The test number validation from the list of values is implemented.  
6. The received status validation ensures only 'Y' or 'N' values.  
7. The received date is displayed in the correct format.  
8. The Ok and Back buttons perform their respective actions correctly.  
9. All acceptance criteria are met, and the feature is tested and verified by QA.  
  
DB queries for Table reference CRUD operations only(With Usage):  
```sql  
SELECT sys\_code AS DOC, UPPER(TRIM(sys\_desc)) AS DESCRIPTION  
FROM azbj\_system\_constants  
WHERE sys\_type = 'FR\_REQ'  
AND NVL(TO\_DATE(:susac.sa\_daterecd, 'DD/MM/RRRR'), pme\_api.opus\_date) BETWEEN start\_date AND NVL(end\_date, TO\_DATE('01-jan-3000', 'dd/mm/rrrr'))  
ORDER BY DESCRIPTION;  
```  
This query is used to fetch the list of valid test numbers and their descriptions for validation purposes.

# Automatically update received date based on requirement status

Type: TFURTHER\_REQ

Detailed description: As a user, I want to record whether a specific requirement has been received, so that the system can automatically update the received date and maintain the integrity of the data.  
  
Acceptance criteria:  
1. When the user marks the requirement as received ('Y'), the system should automatically populate the received date with the current date.  
2. If the requirement is not marked as received ('N'), the received date should be cleared, and a global flag should be set to indicate that the form is not clean.  
  
Definition of Done:  
- The user can mark a requirement as received or not received.  
- The system automatically updates the received date based on the user's input.  
- The global flag is correctly set when the requirement is not received.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided logic does not include direct database operations.

# Navigate Back to Further Requirements Section

Type: TFURTHER\_REQ

Title: Navigate Back to Further Requirements Section  
  
Acceptance Criteria:  
- When the 'Back' button is clicked, the system should automatically navigate to the 'Further Requirements' section.  
- The focus should be set to the 'Add Test' field within the 'Further Requirements' section after navigation.  
  
Definition of Done:  
- The 'Back' button is visible and clickable on the current screen.  
- Clicking the 'Back' button successfully navigates the user to the 'Further Requirements' section.  
- The 'Add Test' field within the 'Further Requirements' section is focused after navigation.  
- 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 XML content does not include any database queries.

# User can select test number from LOV and auto-populate date called field

Type: TFURTHER\_REQ

Title: User can select test number from LOV and auto-populate date called field  
  
Acceptance Criteria:  
1. When the user clicks on the test number field, a list of values (LOV) should be displayed, allowing the user to select a test number.  
2. When the user navigates to the test number field, the current date should be automatically populated in the date called field.  
3. The LOV should fetch data from the database where the system type is 'FR\_REQ' and the date is within the valid range.  
  
Definition of Done:  
1. The test number field should display a list of values when clicked.  
2. The date called field should automatically populate with the current date when the test number field is navigated to.  
3. The LOV should correctly fetch and display data based on the specified criteria.  
  
DB queries for Table reference CRUD operations only(With Usage):  
```sql  
SELECT sys\_code AS DOC, UPPER(TRIM(sys\_desc)) AS DESCRIPTION  
FROM azbj\_system\_constants  
WHERE sys\_type = 'FR\_REQ'  
AND NVL(TO\_DATE(:susac.sa\_daterecd, 'DD/MM/RRRR'), pme\_api.opus\_date) BETWEEN start\_date AND NVL(end\_date, TO\_DATE('01-jan-3000', 'dd/mm/rrrr'))  
ORDER BY DESCRIPTION;  
```

# Implement Insurance Policy Type Dropdown

Type: TFURTHER\_REQ

Title: Implement Insurance Policy Type Dropdown  
  
Acceptance Criteria:  
- The insurance policy type field should be a dropdown list with a maximum length of 2 characters.  
- The dropdown list should contain 10 predefined options.  
- The field should be positioned correctly on the form and should be keyboard navigable.  
- The input should be restricted to uppercase letters.  
- The field should display a prompt labeled "Life" above it.  
- The prompt should be bold and aligned to the center.  
- The field should not display horizontal or vertical scrollbars.  
  
Definition of Done:  
- The insurance policy type dropdown list is implemented and displays the correct options.  
- The field is correctly positioned and styled as per the requirements.  
- The field is functional and allows users to select an option from the list.  
- The prompt "Life" is displayed correctly above the field.  
- The field adheres to the specified input restrictions and navigability.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- No specific database queries are required for this user story as the field is not directly linked to a database item.

# Automatically record the date when a test is marked as received

Type: TFURTHER\_REQ

Title: Automatically record the date when a test is marked as received  
  
Acceptance Criteria:  
1. When the checkbox labeled "Test Received" is selected, the system should automatically populate the date field with the current date.  
2. If the checkbox is not selected, the date field should remain unchanged.  
  
Definition of Done:  
1. The checkbox labeled "Test Received" is present on the form.  
2. Selecting the checkbox automatically records the current date in the date field.  
3. The functionality is tested and verified to ensure the date is correctly recorded only when the checkbox is selected.  
  
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.

# Add Requirement Functionality

Type: TFURTHER\_REQ

Title: Add Requirement Functionality  
  
Acceptance Criteria:  
1. When the "Add Requirement" button is pressed, the system should enable the input field for the test number.  
2. The system should set the list of values (LOV) for the test number field to display available further requirements.  
3. The requirement type should be set to 'FR-REQ'.  
4. The system should navigate to the test number input field.  
5. The test number and test description fields should be cleared and ready for new input.  
6. If any error occurs during this process, an error message should be displayed to the user.  
  
Definition of Done:  
- The "Add Requirement" button is functional and performs the described actions.  
- The test number input field is enabled and linked to the correct LOV.  
- The requirement type is correctly set to 'FR-REQ'.  
- The system navigates to the test number input field and clears it along with the test description field.  
- Error handling is implemented, and appropriate error messages are displayed.  
  
DB queries for Table reference CRUD operations only (With Usage):  
```sql  
SELECT sys\_code AS DOC, UPPER(TRIM(sys\_desc)) AS DESCRIPTION  
FROM azbj\_system\_constants  
WHERE sys\_type = 'FR\_REQ'  
AND NVL(TO\_DATE(:susac.sa\_daterecd, 'DD/MM/RRRR'), pme\_api.opus\_date) BETWEEN start\_date AND NVL(end\_date, TO\_DATE('01-jan-3000', 'dd/mm/rrrr'))  
ORDER BY DESCRIPTION;  
```  
- This query is used to fetch the list of further requirements to be displayed in the LOV for the test number field.

# Delete Requirement with WIP Check

Type: TFURTHER\_REQ

Title: Delete Requirement with WIP Check  
  
Acceptance Criteria:  
1. When the delete button is pressed, the system should navigate to the 'further\_req' block.  
2. If the 'WIP Continue' flag is set to true, the system should check for the existence of a corresponding record in the 'WIP\_AZBJ\_MED\_UW' table using the 'TESTNO' field.  
3. If a corresponding record is found, it should be deleted from the 'WIP\_AZBJ\_MED\_UW' table.  
4. The requirement should then be deleted from the 'further\_req' block.  
  
Definition of Done:  
- The delete button should be functional and perform the described operations.  
- The system should handle the deletion of associated records in the WIP table based on the 'WIP Continue' flag.  
- The requirement should be removed from the list of further requirements.  
- The user should receive appropriate feedback if the deletion is successful or if any errors occur.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Check for the existence of a record in the 'WIP\_AZBJ\_MED\_UW' table:  
 ```sql  
 SELECT COUNT() INTO azbj\_mn\_ext\_row   
 FROM WIP\_AZBJ\_MED\_UW   
 WHERE TESTNO = :further\_req.fr\_TESTNO;  
 ```  
  
- Delete the record from the 'WIP\_AZBJ\_MED\_UW' table:  
 ```sql  
 DELETE FROM WIP\_AZBJ\_MED\_UW   
 WHERE TESTNO = :further\_req.fr\_TESTNO;  
 ```  
  
- Delete the requirement from the 'further\_req' block:  
 ```sql  
 DELETE FROM further\_req   
 WHERE fr\_TESTNO = :further\_req.fr\_TESTNO;  
 ```

# Synchronize Further Request Records

Type: TFURTHER\_REQ

Title: Synchronize Further Request Records  
  
Acceptance Criteria:  
1. When the 'Ok' button is pressed, the system should navigate to the 'TFurther Request' block and iterate through all records.  
2. For each record in the 'TFurther Request' block where the 'Select Request' field is marked as 'Y':  
 - The system should navigate to the 'Further Request' block and check if a corresponding record exists based on the 'Test Number'.  
 - If a corresponding record exists, the system should update the following fields in the 'Further Request' block:  
 - Date Called  
 - Result Received  
 - Date Received  
 - IP Type  
 - If no corresponding record exists, the system should create a new record in the 'Further Request' block and populate it with the data from the 'TFurther Request' block.  
3. After processing all records, the system should return to the 'Further Request' block and display the first record.  
  
Definition of Done:  
- The 'Ok' button functionality is implemented and tested.  
- The system correctly updates or creates records in the 'Further Request' block based on the 'TFurther Request' block.  
- The data synchronization between the 'TFurther Request' and 'Further Request' blocks is verified and validated.  
- The user is able to see the updated records in the 'Further Request' block after pressing the 'Ok' button.

# View and Manage Underwriting Comments

Type: UW\_COMMENTS

Detailed description: As a user, I want to view and manage underwriting comments, including the ability to see the user ID, date and time of the comment, and the comment text itself. Additionally, I want to have buttons to add new comments and refresh the comment list.  
  
Acceptance criteria:  
1. The user should be able to see the user ID and date & time of each comment, but these fields should be read-only.  
2. The user should be able to view the comment text, but not edit it.  
3. There should be a button labeled "Medical Underwriting" to add new comments.  
4. There should be a button labeled "Refresh" to refresh the list of comments.  
  
Definition of Done:  
1. The user interface displays the user ID, date & time, and comment text for each comment.  
2. The user ID and date & time fields are read-only.  
3. The comment text field is read-only.  
4. The "Medical Underwriting" button is functional and allows the user to add new comments.  
5. The "Refresh" button is functional and refreshes the list of comments.  
  
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.

# Medical Underwriting Requirements Management

Type: UW\_COMMENTS

Title: Medical Underwriting Requirements Management  
  
Acceptance Criteria:  
1. When the "Medical Underwriting" button is pressed, the system should:  
 - Retrieve the contract ID and policy reference from the control block and store them in global variables.  
 - Open the "Medical Underwriting" form.  
 - Retrieve the event number from the global variable and store it in the control block.  
 - For each test code and type 'F' in the "Underwriting Requirements Raised" table:  
 - Check if the test code already exists in the "Further Requirements" block.  
 - If it does not exist, create a new record in the "Further Requirements" block with the test code, description, IP type, date called, and other relevant details.  
 - For each test code and type 'M' in the "Underwriting Requirements Raised" table:  
 - Check if the test code already exists in the "Medical Underwriting" block.  
 - If it does not exist, create a new record in the "Medical Underwriting" block with the test code, description, IP type, date called, and other relevant details.  
 - Apply specific visual attributes to certain test codes in the "Further Requirements" block.  
  
Definition of Done:  
- The "Medical Underwriting" button functionality is implemented and tested.  
- The system correctly retrieves and stores contract and event information.  
- The system correctly processes and records test codes and details in the "Further Requirements" and "Medical Underwriting" blocks.  
- Specific visual attributes are applied to certain test codes as required.  
- All acceptance criteria are met, and the functionality is verified through testing.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Retrieve test codes and details from the "Underwriting Requirements Raised" table and the "System Constants" table based on contract ID, event number, and test type.  
- Insert new records into the "Further Requirements" and "Medical Underwriting" blocks with the relevant details.

# Refresh Comments Section

Type: UW\_COMMENTS

Title: Refresh Comments Section  
  
Acceptance Criteria:  
1. When the refresh button is pressed, the system should navigate to the comments section.  
2. The system should fetch and display the first record in the comments section.  
3. If the user ID starts with 'P00%', the system should:  
 - Retrieve all comments related to the contract ID.  
 - Populate the comments section with the user ID, comment date, and comments for each record.  
4. If the user's profile is not '1', the system should:  
 - Retrieve comments related to the contract ID where the user ID does not start with 'P00%' and the flag is not 'Y'.  
 - Populate the comments section with the user ID, comment date, and comments for each record.  
5. For all other cases, the system should:  
 - Retrieve comments related to the contract ID where the flag is 'N'.  
 - Populate the comments section with the user ID, comment date, and comments for each record.  
  
Definition of Done:  
- The refresh button should correctly navigate to the comments section.  
- The comments section should display the latest comments based on the specified conditions.  
- The data displayed should be accurate and reflect the latest information from the database.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Retrieve all comments related to the contract ID:  
 ```sql  
 SELECT FROM AZBJ\_UW\_COMMENTS WHERE contract\_id = :control.cn\_contract\_id;  
 ```  
- Retrieve comments where the user ID does not start with 'P00%' and the flag is not 'Y':  
 ```sql  
 SELECT EVENT\_NO, CONTRACT\_ID, POLICY\_NO, IP\_NO, MOVE\_CODE, POLICY\_STATUS, USER\_ID, COMMENT\_DATE, COMMENTS, USERROLE, OPINION, FLAG  
 FROM AZBJ\_UW\_COMMENTS  
 WHERE contract\_id = :control.cn\_contract\_id  
 AND 0 = (CASE WHEN (user\_id LIKE 'P00%' AND NVL(flag, 'N') = 'Y') THEN 1 ELSE 0 END);  
 ```  
- Retrieve comments where the flag is 'N':  
 ```sql  
 SELECT FROM AZBJ\_UW\_COMMENTS  
 WHERE contract\_id = :control.cn\_contract\_id  
 AND NVL(flag, 'N') = 'N';  
 ```

# Manage Lateral Shift Details in Insurance Policies

Type: LATERAL\_SHIFT

Title: Manage Lateral Shift Details in Insurance Policies  
  
Acceptance Criteria:  
1. The system should display a form with fields for entering and viewing the following details:  
 - IP Number  
 - Partner ID (read-only)  
 - Partner Name (read-only)  
 - Old Policy Reference  
 - Old Product (with a dropdown list for selection)  
 - Old Sum Assured  
 - Old Document Date  
 - Last Renewal Date  
 - Continuous Years, Months, and Days (with dropdown lists for selection)  
 - Claim-Free Flag (checkbox)  
 - Waiting Period  
 - Old Company (read-only)  
2. The "Save" button should be available to save the entered details.  
3. The "Old Product" field should provide a list of values (LOV) for selection, populated from the `azbj\_other\_insurance\_policy` table.  
4. The form should validate the data types for each field (e.g., dates, numbers, characters) and ensure that read-only fields cannot be edited.  
  
Definition of Done:  
- The form is implemented and displays all the required fields.  
- The "Save" button functionality is implemented and tested.  
- The "Old Product" field correctly displays a list of values from the `azbj\_other\_insurance\_policy` table.  
- Data validation is implemented for each field.  
- The form is tested and verified to ensure that read-only fields cannot be edited.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- To populate the "Old Product" field:  
 ```sql  
 SELECT COMPANY\_NAME, POLICY\_NAME FROM azbj\_other\_insurance\_policy;  
 ```

# Save Button Functionality for Lateral Shift Data

Type: LATERAL\_SHIFT

Title: Save Button Functionality for Lateral Shift Data  
  
Acceptance Criteria:  
1. When the "Save" button is pressed, the system should navigate to the Lateral Shift data section.  
2. The system should identify the total number of records in the Lateral Shift data section.  
3. The system should then return to the first record in the Lateral Shift data section.  
  
Definition of Done:  
- The "Save" button should trigger the navigation to the Lateral Shift data section.  
- The total number of records should be correctly identified and stored.  
- The system should successfully navigate back to the first record in the Lateral Shift data section.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided logic does not include any direct database CRUD operations.

# Dropdown List for Continuous Months in Policy Members

Type: LATERAL\_SHIFT

Title: Dropdown List for Continuous Months in Policy Members  
  
Acceptance Criteria:  
- The user should be able to see a dropdown list labeled "Months" when managing policy members.  
- The dropdown list should contain 13 predefined options for the number of continuous months.  
- The default value of the dropdown list should be set to 0.  
- The dropdown list should be positioned correctly within the user interface, ensuring it is easily accessible and visible.  
  
Definition of Done:  
- The dropdown list for selecting continuous months is implemented and visible in the policy members section.  
- The dropdown list contains 13 predefined options.  
- The default value is set to 0.  
- The dropdown list is correctly positioned and labeled "Months".  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include any specific database queries.

# Dropdown List for Continuous Days

Type: LATERAL\_SHIFT

Title: Dropdown List for Continuous Days  
  
Acceptance Criteria:  
1. The user should be able to see a dropdown list labeled "Days" on the "Policy Members" tab.  
2. The dropdown list should be initialized with a default value of 0.  
3. The dropdown list should contain 32 predefined options for the number of continuous days.  
4. The dropdown list should be positioned correctly on the screen as per the design specifications.  
5. The dropdown list should be displayed on a tabbed canvas named "NBTABS" within the "New Business" window.  
  
Definition of Done:  
- The dropdown list for continuous days is implemented and visible on the "Policy Members" tab.  
- The dropdown list is initialized with a default value of 0.  
- The dropdown list contains 32 predefined options.  
- The dropdown list is correctly positioned and displayed on the "NBTABS" canvas within the "New Business" window.  
- The feature has been tested and verified to meet the acceptance criteria.  
  
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 an old product from a list of values

Type: LATERAL\_SHIFT

Title: User can select an old product from a list of values  
  
Acceptance Criteria:  
1. When the user double-clicks on the "Old Product" field, a list of values (LOV) should be displayed.  
2. The LOV should fetch and display the company name and policy name from the "azbj\_other\_insurance\_policy" table.  
3. The user should be able to select a value from the LOV, and the selected value should populate the "Old Product" field.  
  
Definition of Done:  
1. The "Old Product" field should be interactive and respond to double-click events.  
2. The LOV should correctly fetch data from the database and display it to the user.  
3. The selected value from the LOV should be correctly populated in the "Old Product" field.  
4. The functionality should be tested and verified to ensure it works as expected.  
  
DB queries for Table reference CRUD operations only(With Usage):  
```sql  
SELECT COMPANY\_NAME, POLICY\_NAME   
FROM azbj\_other\_insurance\_policy;  
```  
This query is used to fetch the company name and policy name for the LOV.

# Dropdown for Continuous Years in New Business Section

Type: LATERAL\_SHIFT

Title: Dropdown for Continuous Years in New Business Section  
  
Acceptance Criteria:  
1. The user should see a dropdown list labeled "Yrs" in the "New Business" section.  
2. The dropdown list should be positioned appropriately within the form.  
3. The dropdown list should have a default value of "0".  
4. The dropdown list should contain 16 predefined options for the number of continuous years.  
5. The dropdown list should be part of the "Policy Members" tab.  
  
Definition of Done:  
1. The dropdown list is visible and correctly labeled as "Yrs".  
2. The dropdown list is correctly positioned within the "New Business" section.  
3. The dropdown list defaults to "0" when the form is initialized.  
4. The dropdown list contains 16 options for the number of continuous years.  
5. The dropdown list is accessible within the "Policy Members" tab.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include any specific database queries.

# Manage Scrutiny Failure Process

Type: SCRUTINY\_FAIL

Title: Manage Scrutiny Failure Process  
  
Acceptance Criteria:  
1. The form should display fields for:  
 - Document Description  
 - Whether Required Documents are Received (with a default value of 'Y')  
 - Additional Comments  
2. The form should have buttons for:  
 - Save (initially disabled)  
 - Update (enabled)  
 - Exit  
3. The form should be user-friendly and visually organized, with appropriate labels and prompts for each field.  
4. The form should be modal, ensuring that the user completes the scrutiny failure process before proceeding to other tasks.  
  
Definition of Done:  
- The form is implemented and tested to ensure it meets the acceptance criteria.  
- The form is visually verified to ensure all fields and buttons are correctly displayed and functional.  
- The form is user-tested to ensure ease of use and proper functionality.  
  
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.

# Exit Button Functionality in Scrutiny Failure Process

Type: SCRUTINY\_FAIL

Title: Exit Button Functionality in Scrutiny Failure Process  
  
Acceptance Criteria:  
- When the "Exit" button is pressed, the system should navigate to the main menu.  
- The "Scrutiny Failure Process" screen should be hidden from view.  
  
Definition of Done:  
- The "Exit" button is functional and correctly navigates the user to the main menu.  
- The "Scrutiny Failure Process" screen is hidden upon pressing the "Exit" button.  
- The functionality is tested and verified to ensure it works as expected.

# Enable/Disable Comments Field Based on Document Description

Type: SCRUTINY\_FAIL

Title: Enable/Disable Comments Field Based on Document Description  
  
Acceptance Criteria:  
- When the document description is set to "Others", the "Comments" field should be editable.  
- When the document description is not set to "Others", the "Comments" field should be read-only.  
  
Definition of Done:  
- The "Comments" field is editable only when the document description is "Others".  
- The "Comments" field is read-only when the document description is not "Others".  
- The changes are tested and verified to ensure the correct behavior of the "Comments" field based on the document description.  
  
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.

# Indicate Document Receipt Status in Scrutiny Failure Process

Type: SCRUTINY\_FAIL

Title: Indicate Document Receipt Status in Scrutiny Failure Process  
  
Acceptance Criteria:  
1. The user should be able to select from a list of options to indicate whether the required document has been received.  
2. The default value for this selection should be "Yes".  
3. The selection should be clearly visible and easy to interact with on the user interface.  
4. The selection should be saved and retrievable for future reference.  
  
Definition of Done:  
1. The user interface displays a list item for indicating document receipt status.  
2. The list item defaults to "Yes" upon initialization.  
3. The user can change the selection and the change is saved correctly.  
4. The selection is displayed with appropriate styling and is user-friendly.  
5. 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 XML content does not include specific database queries.

# Update Scrutiny Fail Document Details

Type: SCRUTINY\_FAIL

Title: Update Scrutiny Fail Document Details  
  
Acceptance Criteria:  
1. When the user presses the "Update" button, a confirmation alert should be displayed asking, "Are You Sure? You Want To Update the Scrutiny Fail document details?".  
2. If the user confirms the action, the system should proceed to update the scrutiny fail document details.  
3. The system should iterate through all records in the scrutiny fail block and update the following fields in the `azbj\_scrutiny\_fail` table:  
 - `document\_checked` with the value from the `SCRUTINY\_FAIL\_DOCUMENT` field.  
 - `UPDATE\_USER` with the current user's ID.  
 - `UPDATE\_DT` with the current date and time.  
 - `COMMENTS` with the value from the `other` field.  
4. The update should be performed for records where `contract\_id` matches the value in the `cn\_contract\_id` field and `document\_code` matches the value in the `doc\_code` field.  
5. If any error occurs during the update, an error message should be displayed with the error details.  
6. After all records are processed, the changes should be committed to the database.  
  
Definition of Done:  
- The "Update" button triggers the confirmation alert.  
- The system updates the scrutiny fail document details as per the acceptance criteria.  
- Error handling is implemented to display appropriate error messages.  
- The changes are committed to the database after processing all records.  
  
DB queries for Table reference CRUD operations only(With Usage):  
```sql  
UPDATE azbj\_scrutiny\_fail  
SET document\_checked = :SCRUTINY\_FAIL.SCRUTINY\_FAIL\_DOCUMENT,  
 UPDATE\_USER = :boiler.userid,  
 UPDATE\_DT = SYSDATE,  
 COMMENTS = :SCRUTINY\_FAIL.other  
WHERE contract\_id = :control.cn\_contract\_id  
 AND document\_code = :SCRUTINY\_FAIL.doc\_code;  
```

# Save Scrutiny Failure Document Details

Type: SCRUTINY\_FAIL

Title: Save Scrutiny Failure Document Details  
  
Acceptance Criteria:  
1. When the save button is pressed, the system should:  
 - Navigate to the 'SCRUTINY\_FAIL' section and check each record to ensure that the 'SCRUTINY\_FAIL\_DOCUMENT' field is marked as 'N'.  
 - If any record has 'SCRUTINY\_FAIL\_DOCUMENT' marked as 'N', the process should stop, and no data should be saved.  
 - If all records are valid, the system should prompt the user with a confirmation message: "Are You Sure? You Want To save the Scrutiny Fail document details?"  
 - If the user confirms, the system should:  
 - Retrieve the scrutiny number, user ID, and user name from the database based on the application number.  
 - Insert the scrutiny failure document details into the 'azbj\_scrutiny\_fail' table for each valid record.  
 - Commit the transaction.  
 - If the user cancels, the process should be aborted, and no data should be saved.  
 - If no valid scrutiny failure document is found, the system should display an error message: "There is No Scrutiny Fail Document mentioned."  
  
Definition of Done:  
- The save functionality should be implemented as described.  
- All validations and user confirmations should be in place.  
- Data should be correctly inserted into the database and committed only after user confirmation.  
- Appropriate error messages should be displayed when necessary.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Retrieve scrutiny number, user ID, and user name:  
 ```sql  
 SELECT SCRUTINY\_NO, USER\_ID, az\_pk2\_general.getUserName(USER\_ID)  
 INTO v\_scrutiny\_no, v\_user\_id, v\_user\_name  
 FROM azbj\_phub\_scrutiny\_prop  
 WHERE APPLICATION\_NO = TO\_CHAR(NVL(:INSURED\_PERSON.IP\_VERF\_NO, :INSURED\_PERSON.IP\_SIGN\_CARD\_NO));  
 ```  
  
- Insert scrutiny failure document details:  
 ```sql  
 INSERT INTO azbj\_scrutiny\_fail (  
 POLICY\_REF, CONTRACT\_ID, DOCUMENT\_desc, DOCUMENT\_TYPE, DOCUMENT\_CODE,  
 SCRUTINY\_FAIL\_DOCUMENT, INSERT\_USER, INSERT\_DT, UPDATE\_USER, UPDATE\_DT,  
 COMMENTS, BRANCH\_CODE, HUB\_CODE, IC\_CODE, STM\_CODE, SCRUTINY\_NO, RECEIPT\_NO,  
 USER\_ID, USER\_NAME  
 ) VALUES (  
 :control.cn\_policy\_ref, :control.cn\_contract\_id, :SCRUTINY\_FAIL.doc\_DESC,  
 :SCRUTINY\_FAIL.doc\_type, :SCRUTINY\_FAIL.doc\_code,  
 :SCRUTINY\_FAIL.SCRUTINY\_FAIL\_DOCUMENT, :boiler.userid, SYSDATE, NULL, NULL,  
 :SCRUTINY\_FAIL.other, :control.CN\_BRANCH\_CODE, SUBSTR(:boiler.userid, 1, 3),  
 :agents.AG\_AGENT\_CODE, :agents.AG\_UNIT\_MGR, v\_scrutiny\_no,  
 :control.cn\_permrcpt\_no, v\_user\_id, v\_user\_name  
 );  
 ```

# Manage Fake Document Records

Type: FAKE\_DOCUMENT

Detailed description: As a user, I want to manage fake document records, including adding, updating, and validating various attributes such as document description, type, code, proof type, and category, so that I can ensure the integrity and accuracy of the document data.  
  
Acceptance criteria:  
1. The system should allow the user to input and display the following attributes for a fake document:  
 - Document Description  
 - Document Type  
 - Document Code  
 - Whether the document is fake  
 - Document Proof Type  
 - Comments  
 - Category  
 - Document Checked status  
2. The system should provide a Save button to save new fake document records.  
3. The system should provide an Update button to update existing fake document records.  
4. The system should provide an Exit button to close the fake document management interface.  
5. The system should validate the 'Category' and 'Comments' fields against predefined lists.  
6. The system should initialize the 'Whether Fake Doc' and 'Issuance not allowed' fields with default values.  
7. The system should fetch and display comments from a predefined list of values based on a specific query.  
  
Definition of Done:  
- The user can successfully add, update, and validate fake document records.  
- The Save, Update, and Exit buttons function as expected.  
- The 'Category' and 'Comments' fields are validated against predefined lists.  
- Default values are correctly initialized for the 'Whether Fake Doc' and 'Issuance not allowed' fields.  
- Comments are fetched and displayed based on the predefined query.  
  
DB queries for Table reference CRUD operations only (With Usage):  
```sql  
SELECT sys\_code, SYS\_DESC  
FROM azbj\_system\_constants  
WHERE sys\_type = 'FAKE\_LOV';  
```  
- This query is used to fetch the list of values for the 'Comments' field.

# Exit Button Functionality

Type: FAKE\_DOCUMENT

Title: Exit Button Functionality  
  
Acceptance Criteria:  
1. When the "Exit" button is pressed, the system should navigate to the main document management screen.  
2. The current document view should be hidden from the user interface.  
  
Definition of Done:  
1. The "Exit" button is visible and accessible on the document view screen.  
2. Pressing the "Exit" button successfully navigates the user to the main document management screen.  
3. The current document view is no longer visible after pressing the "Exit" button.  
4. 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 XML content does not include any database CRUD operations.

# Automatic Uncheck of 'Issuance not allowed' Checkbox

Type: FAKE\_DOCUMENT

Detailed description: As a user, I want to ensure that the 'Issuance not allowed' checkbox is automatically unchecked if the associated document is not marked as fake, so that the system maintains data integrity and prevents incorrect information from being saved.  
  
Acceptance criteria:  
1. When the 'Issuance not allowed' checkbox is checked, the system should verify if the associated document is marked as fake.  
2. If the document is not marked as fake, the system should automatically uncheck the 'Issuance not allowed' checkbox.  
3. The system should display an error message stating, 'Please untick the check box as Fake document not marked.'  
  
Definition of Done:  
1. The functionality is implemented and tested to ensure that the checkbox is automatically unchecked if the document is not marked as fake.  
2. The error message is displayed correctly when the checkbox is checked but the document is not marked as fake.  
3. The feature is reviewed and approved by the stakeholders.  
4. The changes are documented and deployed to the production environment.

# Update Fake Document Status

Type: FAKE\_DOCUMENT

Title: Update Fake Document Status  
  
Acceptance Criteria:  
1. When the "Update Fake Document" button is pressed, the system should:  
 - Check if the "Fake Document" status is marked as 'Y' and ensure that comments and category fields are not empty.  
 - Display an error message if any required fields are missing.  
 - Prompt the user with a confirmation alert asking if they wish to continue generating a report for the fake document.  
 - If the user chooses to proceed, update the relevant records in the database with the provided information.  
 - Commit the changes to the database.  
 - Navigate to the "MED\_UW" section and display the first record.  
  
Definition of Done:  
- The system validates the required fields for documents marked as "Fake".  
- The user is prompted with a confirmation alert before proceeding with the update.  
- The database is updated with the new information for each document.  
- The changes are committed to the database.  
- The user is navigated to the "MED\_UW" section and the first record is displayed.  
  
DB queries for Table reference CRUD operations only(With Usage):  
```sql  
UPDATE azbj\_fake\_document  
SET document\_checked = :FAKE\_DOCUMENT.FAKE\_DOCUMENT,  
 UPDATE\_USER = :boiler.userid,  
 UPDATE\_DT = sysdate,  
 COMMENTS = :FAKE\_DOCUMENT.other,  
 FAKE\_MARK = :FAKE\_DOCUMENT.FAKE\_MARK,  
 doc\_proof\_type = :FAKE\_DOCUMENT.doc\_proof\_type,  
 category = :FAKE\_DOCUMENT.category  
WHERE contract\_id = :control.cn\_contract\_id  
 AND document\_code = :FAKE\_DOCUMENT.doc\_code;  
```

# Toggle Editability of Comments Field Based on FAKE\_DOCUMENT Field Value

Type: FAKE\_DOCUMENT

Title: Toggle Editability of Comments Field Based on FAKE\_DOCUMENT Field Value  
  
Acceptance Criteria:  
1. The "Comments" field should be editable if the value of the "FAKE\_DOCUMENT" field is 'Y'.  
2. The "Comments" field should not be editable if the value of the "FAKE\_DOCUMENT" field is not 'Y'.  
3. When the "Comments" field is double-clicked, a list of values should be displayed for selection.  
  
Definition of Done:  
1. The "Comments" field's editability is correctly toggled based on the value of the "FAKE\_DOCUMENT" field.  
2. The list of values is displayed correctly when the "Comments" field is double-clicked.  
3. The functionality is tested and verified to work as expected.  
  
DB queries for Table reference CRUD operations only(With Usage):  
```sql  
SELECT sys\_code, SYS\_DESC  
FROM azbj\_system\_constants  
WHERE sys\_type = 'FAKE\_LOV';  
```  
This query is used to fetch the list of values for the "Comments" field.

# Dynamic Enable/Disable of 'Other' Field Based on 'FAKE\_DOCUMENT' Field

Type: FAKE\_DOCUMENT

Detailed description: As a user, I want the "Category" field in the document form to dynamically enable or disable the "Other" field based on the value of the "FAKE\_DOCUMENT" field, so that the form behavior is consistent with the business rules.  
  
Acceptance criteria:  
1. If the "FAKE\_DOCUMENT" field is set to 'Y', the "Other" field should be enabled for both updates and inserts.  
2. If the "FAKE\_DOCUMENT" field is not set to 'Y', the "Other" field should be disabled for both updates and inserts.  
3. The "Category" field should display a list of values fetched from the database, specifically from the `azbj\_system\_constants` table where `sys\_type` is 'FAKE\_LOV'.  
4. Double-clicking on the "Category" field should display the list of values for selection.  
  
Definition of Done:  
- The "Category" field dynamically enables or disables the "Other" field based on the value of the "FAKE\_DOCUMENT" field.  
- The list of values for the "Category" field is correctly fetched and displayed from the database.  
- Double-click functionality on the "Category" field works as expected.  
- All changes are tested and verified to meet the acceptance criteria.  
  
DB queries for Table reference CRUD operations only (With Usage):  
```sql  
SELECT sys\_code, SYS\_DESC  
FROM azbj\_system\_constants  
WHERE sys\_type = 'FAKE\_LOV';  
```  
This query is used to fetch the list of values for the "Category" field.

# Dynamic Enable/Disable of Document Proof Type Field

Type: FAKE\_DOCUMENT

Detailed description: As a user, I want the system to dynamically enable or disable the "Document Proof Type" field based on the value of the "Fake Document" field and the "Document Code" field, so that the field's editability is controlled according to specific conditions.  
  
Acceptance criteria:  
1. If the "Fake Document" field is set to 'Y':  
 - The "Document Proof Type" field should be enabled for both updates and inserts.  
 - The "Fake Mark" field should be enabled for both updates and inserts.  
2. If the "Fake Document" field is not set to 'Y':  
 - The "Document Proof Type" field should be disabled for both updates and inserts.  
 - The "Fake Mark" field should be disabled for both updates and inserts.  
3. When the "Document Proof Type" field is double-clicked:  
 - If the "Document Code" is 'M02' and "Fake Document" is 'Y', a list of values (LOV) should be displayed with address proof types.  
 - If the "Document Code" is 'M03' and "Fake Document" is 'Y', a LOV should be displayed with income proof types.  
 - If the "Document Code" is 'M04' and "Fake Document" is 'Y', a LOV should be displayed with ID proof types.  
 - If the "Document Code" is 'M01' and "Fake Document" is 'Y', a LOV should be displayed with age proof types.  
 - If none of the above conditions are met, the "Fake Mark" field should be disabled for both updates and inserts.  
  
Definition of Done:  
- The "Document Proof Type" field's editability is correctly controlled based on the "Fake Document" and "Document Code" fields.  
- The appropriate LOV is displayed when the "Document Proof Type" field is double-clicked, based on the "Document Code" and "Fake Document" values.  
- The "Fake Mark" field's editability is correctly controlled based on the "Fake Document" and "Document Code" fields.  
- All functionalities are tested and verified to work as expected.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- The following query is used to populate the LOV for address proof types:  
 ```sql  
 SELECT DISTINCT Proof\_type, proof\_desc, display\_message  
 FROM azbj\_aml\_proof\_master  
 WHERE partner\_type = :control.cn\_partner\_type  
 AND document\_type IN ('COMPANY\_ADDRESS', 'CURRENT\_ADDRESS', 'PARTNERSHIP\_FIRM\_ADDRESS', 'PERMANENT\_ADDRESS', 'PRINCIPAL\_PLACE\_OF\_BUSINESS', 'TELEPHONE\_FAX\_NUMBER', 'TELEPHONE\_NUMBER', 'TRUSTEES\_ADDRESSES');  
 ```  
  
- The following query is used to populate the LOV for income proof types:  
 ```sql  
 SELECT DISTINCT Proof\_type, proof\_desc, display\_message  
 FROM azbj\_aml\_proof\_master  
 WHERE partner\_type = :control.cn\_partner\_type  
 AND document\_type IN ('SOURCE\_OF\_FUNDS1', 'SOURCE\_OF\_FUNDS2', 'SOURCE\_OF\_FUNDS3');  
 ```  
  
- The following query is used to populate the LOV for ID proof types:  
 ```sql  
 SELECT DISTINCT Proof\_type, proof\_desc, display\_message  
 FROM azbj\_aml\_proof\_master  
 WHERE partner\_type = :control.cn\_partner\_type  
 AND document\_type IN ('COMPANY\_NAME', 'LEGAL\_NAME', 'PARTNER\_NAMES\_AND\_ADDRESSES', 'TRUSTEES\_NAME');  
 ```  
  
- The following query is used to populate the LOV for age proof types:  
 ```sql  
 SELECT PROOF\_TYPE Proof\_type, PROOF\_DESC proof\_desc, null display\_message  
 FROM azbj\_aml\_requirements  
 WHERE DOCUMENT\_TYPE = 'AGE\_PROOF'  
 AND ENABLED\_FLAG = 'Y'  
 ORDER BY proof\_type;  
 ```

# Save Fake Document Information

Type: FAKE\_DOCUMENT

Detailed description: As a user, I want to be able to save information about documents marked as fake, ensuring that all necessary details are provided before the save operation is completed.  
  
Acceptance criteria:  
1. When the save button is pressed, the system should check if any document is marked as fake.  
2. If a document is marked as fake, the system should ensure that comments and category fields are filled in for each fake document.  
3. If any required fields are missing, the system should display an appropriate error message and halt the save operation.  
4. If all required fields are filled, the system should prompt the user with a confirmation message asking if they wish to continue generating a report for the fake document.  
5. If the user confirms, the system should save the fake document details into the database.  
6. The system should commit the transaction and disable the save button after a successful save operation.  
7. If no document is marked as fake, the system should display a message indicating that no fake document is mentioned.  
  
Definition of Done:  
- The save functionality should be implemented and tested to ensure it meets all acceptance criteria.  
- Error messages should be clear and guide the user on what needs to be corrected.  
- The confirmation prompt should correctly handle user responses.  
- The save operation should correctly insert data into the database and handle any exceptions gracefully.  
- The save button should be disabled after a successful save operation.  
- The system should navigate back to the initial state after the save operation is completed.  
  
DB queries for Table reference CRUD operations only (With Usage):  
```sql  
INSERT INTO azbj\_fake\_document (  
 POLICY\_REF, CONTRACT\_ID, DOCUMENT\_desc, DOCUMENT\_TYPE, DOCUMENT\_CODE, FAKE\_DOCUMENT,  
 INSERT\_USER, INSERT\_DT, UPDATE\_USER, UPDATE\_DT, COMMENTS, BRANCH\_CODE, HUB\_CODE, IC\_CODE, STM\_CODE, fake\_mark,  
 doc\_proof\_type, category  
) VALUES (  
 :control.cn\_policy\_ref, :control.cn\_contract\_id, :FAKE\_DOCUMENT.doc\_DESC, :FAKE\_DOCUMENT.doc\_type, :FAKE\_DOCUMENT.doc\_code,  
 :FAKE\_DOCUMENT.FAKE\_DOCUMENT, :boiler.userid, sysdate, null, null, :FAKE\_DOCUMENT.other, :control.CN\_BRANCH\_CODE,  
 substr(:boiler.userid, 1, 3), :agents.AG\_AGENT\_CODE, :agents.AG\_UNIT\_MGR, :FAKE\_DOCUMENT.fake\_mark,  
 :FAKE\_DOCUMENT.doc\_proof\_type, :FAKE\_DOCUMENT.category  
);  
```

# Enable or Disable FAKE\_MARK based on FAKE\_DOCUMENT and DOC\_CODE

Type: FAKE\_DOCUMENT

Detailed description: As a user, I want the system to enable or disable the "FAKE\_MARK" field based on the value of the "FAKE\_DOCUMENT" field and the type of document code selected, so that I can ensure the correct fields are editable based on the document type.  
  
Acceptance criteria:  
1. When the "FAKE\_DOCUMENT" field is set to 'Y' and the "DOC\_CODE" is 'M02', the system should:  
 - Enable the "FAKE\_MARK" field.  
 - Populate a list of address proof types from the `azbj\_aml\_proof\_master` table where the document type is one of the specified address types.  
 - Display the list of address proof types to the user.  
2. When the "FAKE\_DOCUMENT" field is set to 'Y' and the "DOC\_CODE" is 'M03', the system should:  
 - Enable the "FAKE\_MARK" field.  
 - Populate a list of income proof types from the `azbj\_aml\_proof\_master` table where the document type is one of the specified income types.  
 - Display the list of income proof types to the user.  
3. When the "FAKE\_DOCUMENT" field is set to 'Y' and the "DOC\_CODE" is 'M04', the system should:  
 - Enable the "FAKE\_MARK" field.  
 - Populate a list of ID proof types from the `azbj\_aml\_proof\_master` table where the document type is one of the specified ID types.  
 - Display the list of ID proof types to the user.  
4. When the "FAKE\_DOCUMENT" field is set to 'Y' and the "DOC\_CODE" is 'M01', the system should:  
 - Enable the "FAKE\_MARK" field.  
 - Populate a list of age proof types from the `azbj\_aml\_requirements` table where the document type is 'AGE\_PROOF' and the enabled flag is 'Y'.  
 - Display the list of age proof types to the user.  
5. If the "FAKE\_DOCUMENT" field is not set to 'Y' or the "DOC\_CODE" does not match any of the specified codes, the system should:  
 - Disable the "FAKE\_MARK" field.  
  
Definition of Done:  
- The "FAKE\_MARK" field is enabled or disabled based on the value of the "FAKE\_DOCUMENT" field and the document code.  
- The correct list of proof types is populated and displayed to the user based on the document code.  
- The system behaves as expected for all specified document codes and conditions.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- For Address Proof:  
 ```sql  
 SELECT DISTINCT Proof\_type, proof\_desc, display\_message   
 FROM azbj\_aml\_proof\_master   
 WHERE partner\_type = :control.cn\_partner\_type   
 AND document\_type IN ('COMPANY\_ADDRESS', 'CURRENT\_ADDRESS', 'PARTNERSHIP\_FIRM\_ADDRESS', 'PERMANENT\_ADDRESS', 'PRINCIPAL\_PLACE\_OF\_BUSINESS', 'TELEPHONE\_FAX\_NUMBER', 'TELEPHONE\_NUMBER', 'TRUSTEES\_ADDRESSES');  
 ```  
  
- For Income Proof:  
 ```sql  
 SELECT DISTINCT Proof\_type, proof\_desc, display\_message   
 FROM azbj\_aml\_proof\_master   
 WHERE partner\_type = :control.cn\_partner\_type   
 AND document\_type IN ('SOURCE\_OF\_FUNDS1', 'SOURCE\_OF\_FUNDS2', 'SOURCE\_OF\_FUNDS3');  
 ```  
  
- For ID Proof:  
 ```sql  
 SELECT DISTINCT Proof\_type, proof\_desc, display\_message   
 FROM azbj\_aml\_proof\_master   
 WHERE partner\_type = :control.cn\_partner\_type   
 AND document\_type IN ('COMPANY\_NAME', 'LEGAL\_NAME', 'PARTNER\_NAMES\_AND\_ADDRESSES', 'TRUSTEES\_NAME');  
 ```  
  
- For Age Proof:  
 ```sql  
 SELECT PROOF\_TYPE Proof\_type, PROOF\_DESC proof\_desc, null display\_message   
 FROM azbj\_aml\_requirements   
 WHERE DOCUMENT\_TYPE = 'AGE\_PROOF'   
 AND ENABLED\_FLAG = 'Y'   
 ORDER BY proof\_type;  
 ```

# View and Manage Previous Fake Document Records

Type: PREV\_FAKE\_DOC

Title: View and Manage Previous Fake Document Records  
  
Acceptance Criteria:  
1. The system should display a list of previous fake document records with the following details:  
 - Serial Number  
 - Application Number  
 - Document Description  
 - Document Type  
 - Document Category  
 - Policy Number  
2. The list should be displayed in a grid format with a maximum of 5 records visible at a time.  
3. Each record should be displayed with appropriate labels and formatting for easy readability.  
4. There should be a button labeled "Exit" to close the view.  
5. There should be a button labeled "View" to view the image associated with the document.  
  
Definition of Done:  
- The user interface displays the previous fake document records as specified.  
- The "Exit" button successfully closes the view.  
- The "View" button successfully displays the image associated with the selected document.  
- The functionality is tested and verified to ensure it meets the acceptance criteria.  
  
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: PREV\_FAKE\_DOC

Title: Exit Button Functionality  
  
Acceptance Criteria:  
- When the "Exit" button is pressed, the system should hide the current document view window.  
- The system should then navigate the user back to the main document management screen.  
  
Definition of Done:  
- The "Exit" button is functional and correctly hides the current document view window.  
- The user is successfully navigated back to the main document management screen upon pressing the "Exit" button.  
- 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 Image Functionality

Type: PREV\_FAKE\_DOC

Detailed description: As a user, I want to view the image associated with a specific policy number so that I can verify the document details.  
  
Acceptance criteria:  
1. When the "View" button is pressed, the system should check if the policy number is not null.  
2. If the policy number is not null, the system should generate a URL using the `azbj\_encrypt\_dms\_link` function with the policy number as a parameter.  
3. If the generated URL is not null, the system should open the URL in a web browser.  
4. If there is an error in generating the URL, the system should display a message indicating that there is an issue with the URL.  
  
Definition of Done:  
- The "View" button functionality is implemented and tested.  
- The system correctly checks for a non-null policy number.  
- The URL is generated and opened in a web browser if valid.  
- An appropriate error message is displayed if there is an issue with the URL.  
  
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.

# Display Reinsurance Policy Details

Type: REINSURANCE\_DETAILS

Title: Display Reinsurance Policy Details  
  
Acceptance Criteria:  
1. The sum assured field should be displayed and should not be editable.  
2. The policy reference number field should be displayed and should not be editable.  
3. Both fields should be properly aligned and formatted for readability.  
  
Definition of Done:  
1. The reinsurance details section is implemented and displays the sum assured and policy reference number.  
2. Both fields are non-editable and correctly formatted.  
3. The user interface is tested to ensure that the fields are displayed as expected and are not editable.  
4. The changes are reviewed and approved by the stakeholders.

# Manage Third-Party Cheque Details

Type: AZBJ\_THIRD\_PARTY\_CHQ

Title: Manage Third-Party Cheque Details  
  
Acceptance Criteria:  
1. The system should allow users to input the payment mode for third-party cheques.  
2. The system should provide a text field for users to enter questions related to the third-party cheque, with a maximum length of 1000 characters.  
3. The system should offer a dropdown list for users to select the confidence percentage related to the third-party cheque, with up to 20 predefined options.  
  
Definition of Done:  
- The user can successfully input and save the payment mode for third-party cheques.  
- The user can enter and save questions related to the third-party cheque.  
- The user can select and save a confidence percentage from a dropdown list.  
- All inputs are displayed correctly on the user interface and stored in the database.  
  
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 Confidence Percentage Selection

Type: AZBJ\_THIRD\_PARTY\_CHQ

Title: Implement Confidence Percentage Selection  
  
Acceptance Criteria:  
1. The confidence percentage should be selectable from a predefined list of 20 elements.  
2. The list should be displayed in a dropdown format.  
3. The selected confidence percentage should be displayed in a specific section of the form.  
4. The form should be visually organized with the confidence percentage field aligned and styled consistently with other form elements.  
  
Definition of Done:  
1. The confidence percentage field is implemented and functional.  
2. The dropdown list contains 20 predefined elements.  
3. The field is properly aligned and styled according to the design specifications.  
4. The form is tested to ensure the confidence percentage can be selected and saved correctly.  
  
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 table references.

# View and Manage Reinsurance Cover Details

Type: REINS\_COVER\_DETAILS

Title: View and Manage Reinsurance Cover Details  
  
Acceptance Criteria:  
1. The user should be able to view the following fields in the reinsurance cover details section:  
 - Cover Code  
 - Sum Assured  
 - ML Percentage  
 - Sr Percentage  
 - OCC Percentage  
 - Res Percentage  
2. All fields should be displayed in a read-only format.  
3. The fields should be aligned and positioned as follows:  
 - Cover Code at the top left  
 - Sum Assured to the right of Cover Code  
 - ML Percentage to the right of Sum Assured  
 - Sr Percentage to the right of ML Percentage  
 - OCC Percentage to the right of Sr Percentage  
 - Res Percentage to the right of OCC Percentage  
4. The fields should have appropriate labels:  
 - Cover Code  
 - Sum Assured  
 - ML Perc  
 - Sr Perc  
 - OCC Perc  
 - Res Perc  
5. The data type for Sum Assured, ML Percentage, Sr Percentage, OCC Percentage, and Res Percentage should be numeric.  
6. The Cover Code should be in uppercase letters.  
  
Definition of Done:  
- The reinsurance cover details section is implemented and displays all required fields in a read-only format.  
- The fields are correctly aligned and positioned as specified.  
- The labels for each field are displayed correctly.  
- Numeric fields are validated to ensure they only accept numeric input.  
- The Cover Code field is validated to ensure it is in uppercase letters.  
- The implementation is tested and verified to meet all acceptance criteria.

# Validate OCC Perc Field in Reinsurance Cover Details

Type: REINS\_COVER\_DETAILS

Title: Validate OCC Perc Field in Reinsurance Cover Details  
  
Acceptance Criteria:  
1. The "OCC Perc" field should be a numeric field with a maximum length of 15 characters.  
2. The field should be disabled for both insertion and updates.  
3. When validating the "OCC Perc" field, if the product definition is 'CAPITAL\_SHIELD' and the value of "OCC Perc" is greater than 0, an error message should be displayed indicating that the product is not allowed on extra premium.  
  
Definition of Done:  
1. The "OCC Perc" field is correctly displayed in the Reinsurance Cover Details section.  
2. The field is non-editable and non-insertable.  
3. The validation logic is implemented and tested to ensure that the error message is displayed under the specified conditions.  
4. All acceptance criteria are met and verified through testing.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include any direct database operations.

# Validate Special Risk Percentage Field

Type: REINS\_COVER\_DETAILS

Detailed description: As a user, I want to ensure that the "Special Risk Percentage" field in the "Reinsurance Cover Details" section is validated according to specific business rules, so that only valid values are accepted.  
  
Acceptance criteria:  
1. If the product associated with the "Special Risk Percentage" field is 'CAPITAL\_SHIELD' and the value entered is greater than 0, an error message should be displayed stating, "Product not allowed on extra Premium".  
2. The validation should be triggered when the user attempts to save or move out of the "Special Risk Percentage" field.  
  
Definition of Done:  
- The validation logic is implemented and tested.  
- Error messages are displayed correctly based on the specified conditions.  
- The user is prevented from saving invalid data in the "Special Risk Percentage" field.  
- The functionality is 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 operations.

# Automatic Calculation and Validation of Entry Age, Benefit Term, and Premium Term

Type: REINS\_COVER\_DETAILS

Title: Automatic Calculation and Validation of Entry Age, Benefit Term, and Premium Term  
  
Acceptance Criteria:  
1. The system should calculate the entry age for both the insured person and the policyholder using their respective dates of birth and the inception or effective date.  
2. If the product is 'SWARNA\_RAKSHA1\_JLLS', the system should use the later date of birth between the insured person and the policyholder for age calculation.  
3. For specific cover codes and product IDs, the system should use the policyholder's date of birth for age calculation.  
4. The system should update the entry age, benefit term, and premium term fields based on the calculated values.  
5. If the cover code is 'R023A01', the entry age should not be recalculated.  
6. For product ID 4, the benefit term should be calculated as the maturity age minus the entry age, and the premium term should be set to the same value as the benefit term.  
7. For product ID 11, the benefit term should be calculated based on the package code, and the premium term should be set to 18 minus the entry age.  
8. The system should handle exceptions and display error messages if any issues occur during the calculations.  
  
Definition of Done:  
- The system correctly calculates and updates the entry age, benefit term, and premium term based on the provided logic.  
- The calculations are performed without any errors, and appropriate error messages are displayed if any issues occur.  
- The updated values are displayed and stored in the system as expected.  
- The functionality is tested and verified to ensure accuracy and reliability.

# Validation of Medical Loading Percentage for Riders and Products

Type: REINS\_COVER\_DETAILS

Title: Validation of Medical Loading Percentage for Riders and Products  
  
Acceptance Criteria:  
1. If the cover code is 'R001A01', 'R004A01', 'R005A01', 'R020A01', or 'R021A01', the medical loading percentage must be either 0 or 100.  
2. If the cover code is 'R003A01', the medical loading percentage must be one of the following values: 0, 25, 50, 75, 100, or 150.  
3. If the cover code starts with 'L' or is 'R002A01', the medical loading percentage must be between 0 and 500.  
4. If the cover code is 'R018A01' or 'L066A01', the medical loading percentage must be between 0 and 100.  
5. If the product definition is 'CAPITAL\_SHIELD' and the non-resident insurance percentage is greater than 0, an error message should be displayed indicating that the product is not allowed on extra premium.  
  
Definition of Done:  
- The validation rules for medical loading percentage are implemented and tested.  
- Error messages are displayed correctly when validation rules are not met.  
- The system prevents the user from saving invalid data.  
- All acceptance criteria are met and verified through testing.

# Medical Loading Percentage Validation

Type: REINS\_COVER\_DETAILS

Title: Medical Loading Percentage Validation  
  
Acceptance Criteria:  
1. If the cover code is one of 'R001A01', 'R004A01', 'R005A01', 'R020A01', or 'R021A01', the Medical Loading Percentage must be either 0 or 100. If it is not, an error message should be displayed: "The Medical Loading Percentage can be 0 or 100 for this Rider."  
2. If the cover code is 'R003A01', the Medical Loading Percentage must be one of the following values: 0, 25, 50, 75, 100, or 150. If it is not, an error message should be displayed: "The Medical Loading Percentage can be 0, 25, 50, 75, 100, 150 for this Rider."  
3. If the cover code starts with 'L' or is 'R002A01', the Medical Loading Percentage must be between 0 and 500. If it is not, an error message should be displayed: "The Medical Loading Percentage can be between 0 and 500."  
4. If the cover code is 'R018A01' or 'L066A01', the Medical Loading Percentage must be between 0 and 100. If it is not, an error message should be displayed: "The Medical Loading Percentage can be between 0 and 100."  
5. If the product definition for the given product ID is 'CAPITAL\_SHIELD' and the Medical Loading Percentage is greater than 0, an error message should be displayed: "Product not allowed on extra Premium."  
  
Definition of Done:  
- The validation rules for the Medical Loading Percentage are implemented and tested.  
- Error messages are displayed correctly based on the validation rules.  
- The system prevents users from entering invalid Medical Loading Percentage values based on the cover code and product definition.  
- All acceptance criteria are met and verified through testing.

# Calculate and Validate Insurance Premiums

Type: REINS\_COVER\_DETAILS

User Story: Calculate and Validate Insurance Premiums  
  
Detailed Description:  
As an insurance system, I need to calculate and validate the insurance premiums for different types of policies, ensuring that all necessary conditions and validations are met before finalizing the premium amount.  
  
Acceptance Criteria:  
1. Premium Calculation:  
 - The system should calculate the premium based on the sum assured, booking frequency, and other relevant factors.  
 - If the booking frequency is null, the system should raise an error indicating that the booking frequency must be selected.  
 - For specific products, additional calculations such as health covers or non-standard age proof loadings should be applied.  
  
2. Validation of Age Proof:  
 - For certain relations (e.g., Mother, Father), the system should validate the age proof type and count the number of non-standard age proofs.  
 - If the age proof type is non-standard, additional loadings should be applied to the premium.  
  
3. Handling Different Product Types:  
 - For specific product IDs, the system should apply different premium calculations and loadings.  
 - For example, products like 'CARE\_FIRST' should include health cover calculations, while others may have different multipliers or factors based on the product definition.  
  
4. Service Tax and GST Calculations:  
 - The system should calculate the applicable service tax and GST based on the premium amount and other factors.  
 - If the GST flag is enabled, the system should fetch the GST amount and apply it to the premium.  
 - The system should handle exemptions and specific conditions for different product IDs.  
  
5. Error Handling and Messaging:  
 - The system should provide clear error messages for any validation failures, such as missing booking frequency, non-standard age proofs, or service tax calculation errors.  
 - Errors should be logged and displayed to the user, preventing further processing until resolved.  
  
Definition of Done:  
- The premium calculation logic is implemented and tested for various product types and conditions.  
- Age proof validations are correctly applied, and additional loadings are calculated as needed.  
- Service tax and GST calculations are accurately performed, with exemptions handled appropriately.  
- Clear error messages are provided for any validation failures, and errors are logged.  
- The system is tested with different scenarios to ensure all acceptance criteria are met.  
  
DB Queries for Table Reference CRUD Operations:  
- Not applicable as the provided XML content does not include specific CRUD operations that can be executed directly in the database without modification.

# View and Interact with Risk Details

Type: BALIC\_RISK

Title: View and Interact with Risk Details  
  
Acceptance Criteria:  
1. The system should display the following fields for each policy:  
 - Policy Number  
 - Contract ID  
 - Sum Assured  
 - Date of Risk  
 - Product Name  
 - Annual Premium  
 - Term of Acceptance  
 - Time Stamp  
 - Paid By  
 - Main Cover Rated Up  
 - Reason for Rating  
 - Value  
 - Rider Cover Rated Up  
 - BT  
 - PT  
2. The fields should be read-only and not allow any updates or insertions.  
3. The system should provide buttons for viewing images and adverse factor details.  
4. The fields should be displayed in a tabbed layout under the "Questionnaire" tab.  
  
Definition of Done:  
- The user can view all the specified fields in a read-only format.  
- The buttons for viewing images and adverse factor details are functional.  
- The layout is organized under the "Questionnaire" tab.  
- The system meets all the acceptance criteria and passes user acceptance testing.

# View Image Button Functionality

Type: BALIC\_RISK

Title: View Image Button Functionality  
  
Acceptance Criteria:  
1. When the "Images" button is pressed, the system should check if the policy number is not null.  
2. If the policy number is not null, the system should generate a URL using the `azbj\_encrypt\_dms\_link` function with the parameters 'NB', 'POLICY\_DOCS', and the policy number.  
3. If the generated URL is not null, the system should open the URL in a web browser.  
4. If an error occurs during the URL generation or opening process, the system should display a message indicating that there is an issue with the URL.  
  
Definition of Done:  
- The "Images" button functionality is implemented and tested.  
- The system correctly checks for a non-null policy number.  
- The URL is generated and opened correctly when the policy number is valid.  
- Appropriate error messages are displayed if there are issues with the URL.  
  
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.

# Manage Reinsurer Details

Type: REINSURER\_DETAILS

Detailed description: As a user, I want to manage reinsurer details within the system, including capturing and displaying various reinsurer-related information such as codes, reasons, recommendations, comments, and financial details. This will help in maintaining accurate and comprehensive records of reinsurer interactions and decisions.  
  
Acceptance criteria:  
1. The system should allow the user to view the reinsurer code, reinsurer reason, and reinsurer's recommendations.  
2. The system should display the current and previous amounts for CI Rider TASA, main TASA, and SAR.  
3. The system should provide fields for capturing underwriter's recommendations, medical comments, financial comments, and final comments.  
4. The system should display reinsurer's medical comments, financial comments, and final comments.  
5. The system should allow the user to refer a case to the reinsurer by checking a checkbox.  
6. The system should display various TASA amounts related to RI, PH, and IP.  
7. The system should have a button to navigate back to the previous screen.  
8. The system should ensure that certain fields are read-only and cannot be updated by the user.  
  
Definition of Done:  
1. The user can view and interact with the reinsurer details section as described.  
2. All fields and functionalities are implemented and tested to ensure they meet the acceptance criteria.  
3. The user interface is intuitive and aligns with the overall design of the application.  
4. The system handles data validation and displays appropriate error messages where necessary.  
5. 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 SQL queries or database operations.

# Underwriter's Recommendation Field on Reinsurer Selection Screen

Type: REINSURER\_DETAILS

Title: Underwriter's Recommendation Field on Reinsurer Selection Screen  
  
Acceptance Criteria:  
1. The recommendation field should display a list of five predefined options.  
2. The field should be disabled by default, preventing any modifications.  
3. The field should be clearly labeled as "Underwriter's Recommendation".  
4. The field should be positioned at the specified coordinates on the Reinsurer Selection Screen.  
5. The Reinsurer Selection Screen should have a specific size and should not allow vertical scrolling.  
  
Definition of Done:  
1. The recommendation field is visible and correctly labeled on the Reinsurer Selection Screen.  
2. The field displays the predefined list of five options.  
3. The field is disabled and cannot be modified by the user.  
4. The Reinsurer Selection Screen matches the specified dimensions and properties.  
5. All acceptance criteria are met and verified through testing.

# Navigate Back to Previous Section Using 'Back' Button

Type: REINSURER\_DETAILS

Title: Navigate Back to Previous Section Using 'Back' Button  
  
User Story:  
As a user, I want to navigate back to the previous section from the current section when I press the 'Back' button, so that I can easily return to the previous screen without losing my current context.  
  
Acceptance Criteria:  
1. When the 'Back' button is pressed, the system should navigate to the previous section named 'COVERS'.  
2. The current window named 'C\_OFFER' should be hidden upon pressing the 'Back' button.  
  
Definition of Done:  
- The 'Back' button is functional and navigates to the 'COVERS' section.  
- The 'C\_OFFER' window is hidden when the 'Back' button is pressed.  
- The functionality is tested and verified to ensure smooth navigation without errors.  
  
Block Name: REINSURER\_DETAILS

# Retention Amount Field in Reinsurer Selection Screen

Type: REINSURER\_DETAILS

Detailed description: As a user, I want to be able to input and view the retention amount for a reinsurer in a dedicated section of the Reinsurer Selection Screen, so that I can manage and review the retention details efficiently.  
  
Acceptance criteria:  
1. The retention amount field should be clearly labeled as "Retain Amt".  
2. The field should be positioned appropriately within the Reinsurer Selection Screen.  
3. The field should accept alphanumeric characters up to a maximum length of 150 characters.  
4. The field should be displayed with a white background and use the "Microsoft Sans Serif" font.  
5. The field should be part of a list item type and should be displayed with a height of 18 units and a width of 124 units.  
6. The prompt for the field should be styled with the "Arial" font, bold weight, and a size of 800 units.  
7. The Reinsurer Selection Screen should be a dialog window with a title "Reinsurer Selection Screen" and should not allow minimizing but should allow maximizing.  
  
Definition of Done:  
- The retention amount field is implemented and visible on the Reinsurer Selection Screen.  
- The field meets all the specified acceptance criteria.  
- The screen layout is verified to ensure the field is positioned and styled correctly.  
- The functionality is tested to ensure the field accepts the correct input and displays 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 or operations.

# Manage Account Details for Policy Holders

Type: ACCOUNT\_DET

Title: Manage Account Details for Policy Holders  
  
Acceptance Criteria:  
1. The system should allow the user to input and display the following account details:  
 - Account Number  
 - Account Holder Name  
 - Bank Name  
 - Branch  
 - MICR  
 - IFSC Code  
 - Type of Account  
 - Relationship with Premium Payer  
2. The system should provide a checkbox to bypass bank details.  
3. The system should display the status of penny drop transactions as either successful or failed.  
4. The system should fetch bank details based on the IFSC code from the database using the following query:  
 ```sql  
 SELECT BANK\_IFSC, BANK\_NAME, BANK\_BRANCH, BANK\_MICR  
 FROM azbj\_bank\_ifsc\_detail  
 WHERE BANK\_IFSC = :ACCOUNT\_DET.IFSC\_CODE  
 ```  
  
Definition of Done:  
- The user can input and view all required account details.  
- The bypass bank details checkbox is functional.  
- The penny drop transaction status is displayed correctly.  
- The system fetches and displays bank details based on the IFSC code using the specified query.  
- All functionalities are tested and verified to be working as expected.

# 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 "Policy Holder" tab.  
2. The dropdown list should contain 18 predefined options for the relationship between the insured person and the premium payer.  
3. The dropdown list should be positioned correctly on the form and should be easily accessible.  
4. The selected value should be saved and retrievable for future reference.  
  
Definition of Done:  
1. The dropdown list is implemented and visible on the "Policy Holder" tab.  
2. The dropdown list contains 18 predefined options.  
3. The selected value is saved in the database and can be retrieved when needed.  
4. The feature is tested and verified to work as expected.  
  
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.

# Bypass Bank Details Functionality

Type: ACCOUNT\_DET

Title: Bypass Bank Details Functionality  
  
Acceptance Criteria:  
1. When the "Bypass Bank Details" checkbox is checked:  
 - The fields for account number, account holder name, bank name, account relation, account branch, IFSC code, and MICR should be disabled.  
 - The system should clear any existing data in these fields.  
2. When the "Bypass Bank Details" checkbox is unchecked:  
 - The fields for account number, account holder name, bank name, account relation, account branch, IFSC code, and MICR should be enabled.  
 - The system should allow the user to enter or update data in these fields.  
  
Definition of Done:  
- The "Bypass Bank Details" checkbox functionality is implemented and tested.  
- The fields for account number, account holder name, bank name, account relation, account branch, IFSC code, and MICR are correctly enabled or disabled based on the checkbox state.  
- Existing data in the fields is cleared when the checkbox is checked.  
- The user can enter or update data in the fields when the checkbox is unchecked.  
- The feature is reviewed and approved by stakeholders.

# 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, branch, and MICR code.  
2. If the IFSC code does not match any record in the bank details database, the bank name, branch, and MICR code fields should be cleared.  
  
Definition of Done:  
- The user can select a type of account from a list.  
- Upon selection, the system fetches and displays the bank name, branch, and MICR code based on the IFSC code.  
- If no matching IFSC code is found, the bank name, branch, and MICR code fields are cleared.  
- The functionality is tested and verified to work as expected.  
  
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;  
```

# Select Relationship Type for Account

Type: ACCOUNT\_DET

Title: Select Relationship Type for Account  
  
Acceptance Criteria:  
- The relationship type should be selectable from a dropdown list.  
- The dropdown list should contain at least two predefined relationship types.  
- The selected relationship type should be displayed in a specific section of the form.  
- The form should be visually organized with the relationship type field positioned appropriately.  
  
Definition of Done:  
- The dropdown list for selecting the relationship type is implemented and functional.  
- The dropdown list contains at least two predefined options.  
- The selected relationship type is displayed correctly in the form.  
- The form layout is visually organized and user-friendly.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include specific database queries.

# 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 associated with the entered IFSC code.  
2. If the IFSC code is not found in the database, the system should prompt the user with a list of available IFSC codes to choose from.  
  
Definition of Done:  
1. The user can enter an IFSC code in the account details section.  
2. Upon moving to the next field, the system fetches and displays the bank name, branch, and MICR code if the IFSC code is valid.  
3. If the IFSC code is invalid, the system prompts the user with a list of available IFSC codes.  
4. The functionality is tested and verified to ensure it works as expected.  
  
DB queries for Table reference CRUD operations only(With Usage):  
```sql  
SELECT BANK\_NAME, SUBSTR(BANK\_BRANCH, 1, 30), 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;  
```

# Display Same Bank Details for Account

Type: ACCOUNT\_DET

User Story: Display Same Bank Details for Account  
  
Detailed Description:  
As a user, I want to be able to view the details of the same bank account associated with a policy holder, so that I can verify the account information and ensure it matches the policy holder's details.  
  
Acceptance Criteria:  
1. When the "Same Bank Dtls" button is pressed, the system should check if the account number and IFSC code are provided.  
2. If either the account number or IFSC code is missing, the system should display a warning message: "Please enter the Account No and IFSC Code.!" and halt further processing.  
3. If both the account number and IFSC code are provided, the system should:  
 - Create a parameter list and add the following parameters:  
 - Account number  
 - IFSC code  
 - Form name (set to 'BBU')  
 - Insured person ID  
 - Policy holder ID (if the policy holder is not the same as the insured person)  
 - Call the form to display the same bank details using the provided parameters.  
  
Definition of Done:  
- The "Same Bank Dtls" button functionality is implemented and tested.  
- The system correctly validates the presence of account number and IFSC code.  
- Appropriate messages are displayed for missing information.  
- The form to display same bank details is called with the correct parameters when all required information is present.  
- The feature is reviewed and approved by the stakeholders.  
  
DB Queries for Table Reference CRUD Operations:  
- Not applicable as the provided XML content does not include any direct database queries for CRUD operations.

# Manage Requirement Checklist

Type: RHOBR\_CHECKLIST

Title: Manage Requirement Checklist  
  
Acceptance Criteria:  
1. The checklist should display the following fields:  
 - Requirement Code (Read-only)  
 - Requirement Description (Read-only)  
 - Requirement Completely Received (Checkbox)  
 - Date when the requirement was ticked  
 - User who ticked the requirement  
2. The checklist should have the following buttons:  
 - Exit: Closes the checklist view.  
 - Change Status to RHOBR: Changes the status of the selected requirement to RHOBR (Initially hidden).  
3. The checklist should display the Rhobr User ID (Read-only).  
  
Definition of Done:  
1. The checklist interface is implemented and displays all required fields.  
2. The "Exit" button successfully closes the checklist view.  
3. The "Change Status to RHOBR" button is initially hidden and can be made visible based on specific conditions.  
4. The fields for Requirement Code, Requirement Description, and Rhobr User ID are read-only.  
5. The Requirement Completely Received checkbox can be checked or unchecked.  
6. The Date and User fields are updated when the requirement is ticked.  
7. The interface is user-friendly and accessible.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- No specific DB queries provided in the XML content.

# Change Status to RHOBR via Button on RHOBR Checklist Screen

Type: RHOBR\_CHECKLIST

Title: Change Status to RHOBR via Button on RHOBR Checklist Screen  
  
Acceptance Criteria:  
1. The button should be labeled "Change Status to RHOBR".  
2. The button should be located at a specific position on the screen.  
3. The button should be visible and clickable.  
4. Upon clicking the button, the status should be updated to RHOBR.  
  
Definition of Done:  
1. The button is implemented and labeled correctly.  
2. The button is positioned correctly on the screen.  
3. The button is visible and functional.  
4. Clicking the button successfully updates the status to RHOBR.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- No specific DB queries mentioned in the provided XML content.  
  
Block Name: RHOBR\_CHECKLIST

# Exit Button Functionality in RHOBR Checklist

Type: RHOBR\_CHECKLIST

Title: Exit Button Functionality in RHOBR Checklist  
  
Description: As a user, I want to be able to exit the current RHOBR checklist window and navigate to a specific item in the COVERHEAD section, so that I can efficiently manage my workflow without unnecessary steps.  
  
Acceptance Criteria:  
1. When the exit button is pressed, the system should check if the commit form button on the horizontal toolbar is disabled. If it is disabled, the system should enable it.  
2. The system should then navigate to a specific item labeled 'VALIDBUTT' in the 'COVERHEAD' section.  
3. The current RHOBR checklist window should be hidden after the navigation.  
  
Definition of Done:  
- The exit button functionality is implemented and tested.  
- The system correctly enables the commit form button if it is disabled.  
- The system navigates to the 'VALIDBUTT' item in the 'COVERHEAD' section.  
- The current RHOBR checklist window is hidden after the navigation.  
- All acceptance criteria are met and verified through testing.  
  
Dependencies:  
- This user story is dependent on the COVERHEAD block being implemented and accessible.

# Display Policy Member Details

Type: FCF\_DECLINE\_MEMBER

Title: Display Policy Member Details  
  
Acceptance Criteria:  
1. The system should display the following fields for each policy member:  
 - Part ID  
 - Policy Member Name  
 - Date of Birth (DOB)  
 - Serial Number (SR No)  
2. The fields should be displayed in a read-only format, meaning users cannot insert or update the information.  
3. The information should be presented in a tabular format with a scrollbar to navigate through the records if there are more than four entries.  
4. The display should be visually organized with appropriate labels and alignment for each field.  
  
Definition of Done:  
1. The user interface displays the policy member details as specified.  
2. All fields are read-only and cannot be modified by the user.  
3. The layout is user-friendly and visually appealing, with clear labels and proper alignment.  
4. The scrollbar functions correctly, allowing users to navigate through multiple records.  
5. The feature has been tested and verified to meet the acceptance criteria.  
  
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.

# Manage Underwriter Information

Type: UNDERWRITER

Detailed description: As a user, I want to manage underwriter information within a specific section of the application, so that I can view and interact with underwriter details efficiently.  
  
Acceptance criteria:  
1. The user should be able to view the underwriter's user ID and name.  
2. The user should not be able to insert or update the underwriter's user ID and name.  
3. The user should be able to interact with two buttons: "OK" and "Cancel".  
4. The user should be able to input a case to be forwarded to the underwriter, with the input being restricted to uppercase letters and a maximum length of 250 characters.  
  
Definition of Done:  
1. The underwriter's user ID and name are displayed correctly.  
2. The "OK" and "Cancel" buttons are functional and positioned correctly.  
3. The input field for the case to be forwarded to the underwriter is functional, restricted to uppercase, and has a maximum length of 250 characters.  
4. The user interface is consistent with the design specifications and is user-friendly.  
  
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.

# Cancel Button Functionality

Type: UNDERWRITER

Title: Cancel Button Functionality  
  
Acceptance Criteria:  
1. When the "Cancel" button is pressed, the system should navigate to the main screen.  
2. The current window should be hidden upon pressing the "Cancel" button.  
  
Definition of Done:  
- The "Cancel" button is functional and navigates the user to the main screen.  
- The current window is hidden when the "Cancel" button is pressed.  
- The feature has been tested and verified to work as expected.

# Forward Case to Selected Underwriter

Type: UNDERWRITER

Detailed description: As a user, I want to forward a case to a selected underwriter, ensuring that the selected underwriter is valid and not the same as the current user, so that the case can be processed by the appropriate underwriter.  
  
Acceptance criteria:  
1. The user must select an underwriter to whom the case needs to be forwarded.  
2. The selected underwriter's ID must be exactly 8 characters long.  
3. The selected underwriter cannot be the same as the current user.  
4. The system should check if the selected underwriter exists in the list of underwriters.  
5. If the selected underwriter does not exist in the list, an error message should be displayed.  
6. If the selected underwriter exists, the system should enable the option to save and exit the proposal, and display a warning message indicating that the case will be forwarded to the selected underwriter.  
  
Definition of Done:  
- The user can successfully select an underwriter and forward the case.  
- Appropriate error messages are displayed for invalid selections.  
- The system verifies the existence of the selected underwriter.  
- The option to save and exit the proposal is enabled when a valid underwriter is selected.  
- The user is informed that the case will be forwarded to the selected underwriter.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Not applicable as the provided XML content does not include specific CRUD operations or direct database queries.

# Manage Decline Process Details

Type: DECLINE

Title: Manage Decline Process Details  
  
Acceptance Criteria:  
1. The system should allow the user to input the following details:  
 - Father's title, first name, middle name, and surname.  
 - Nominee's title, first name, middle name, and surname.  
 - Subclass business.  
 - Coffer amount.  
 - Remarks.  
 - City.  
 - Declined reason.  
 - Comments.  
2. The system should validate the input for the following fields:  
 - Father's surname and nominee's surname should not exceed 50 characters.  
 - Remarks should not exceed 100 characters.  
 - City should not exceed 40 characters.  
 - Declined reason should not exceed 150 characters.  
 - Comments should not exceed 255 characters.  
3. The system should provide a list of valid options for the following fields:  
 - Father's title and nominee's title.  
 - Subclass business.  
 - Declined reason.  
4. The system should allow the user to add a requirement using a button labeled "Add Requirement <Alt+4>".  
5. The system should provide an "Exit" button to close the form.  
  
Definition of Done:  
- The user can successfully input and save all required details.  
- The system validates the input according to the specified criteria.  
- The system provides dropdown lists for fields with predefined options.  
- The "Add Requirement" button functions as expected.  
- The "Exit" button closes the form.  
  
DB queries for Table reference CRUD operations only (With Usage):  
1. Decline Comments:  
 ```sql  
 SELECT SYS\_DESC, sys\_code   
 FROM azbj\_system\_constants   
 WHERE sys\_type = 'DECL\_REASN'   
 AND end\_date IS NULL;  
 ```  
 - Usage: To populate the list of decline comments.  
  
2. Districts:  
 ```sql  
 SELECT b.district\_name   
 FROM azbj\_states a, azbj\_districts b   
 WHERE a.scode = b.scode   
 AND a.state\_name = :parameter.V\_STATE;  
 ```  
 - Usage: To populate the list of districts based on the selected state.  
  
3. Decline Reasons:  
 ```sql  
 SELECT SYS\_CODE, SYS\_DESC   
 FROM azbj\_system\_constants   
 WHERE sys\_type = 'DECLN\_TYPE';  
 ```  
 - Usage: To populate the list of decline reasons.

# Implement Dropdown for Father's Title Selection

Type: DECLINE

Title: Implement Dropdown for Father's Title Selection  
  
Acceptance Criteria:  
1. The list of titles should be displayed in a dropdown menu.  
2. The dropdown menu should be positioned at the specified location on the form.  
3. The dropdown menu should have a white background and black text.  
4. The dropdown menu should be labeled "Before Title" and the label should be bold and positioned correctly.  
5. The form should be displayed in a modal window with specified dimensions and a gray background.  
  
Definition of Done:  
1. The dropdown menu for selecting the father's title is implemented and functional.  
2. The dropdown menu is correctly positioned and styled as per the requirements.  
3. The label "Before Title" is displayed correctly and is bold.  
4. The form is displayed in a modal window with the specified dimensions and background color.  
5. All acceptance criteria are met and verified through testing.

# City Field LOV Functionality

Type: DECLINE

Title: City Field LOV Functionality  
  
Acceptance Criteria:  
1. When the user double-clicks on the city field, a list of values (LOV) should be displayed.  
2. The LOV should fetch and display the district names based on the state selected by the user.  
3. The LOV should be populated using the following query:  
 ```sql  
 SELECT b.district\_name   
 FROM azbj\_states a, azbj\_districts b  
 WHERE a.scode = b.scode  
 AND a.state\_name = :parameter.V\_STATE;  
 ```  
  
Definition of Done:  
1. The city field should be interactive and respond to double-click events.  
2. The LOV should display the correct district names based on the state selected.  
3. The functionality should be tested and verified to ensure it works as expected.  
4. The feature should be documented for future reference and user guidance.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- The query provided is used to fetch district names based on the selected state.

# Nominee Title Selection

Type: DECLINE

Title: Nominee Title Selection  
  
Acceptance Criteria:  
1. The title selection should be presented as a dropdown list with predefined options.  
2. The dropdown list should be positioned appropriately within the form.  
3. The selected title should be displayed in a specific area of the form.  
4. The form should have a prompt indicating where the title should be selected.  
5. The form should maintain a consistent look and feel with specified font styles and colors.  
  
Definition of Done:  
1. The dropdown list for selecting the nominee's title is implemented and functional.  
2. The dropdown list is correctly positioned within the form.  
3. The selected title is displayed in the designated area.  
4. The prompt for the title selection is visible and correctly positioned.  
5. The form's appearance adheres to the specified design guidelines, including font styles and colors.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- No specific database queries are mentioned in the provided XML content.

# Exit Button Functionality with Validation

Type: DECLINE

Title: Exit Button Functionality with Validation  
  
Acceptance Criteria:  
1. If the "Decline Code" field is set to '03' and the "Comments" field is empty, the system should prompt the user to fill in the "Comments" field.  
2. The system should display a message instructing the user to provide comments.  
3. The focus should be set to the "Comments" field if it is empty and the "Decline Code" is '03'.  
4. Upon successful validation, the system should navigate to the next section and hide the current window.  
5. The form status should be updated to 'N' to indicate that the form is no longer active.  
  
Definition of Done:  
- The user is prompted to fill in the "Comments" field if it is empty and the "Decline Code" is '03'.  
- A message is displayed to the user instructing them to provide comments.  
- The focus is set to the "Comments" field when required.  
- The system successfully navigates to the next section and hides the current window upon validation.  
- The form status is updated to 'N' after the user exits the form.

# Dropdown List Item for 'Prop at RI stage'

Type: DECLINE

Title: Dropdown List Item for 'Prop at RI stage'  
  
Acceptance Criteria:  
- The dropdown list should have two predefined options.  
- The label for the dropdown list should be 'Prop at RI stage'.  
- The dropdown list should be positioned at coordinates (563, 102) on the form.  
- The dropdown list should have a width of 85 and a height of 20.  
- The font for the dropdown list should be 'Times New Roman' with a size of 8 and a plain style.  
- The label font should be 'Tahoma' with a size of 8 and a plain style.  
- The text in the dropdown list should be in uppercase.  
- The background color of the dropdown list should be white, and the foreground color should be black.  
  
Definition of Done:  
- The dropdown list item is implemented and visible in the DECLINE section of the form.  
- The dropdown list meets all the specified acceptance criteria.  
- The dropdown list is tested and verified to ensure it functions correctly and displays the predefined options.  
  
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 Business Sub-Class Dropdown

Type: DECLINE

Title: Implement Business Sub-Class Dropdown  
  
Acceptance Criteria:  
1. The list of business sub-classes should be displayed in a dropdown menu.  
2. The dropdown menu should contain exactly 5 elements.  
3. The selected business sub-class should be displayed in uppercase letters.  
4. The dropdown menu should be positioned at the specified coordinates on the screen.  
5. The dropdown menu should have a white background and black text.  
6. The prompt for the dropdown menu should read "Sub Class Business" and be displayed in bold font.  
  
Definition of Done:  
1. The dropdown menu for selecting a business sub-class is implemented and functional.  
2. The dropdown menu contains exactly 5 elements.  
3. The selected business sub-class is displayed in uppercase letters.  
4. The dropdown menu is correctly positioned on the screen.  
5. The dropdown menu has a white background and black text.  
6. The prompt "Sub Class Business" is displayed in bold font next to the dropdown menu.  
  
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 reason for declining or postponing an action from a predefined list

Type: DECLINE

Detailed description: As a user, I want to be able to select a reason for declining or postponing an action from a predefined list, so that I can provide a standardized reason for the decline or postponement.  
  
Acceptance criteria:  
1. When the user double-clicks on the "Declined/Postpone Reason" field, a list of predefined reasons should be displayed.  
2. The list of predefined reasons should be fetched from the database table `azbj\_system\_constants` where `sys\_type` is 'DECLN\_TYPE'.  
3. The "Declined/Postpone Reason" field should only accept values from this predefined list.  
4. If the user selects a reason with a specific code (e.g., '03'), the "Comments" field should become visible, enabled, and navigable.  
  
Definition of Done:  
- The "Declined/Postpone Reason" field displays a list of predefined reasons upon double-click.  
- The list of reasons is fetched from the database and only valid reasons can be selected.  
- The "Comments" field's visibility, enabled state, and navigability are correctly toggled based on the selected reason code.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- `select SYS\_CODE, SYS\_DESC from azbj\_system\_constants where sys\_type = 'DECLN\_TYPE'` (Used to fetch the list of predefined decline/postpone reasons).

# Add Requirement Button Functionality

Type: DECLINE

Title: Add Requirement Button Functionality  
  
Acceptance Criteria:  
1. When the "Add Requirement" button is pressed, the system should navigate to the "Decline Request" section.  
2. The system should move to the last record in the "Decline Request" section.  
3. The requirement type should be set to 'FR-REQ'.  
4. The system should display a list of values (LOV) for further requirements.  
5. If an error occurs, an error message should be displayed with the error details.  
  
Definition of Done:  
- The "Add Requirement" button is functional and performs the specified actions.  
- The requirement type is correctly set to 'FR-REQ'.  
- The LOV for further requirements is displayed correctly.  
- Error handling is implemented, and error messages are displayed as needed.  
  
DB queries for Table reference CRUD operations only (With Usage):  
```sql  
SELECT sys\_code AS DOC, UPPER(TRIM(sys\_desc)) AS DESCRIPTION  
FROM azbj\_system\_constants  
WHERE sys\_type = 'FR\_REQ'  
AND NVL(TO\_DATE(:susac.sa\_daterecd, 'DD/MM/RRRR'), pme\_api.opus\_date) BETWEEN start\_date AND NVL(end\_date, TO\_DATE('01-jan-3000', 'dd/mm/rrrr'))  
ORDER BY DESCRIPTION;  
```  
- This query is used to fetch the list of further requirements to be displayed in the LOV.

# View and Manage Underwriting Details

Type: MULTIPLE\_UW\_DETAIL

Title: View and Manage Underwriting Details  
  
Acceptance Criteria:  
1. The section should display up to 5 underwriting records at a time.  
2. Each record should include the following fields:  
 - Underwriting Number (read-only)  
 - User ID (read-only)  
 - Underwriting Date (read-only)  
3. The section should have a scrollbar to navigate through additional records if more than 5 exist.  
4. The section should allow for the insertion and updating of records, but the specific fields (Underwriting Number, User ID, Underwriting Date) should remain read-only.  
  
Definition of Done:  
1. The section is implemented and displays up to 5 underwriting records at a time.  
2. The fields Underwriting Number, User ID, and Underwriting Date are displayed as read-only.  
3. The section includes a functional scrollbar for navigating through records.  
4. The section allows for the insertion and updating of records, adhering to the read-only constraints for specific fields.  
5. The implementation is tested and verified to meet the acceptance criteria.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- No specific DB queries provided in the XML content.

# Display Pre-Questionnaire Section

Type: PRE\_QUESTIONNAIRE

Title: Display Pre-Questionnaire Section  
  
Acceptance Criteria:  
1. The pre-questionnaire section should display the following fields:  
 - Pre Questionnaire  
 - Pre Req No  
 - Pre Proposal No  
2. All fields should be displayed in uppercase.  
3. The fields should be read-only and not allow any insert or update actions.  
4. The section should be visually organized with appropriate labels and spacing for clarity.  
  
Definition of Done:  
1. The pre-questionnaire section is implemented and displays the specified fields.  
2. The fields are in uppercase and are read-only.  
3. The section is visually organized and user-friendly.  
4. The implementation is tested and verified to meet the acceptance criteria.  
  
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 Manage Simultaneous Proposals

Type: SIMUTANEOUS\_PROPOSALS

Title: View and Manage Simultaneous Proposals  
  
Acceptance Criteria:  
1. The user should be able to view a list of simultaneous proposals with the following details:  
 - Policy Reference  
 - Partner ID  
 - Contract ID  
 - Request Code  
 - Request Description  
 - Request Status  
 - Called Date  
 - Received Date  
 - IP Number  
 - Request Type  
2. The user should be able to navigate through the list of proposals using a scrollbar.  
3. The user should be able to exit the form using an "Exit" button.  
  
Definition of Done:  
- The user interface displays all the required fields for simultaneous proposals.  
- The scrollbar allows the user to navigate through the list of proposals.  
- The "Exit" button functions correctly and allows the user to exit the form.  
- All fields are displayed with appropriate labels and are read-only where necessary.  
- The form is tested and verified to ensure all functionalities work as expected.  
  
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: SIMUTANEOUS\_PROPOSALS

Title: Exit Button Functionality  
  
Acceptance Criteria:  
1. When the 'Exit' button is pressed, the system should navigate to the 'med\_uw' section.  
2. The current window titled 'Rhobr Check List' should be hidden upon pressing the 'Exit' button.  
  
Definition of Done:  
- The 'Exit' button should be functional and perform the navigation to the 'med\_uw' section.  
- The 'Rhobr Check List' window should be hidden when the 'Exit' button is pressed.  
- The functionality should be tested and verified to ensure it works as expected.

# View IP Numbers in Dropdown Menu

Type: SIMUTANEOUS\_PROPOSALS

Title: View IP Numbers in Dropdown Menu  
  
Acceptance Criteria:  
1. The dropdown menu should display a list of 10 IP numbers.  
2. The dropdown menu should be positioned at the top of the Simultaneous Proposals section.  
3. The dropdown menu should be read-only, meaning users cannot insert or update the IP numbers.  
4. The dropdown menu should have a white background and be styled with a bold Tahoma font.  
  
Definition of Done:  
1. The dropdown menu is implemented and displays 10 IP numbers.  
2. The dropdown menu is correctly positioned and styled as specified.  
3. The dropdown menu is read-only and users cannot modify the IP numbers.  
4. The feature is tested and verified to meet the acceptance criteria.

# Manage Policy Member Loading Details

Type: FCF\_LOADING\_DETAILS

Title: Manage Policy Member Loading Details  
  
Acceptance Criteria:  
1. The system should display the following fields for each policy member:  
 - IP Number (Read-only)  
 - Policy Member Name (Read-only)  
 - Part ID (Read-only)  
 - Relation (Editable)  
 - Cover Code (Read-only)  
 - Cover Description (Read-only)  
 - ML Percentage (Read-only)  
 - ML Amount (Read-only)  
 - Annual ML Amount (Read-only)  
2. The "Back" button should be available to navigate back to the previous screen.  
  
Definition of Done:  
1. The user can view all the specified fields for each policy member.  
2. The "Relation" field should be editable, while all other fields should be read-only.  
3. The "Back" button should function correctly, allowing the user to return to the previous screen.  
4. The layout should be user-friendly and match the specified positions and dimensions for each field.  
5. All changes should be saved and reviewed for accuracy and completeness.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Not applicable as the provided XML content does not include specific database queries or operations.

# Dynamic Field Enable/Disable Based on Relationship Type

Type: FCF\_LOADING\_DETAILS

Title: Dynamic Field Enable/Disable Based on Relationship Type  
  
Acceptance Criteria:  
1. When the user selects "Husband" as the relationship type, the fields for insurance and annual income should be disabled.  
2. When the user selects any other relationship type, the fields for insurance and annual income should be enabled.  
  
Definition of Done:  
- The form dynamically adjusts the enabled/disabled state of the insurance and annual income fields based on the selected relationship type.  
- The changes are reflected immediately upon selection without requiring a page refresh.  
- The functionality is tested and verified to work as expected in different scenarios.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include any direct database operations.

# Navigation and Window Management via 'Back' Button

Type: FCF\_LOADING\_DETAILS

Title: Navigation and Window Management via 'Back' Button  
  
Acceptance Criteria:  
1. When the 'Back' button is pressed, the system should navigate to the 'COVERS' section.  
2. The 'C\_OFFER' window should be hidden upon pressing the 'Back' button.  
  
Definition of Done:  
- The 'Back' button successfully navigates to the 'COVERS' section.  
- The 'C\_OFFER' window is hidden when the 'Back' button is pressed.  
- 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 XML content does not include any database CRUD operations.

# View and Select Test Details from Further Requirements

Type: DECLINE\_REQ

Detailed description: As a user, I want to view and select test details from a list of further requirements, so that I can accurately decline requests based on specific criteria.  
  
Acceptance criteria:  
1. The user should be able to view a list of test numbers and descriptions.  
2. The list of test numbers and descriptions should be fetched from the database based on specific criteria.  
3. The user should be able to select a test number from the list.  
4. The selected test number should be validated against the list of available test numbers.  
5. The test description should be displayed but not editable.  
6. The request type should be displayed.  
  
Definition of Done:  
1. The user interface displays the test number and description fields.  
2. The test number field allows selection from a list of further requirements.  
3. The test description field is displayed as read-only.  
4. The request type field is displayed.  
5. The list of further requirements is fetched from the database using the specified query.  
6. The functionality is tested and verified to meet the acceptance criteria.  
  
DB queries for Table reference CRUD operations only(With Usage):  
```sql  
SELECT sys\_code AS DOC, UPPER(TRIM(sys\_desc)) AS DESCRIPTION  
FROM azbj\_system\_constants  
WHERE sys\_type = 'FR\_REQ'  
AND NVL(TO\_DATE(:susac.sa\_daterecd, 'DD/MM/RRRR'), pme\_api.opus\_date) BETWEEN start\_date AND NVL(end\_date, TO\_DATE('01-jan-3000', 'dd/mm/rrrr'))  
ORDER BY DESCRIPTION;  
```  
  
This query is used to fetch the list of further requirements based on the specified criteria.

# User Interaction with Test Number Field

Type: DECLINE\_REQ

Title: User Interaction with Test Number Field  
  
Acceptance Criteria:  
1. The test number field should display a list of valid test numbers when the user interacts with it.  
2. The list of valid test numbers should be fetched from the database based on specific criteria.  
3. The list should be displayed when the user double-clicks on the test number field or uses a specific key combination.  
4. The list should be filtered to show only those test numbers that are valid within a specific date range.  
  
Definition of Done:  
1. The test number field displays a list of valid test numbers upon user interaction.  
2. The list of test numbers is fetched from the database and meets the specified criteria.  
3. The functionality is tested and verified to work as expected.  
4. The feature is documented and ready for user acceptance testing.  
  
DB queries for Table reference CRUD operations only(With Usage):  
```sql  
SELECT sys\_code AS DOC, UPPER(TRIM(sys\_desc)) AS DESCRIPTION  
FROM azbj\_system\_constants  
WHERE sys\_type = 'FR\_REQ'  
AND NVL(TO\_DATE(:susac.sa\_daterecd, 'DD/MM/RRRR'), pme\_api.opus\_date) BETWEEN start\_date AND NVL(end\_date, TO\_DATE('01-jan-3000', 'dd/mm/rrrr'))  
ORDER BY DESCRIPTION;  
```  
  
This query is used to fetch the list of valid test numbers based on the specified criteria.

# Document Requirements Section

Type: DOCS

Title: Document Requirements Section  
  
Acceptance Criteria:  
1. The document requirements section should display the type of requirement and its description.  
2. The type and description fields should be read-only.  
3. There should be a "Close" button that allows the user to close the document requirements section.  
4. There should be a "Reason" button that allows the user to view the reasons related to the document requirements.  
  
Definition of Done:  
1. The document requirements section is implemented and displays the type and description of requirements.  
2. The type and description fields are set to read-only.  
3. The "Close" button is functional and closes the document requirements section.  
4. The "Reason" button is functional and displays the reasons related to the document requirements.  
5. The implementation is tested and verified to meet the acceptance criteria.

# Close Button Functionality

Type: DOCS

Title: Close Button Functionality  
  
Acceptance Criteria:  
1. When the "Close" button is pressed, the system should navigate to the item labeled "further\_req.rcu".  
2. The current window labeled "docs" should be hidden upon pressing the "Close" button.  
  
Definition of Done:  
- The "Close" button is functional and performs the specified actions.  
- The navigation to the item "further\_req.rcu" is successful.  
- The window labeled "docs" is hidden after the button is pressed.  
- The feature is tested and verified to work as expected.

# View Reasons for Rejected Applications

Type: DOCS

Title: View Reasons for Rejected Applications  
  
Acceptance Criteria:  
1. When the "Reason" button is pressed, the system should:  
 - Retrieve the reason for rejection from the `azbj\_phub\_add\_req\_tracker` table where:  
 - `accept\_req` is 'N'  
 - `application\_no` matches either `IP\_VERF\_NO` or `IP\_SIGN\_CARD\_NO`  
 - `ALTERNATE\_REQ` matches the specified request type.  
 - Display the retrieved reasons in the "REASON" section.  
  
Definition of Done:  
- The "Reason" button should be functional and retrieve the correct data based on the specified conditions.  
- The reasons should be displayed in the "REASON" section without any errors.  
- The system should handle multiple records and display them sequentially.  
  
SQL Query for Reference:  
```sql  
-- Retrieve reasons for rejected applications  
SELECT reason   
FROM azbj\_phub\_add\_req\_tracker  
WHERE accept\_req = 'N'  
AND application\_no = TO\_CHAR(NVL(:INSURED\_PERSON.IP\_VERF\_NO, :INSURED\_PERSON.IP\_SIGN\_CARD\_NO))  
AND ALTERNATE\_REQ = :DOCS.REQ\_TYPE;  
```  
  
Block Name: DOCS

# View and Manage Policy Details

Type: BALIC\_RISK\_PH

Title: View and Manage Policy Details  
  
Acceptance Criteria:  
1. The policy details should include the following fields:  
 - Policy Number  
 - Contract ID  
 - Sum Assured  
 - Date of Risk  
 - Product Name  
 - Annual Premium  
 - Policy Status  
 - Time Stamp  
 - Paid By  
 - Main Cover Rated Up  
 - Reason for Rating  
 - Value  
 - Rider Cover Rated Up  
 - Adverse Factor Details  
 - View Image  
 - BT  
 - PT  
  
2. The fields should be displayed in a user-friendly manner with appropriate labels and formatting.  
  
3. The fields should be read-only and not allow any updates or insertions.  
  
4. The "Adverse Factor Details" and "View Image" should be represented as buttons that, when clicked, provide additional information or images related to the policy.  
  
Definition of Done:  
- The policy details are displayed with all the specified fields.  
- The fields are formatted and labeled correctly.  
- The fields are read-only.  
- The "Adverse Factor Details" and "View Image" buttons are functional and provide the expected additional information or images.  
  
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 Image Button Functionality

Type: BALIC\_RISK\_PH

Title: View Image Button Functionality  
  
Acceptance Criteria:  
1. When the "Images" button is pressed, the system should check if the policy number is not null.  
2. If the policy number is not null, the system should generate a URL using the `azbj\_encrypt\_dms\_link` function with the parameters 'NB', 'POLICY\_DOCS', and the policy number.  
3. If the generated URL is not null, the system should open the URL in a web browser.  
4. If there is an error during the URL generation or opening process, the system should display a message indicating that there is an issue with the URL.  
  
Definition of Done:  
- The "Images" button functionality is implemented and tested.  
- The system correctly checks for a non-null policy number.  
- The URL is generated and opened correctly when the policy number is valid.  
- Appropriate error messages are displayed when there is an issue with the URL.  
  
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 the database.

# Manage Doctor Signatures

Type: DOCTOR\_SIGNATURE

Detailed description: As a user, I want to manage doctor signatures within the application, so that I can ensure the authenticity and verification of documents.  
  
Acceptance criteria:  
1. The user should be able to see a button labeled "Signature" to initiate the process of adding a doctor's signature.  
2. The user should be able to see a button labeled "Back" to navigate back to the previous screen.  
3. The user should be able to see a button labeled "Reset" to clear any input or changes made in the current session.  
4. The user should be able to view an image area labeled "Signature" where the doctor's signature will be displayed. This area should have horizontal and vertical scrollbars for better navigation of the signature image.  
5. The signature image area should not allow insertion or updates directly by the user to maintain data integrity.  
  
Definition of Done:  
1. The "Signature" button is functional and initiates the process of adding a doctor's signature.  
2. The "Back" button successfully navigates the user to the previous screen.  
3. The "Reset" button clears any input or changes made in the current session.  
4. The signature image area displays the doctor's signature correctly and allows scrolling for better navigation.  
5. The signature image area does not allow direct insertion or updates by the user.  
  
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.

# Navigate Back from Doctor Signature Screen

Type: DOCTOR\_SIGNATURE

Title: Navigate Back from Doctor Signature Screen  
  
Acceptance Criteria:  
1. When the "Back" button is clicked, the system should navigate to the previous screen.  
2. The button should be labeled "Back" and be positioned appropriately on the screen.  
3. The button should have a specific appearance, including a gray background and black text.  
  
Definition of Done:  
1. The "Back" button is implemented and visible on the doctor signature screen.  
2. Clicking the "Back" button successfully navigates the user to the previous screen.  
3. The button's appearance matches the specified design (gray background, black text).  
4. The functionality is tested and confirmed to work as expected.

# Reset Doctor Signature Form

Type: DOCTOR\_SIGNATURE

Title: Reset Doctor Signature Form  
  
Acceptance Criteria:  
1. When the reset button is pressed, the system should navigate to the doctor signature section.  
2. The system should clear all the fields in the doctor signature section without validating the current data.  
3. The system should then navigate to the doctor code field and set its value to null.  
  
Definition of Done:  
- The reset button should be functional and perform the described actions.  
- The doctor signature section should be cleared without any validation errors.  
- The doctor code field should be empty after the reset action.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include any direct database operations.

# Verify Doctor's Signature

Type: DOCTOR\_SIGNATURE

Detailed description: As a user, I want to be able to verify a doctor's signature by entering the doctor's code, so that I can ensure the authenticity of the doctor's identity.  
  
Acceptance criteria:  
1. If the doctor code is provided:  
 - The system should set the message level to 25 to suppress unnecessary messages.  
 - The system should filter the doctor signature records based on the provided doctor code and a fixed sub-code '1234'.  
 - The system should navigate to the doctor signature section and execute the query to fetch the relevant signature data.  
 - The system should reset the message level to 0 after the query execution.  
2. If the doctor code is not provided:  
 - The system should display a warning message prompting the user to enter the doctor code.  
3. If no data is found for the provided doctor code:  
 - The system should display a warning message indicating that the doctor code does not exist and prompt the user to check the code.  
4. If any other error occurs:  
 - The system should display a warning message with the error details.  
  
Definition of Done:  
- The user can enter a doctor code and verify the doctor's signature.  
- Appropriate messages are displayed based on the conditions mentioned in the acceptance criteria.  
- The system handles exceptions gracefully and provides meaningful feedback to the user.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- The following query is used to fetch the doctor signature based on the provided doctor code and a fixed sub-code '1234':  
 ```sql  
 SELECT SIGNATURE  
 INTO :DOCTOR\_SIGNATURE.SIGNATURE  
 FROM azbj\_panel\_doctor\_ref a, azbj\_signature b  
 WHERE a.panel\_doctor = b.ic\_code  
 AND b.sub\_ic\_code = '1234'  
 AND ic\_code = :CONTROL.doctor\_code;  
 ```  
- The following cursor is used to iterate through the doctor signature records:  
 ```sql  
 CURSOR c1 IS  
 SELECT ic\_code, signature  
 FROM azbj\_panel\_doctor\_ref a, azbj\_signature b  
 WHERE a.panel\_doctor = b.ic\_code  
 AND b.sub\_ic\_code = '1234'  
 AND ic\_code = :CONTROL.doctor\_code  
 ORDER BY ic\_code;  
 ```

# View and Navigate Business Units

Type: BBU\_Q

Detailed description: As a user, I want to view and navigate through a list of business units, each with a unique identifier and description, so that I can easily manage and reference them.  
  
Acceptance criteria:  
1. The list should display up to 5 records at a time.  
2. Each record should include:  
 - A unique number (Q\_NO) displayed as a read-only field.  
 - A description (Q\_DESC) that is not editable and has a maximum length of 300 characters.  
 - A business unit code (Q\_BBU) that is editable and must be in uppercase.  
3. The description field should be visually distinct with a gray background and black text.  
4. The business unit code field should also have a gray background and black text.  
5. Navigation between fields should be intuitive, with the ability to move from the business unit code to the description and vice versa.  
  
Definition of Done:  
- The user can view up to 5 business unit records at a time.  
- The unique number field is read-only.  
- The description field is non-editable and visually distinct.  
- The business unit code field is editable and enforces uppercase input.  
- Navigation between fields is functional and intuitive.  
  
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.

# Validate Q\_BBU Field Input

Type: BBU\_Q

Title: Validate Q\_BBU Field Input  
  
Acceptance Criteria:  
1. When the user enters a value in the "Q\_BBU" field, the system should check if the value is either 'Y' or 'N'.  
2. If the value entered is not 'Y' or 'N', the system should display an error message: "Kindly enter Y OR N in Answer".  
3. The validation should only occur if the "Q\_BBU" field is not empty.  
  
Definition of Done:  
1. The validation logic is implemented and tested.  
2. The error message is displayed correctly when an invalid value is entered.  
3. The system allows only 'Y' or 'N' values in the "Q\_BBU" field.  
4. The feature is reviewed and approved by the stakeholders.

# Manage SOM Details Interface

Type: SOM

Title: Manage SOM Details Interface  
  
Acceptance Criteria:  
1. The interface should have a "Cancel" button to discard any changes made.  
2. The interface should have an "OK" button to submit the SOM details.  
3. The interface should display a checkbox labeled "Send to Hub Head for Approval (only in case SOM does not exist)" which, when checked, indicates that the SOM details need to be sent for approval.  
4. The interface should display the following fields:  
 - SOM ID: A read-only field that displays the SOM ID.  
 - HUB ID: A read-only field that displays the HUB ID.  
 - HUB Incharge Name: A read-only field that displays the name of the HUB incharge.  
 - SOM Name: A read-only field that displays the name of the SOM.  
5. The SOM ID and SOM Name fields should be validated against a predefined list to ensure they are correct.  
  
Definition of Done:  
- The user interface is implemented with all the specified fields and buttons.  
- The "Cancel" button discards any changes made by the user.  
- The "OK" button submits the SOM details.  
- The checkbox for sending to the Hub Head for approval functions as expected.  
- All fields display the correct information and are read-only where specified.  
- The SOM ID and SOM Name fields are validated against a predefined list.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- The following query is used to fetch SOM details based on the HUB:  
 ```sql  
 SELECT som\_id, som\_name   
 FROM azbj\_som\_ID\_detail  
 WHERE hub = (  
 SELECT substr(received\_user, 1, 3)   
 FROM azbj\_phub\_tracker   
 WHERE application\_no = TO\_CHAR(NVL(:INSURED\_PERSON.IP\_VERF\_NO, :INSURED\_PERSON.IP\_SIGN\_CARD\_NO))  
 );  
 ```

# Cancel Button Functionality

Type: SOM

Title: Cancel Button Functionality  
  
Acceptance Criteria:  
- When the "Cancel" button is pressed, the current window should be hidden.  
- The associated view should also be hidden upon pressing the "Cancel" button.  
  
Definition of Done:  
- The "Cancel" button is functional and hides both the current window and the associated view when pressed.  
- The functionality has been tested and verified to ensure that the window and view are hidden as expected.

# Validate and Update Proposal Status on OK Button Press

Type: SOM

Detailed description: As a user, I want to ensure that when I press the "OK" button, the system validates the presence of a specific flag and updates the proposal status accordingly, so that I can proceed with the necessary actions based on the validation results.  
  
Acceptance criteria:  
1. If the specific flag is not set to 'Y', the system should display an error message indicating that the SOM is not present and the case should be referred to the Hub Incharge for approval.  
2. The system should update the proposal status to 'FR-AR' and set the action to 'F'.  
3. The system should navigate to the 'SAVE' section and execute the associated trigger.  
4. The system should hide the current window and view.  
5. If any error occurs during the process, the system should display a warning message with the error details.  
  
Definition of Done:  
- The "OK" button functionality is implemented and tested.  
- The system correctly validates the flag and updates the proposal status.  
- Appropriate error and warning messages are displayed as per the acceptance criteria.  
- The navigation and trigger execution work as expected.  
- The window and view are hidden after the process completes.  
- All edge cases and exceptions are handled gracefully.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Not applicable as the provided XML content does not include any direct database queries for CRUD operations.

# Send Request to Hub Head for Approval

Type: SOM

Title: Send Request to Hub Head for Approval  
  
Acceptance Criteria:  
1. When the checkbox is checked, the system should:  
 - Retrieve the `hub\_incharge\_opus\_id` and `hub\_incharge\_name` from the database.  
 - The retrieval should be based on the `branch\_code` and `application\_no` of the insured person.  
 - If no data is found, the `hub\_id` and `hub\_incharge\_name` should be set to NULL.  
 - If any other error occurs during the retrieval, an error message should be displayed indicating an issue with fetching the hub incharge details.  
  
Definition of Done:  
- The checkbox functionality should be implemented and tested.  
- The system should correctly retrieve and display the hub incharge details when the checkbox is checked.  
- Appropriate error handling should be in place for scenarios where no data is found or other errors occur.  
- The feature should be tested and verified to ensure it meets the acceptance criteria.  
  
DB queries for Table reference CRUD operations only(With Usage):  
```sql  
SELECT hub\_incharge\_opus\_id,  
 hub\_incharge\_name  
 INTO :som.hub\_id,  
 :som.hub\_incharge\_name  
 FROM azbj\_som\_hub\_mapping a, azbj\_phub\_tracker b  
 WHERE a.branch\_code = b.branch\_code  
 AND b.application\_no = TO\_CHAR(NVL(:insured\_person.ip\_verf\_no, :insured\_person.ip\_sign\_card\_no))  
 AND NVL(a.active\_flag, 'Y') = 'Y';  
```

# Display and Interaction of SOM ID Field

Type: SOM

Title: Display and Interaction of SOM ID Field  
  
Acceptance Criteria:  
1. The "SOM ID" field should be displayed with a white background and black text.  
2. The field should be positioned at coordinates (89, 77) on the screen.  
3. The font used for the "SOM ID" field should be Tahoma, with a font size of 8 and a demilight weight.  
4. The prompt for the "SOM ID" field should be "SOM ID", displayed in bold Tahoma font with a size of 9.  
5. The "SOM ID" field should not allow insertions or updates.  
6. Double-clicking on the "SOM ID" field should not trigger any action.  
  
Definition of Done:  
- The "SOM ID" field is displayed according to the specified design and positioning.  
- The field's prompt is correctly displayed.  
- The field does not allow insertions or updates.  
- Double-clicking on the field does not perform any action.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include any specific database queries.

# Manage Image Details and Associated Questions

Type: BBU\_IMAGE\_DTLS

Title: Manage Image Details and Associated Questions  
  
Acceptance Criteria:  
1. When a new record is created, if the question description is not null:  
 - If the global variable `gv\_show\_image` is true, the system should load the associated images.  
 - If the question description is null, the system should navigate to the previous record and display a message prompting the user to proceed further.  
2. The system should allow navigation and execution of specific actions using function keys (F0 and F1).  
  
Definition of Done:  
- The user can create, read, update, and delete image details and associated questions.  
- The system correctly handles the display and loading of images based on the question description.  
- Function keys (F0 and F1) perform the specified navigation and actions.  
- All functionalities are tested and verified to work as expected.  
  
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 Checkbox State to Trigger Actions Based on Conditions

Type: BBU\_IMAGE\_DTLS

Title: Manage Checkbox State to Trigger Actions Based on Conditions  
  
Acceptance Criteria:  
1. When the checkbox is checked:  
 - If the dependency flag is 'N' and the dependent question is not null, the system should populate the group question using the provided group question number, serial number, and dependent question.  
2. When the checkbox is unchecked:  
 - If the dependency flag is 'N' and the dependent question is not null, the system should delete the group question using the provided group question number, serial number, and dependent question.  
  
Definition of Done:  
- The checkbox state change triggers the appropriate action (populate or delete) based on the specified conditions.  
- The system correctly handles the population or deletion of the group question as per the conditions.  
- The functionality 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 logic involves procedural calls and conditions specific to Oracle Forms.

# Validate and Process Answer Input

Type: BBU\_IMAGE\_DTLS

Title: Validate and Process Answer Input  
  
Acceptance Criteria:  
1. If the answer is 'Y' and there is a dependent question, the system should populate the related group question.  
2. If the answer is 'N' and there is a dependent question, the system should delete the related group question.  
3. If the answer is neither 'Y' nor 'N', the system should display an error message prompting the user to enter 'Y' for Yes or 'N' for No.  
4. When the answer is 'Y', the system should update the requirement description with the value of the requirement if Yes.  
5. When the answer is 'N', the system should update the requirement description with the value of the requirement if No.  
6. The system should handle the population and deletion of related group questions based on the answer and the state of the dependent question and question populated flag.  
  
Definition of Done:  
- The system correctly validates the input answer and performs the appropriate actions based on the conditions specified.  
- Error messages are displayed when the input answer is invalid.  
- Related group questions are populated or deleted as required.  
- Requirement descriptions are updated based on the input answer.  
- All functionalities are tested and verified to work as expected.

# Manage Reinsurance Details for Insurance Covers

Type: REINSURANCE

Detailed description: As a user, I want to manage reinsurance details for various insurance covers, including setting up cover numbers, cover codes, descriptions, reinsurance types, reinsurance codes, reinsurance percentages, and reference numbers. This will help in maintaining accurate and detailed records of reinsurance agreements and their associated parameters.  
  
Acceptance criteria:  
1. The system should allow the user to input and display cover numbers, cover codes, and cover descriptions.  
2. The system should support the selection of reinsurance types and codes.  
3. The system should enable the user to specify reinsurance percentages and reference numbers.  
4. The system should validate that the reinsurance type and code are consistent with the selected cover code and product ID.  
5. The system should calculate and display the reinsurance sum assured based on the input parameters.  
6. The system should handle different reinsurance types such as 'AUTO' and 'FACULTATIVE' and apply the appropriate logic for each type.  
7. The system should ensure that the reinsurance details are correctly linked to the corresponding insurance covers and product IDs.  
  
Definition of Done:  
- The user can successfully input and manage reinsurance details for various insurance covers.  
- The system validates and ensures consistency of reinsurance types and codes with cover codes and product IDs.  
- The reinsurance sum assured is accurately calculated and displayed based on the input parameters.  
- The system handles different reinsurance types and applies the appropriate logic for each type.  
- All reinsurance details are correctly linked to the corresponding insurance covers and product IDs.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- The system should execute the following query to fetch cover codes and descriptions:  
 ```sql  
 SELECT B.COVER\_CODE, C.cover\_description  
 FROM AZBJ\_PACKAGE\_MASTER A,  
 AZBJ\_PACKAGE\_COVERS B,  
 CFG\_V\_PROD\_COVERS\_API C  
 WHERE A.package\_code = B.PACKAGE\_CODE  
 AND A.PRODUCT\_ID = :CONTROL.CN\_PRODUCT\_ID  
 AND A.PACKAGE\_CODE = :COVERHEAD.CH\_PACKAGE  
 AND A.PRODUCT\_ID = C.PRODUCT\_ID  
 AND B.COVER\_CODE = C.COVER\_CODE  
 ORDER BY B.COVER\_CODE;  
 ```

# Delete Record from Reinsurance Data Set

Type: REINSURANCE

Title: Delete Record from Reinsurance Data Set  
  
Acceptance Criteria:  
1. When the delete button is pressed, the system should navigate to the reinsurance data set.  
2. The system should then delete the currently selected record from the reinsurance data set.  
  
Definition of Done:  
- The delete button is functional and visible on the user interface.  
- Pressing the delete button successfully navigates to the reinsurance data set.  
- The selected record in the reinsurance data set is deleted upon pressing the delete button.  
- The user receives a confirmation that the record has been deleted.  
  
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 table references.

# Calculate Reinsurance Amount

Type: REINSURANCE

Title: Calculate Reinsurance Amount  
  
Acceptance Criteria:  
- When the reinsurance percentage is provided, the system should automatically calculate the reinsurance amount.  
- The reinsurance amount should be calculated as: (Reinsurance Percentage / 100) Cover Amount.  
- The calculated reinsurance amount should be stored in the appropriate field.  
  
Definition of Done:  
- The reinsurance amount is correctly calculated and displayed based on the provided reinsurance percentage and cover amount.  
- The calculation logic is implemented and tested to ensure accuracy.  
- The user interface reflects the calculated reinsurance amount without requiring manual input.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include any specific database queries.

# Double-click on Reins Code to retrieve and display reinsurers

Type: REINSURANCE

Detailed description: As a user, I want to be able to double-click on the "Reins Code" field to trigger a process that retrieves and displays a list of reinsurers based on the selected reinsurance type and product ID, so that I can easily select a reinsurer from the list.  
  
Acceptance criteria:  
1. When the user double-clicks on the "Reins Code" field, the system should check if the reinsurance type is not 'AUTO'.  
2. If the reinsurance type is not 'AUTO', the system should execute a query to retrieve reinsurer codes from the `azbj\_reinsurer` table where the reinsurance type matches the selected reinsurance type and the product ID matches the selected product ID.  
3. The retrieved reinsurer codes should be displayed in a list of values (LOV) for the user to select from.  
4. If any error occurs during the process, an appropriate error message should be displayed to the user.  
  
Definition of Done:  
- The double-click action on the "Reins Code" field triggers the retrieval and display of reinsurer codes based on the specified criteria.  
- The list of reinsurers is displayed correctly in a LOV.  
- Error handling is implemented to display appropriate error messages in case of any issues during the process.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Query to retrieve reinsurer codes:  
 ```sql  
 SELECT reinsurer\_code  
 FROM azbj\_reinsurer  
 WHERE REINSURANCE\_TYPE = NVL(:Control.rn\_type, 'AUTO')  
 AND product\_id = :control.cn\_product\_id;  
 ```  
  
- Query to retrieve cover codes and descriptions:  
 ```sql  
 SELECT B.COVER\_CODE, C.cover\_description  
 FROM AZBJ\_PACKAGE\_MASTER A,  
 AZBJ\_PACKAGE\_COVERS B,  
 CFG\_V\_PROD\_COVERS\_API C  
 WHERE a.package\_code = B.PACKAGE\_CODE  
 AND A.PRODUCT\_ID = :CONTROL.CN\_PRODUCT\_ID  
 AND A.PACKAGE\_CODE = :COVERHEAD.CH\_PACKAGE  
 AND A.PRODUCT\_ID = C.PRODUCT\_ID  
 AND B.COVER\_CODE = C.COVER\_CODE  
 ORDER BY b.cover\_code;  
 ```

# Reinsurance Type Selection

Type: REINSURANCE

Title: Reinsurance Type Selection  
  
Acceptance Criteria:  
1. When the user double-clicks on the reinsurance type field, the system should check the current cover codes associated with the product.  
2. If the reinsurance type is not 'AUTO', the system should query the database to retrieve the reinsurer codes based on the selected reinsurance type, product ID, and cover code.  
3. The system should display a list of reinsurer codes for the user to select from.  
4. The system should handle any errors that occur during the query and display an appropriate error message.  
  
Definition of Done:  
- The user can double-click on the reinsurance type field to trigger the selection process.  
- The system correctly retrieves and displays the reinsurer codes based on the specified criteria.  
- The user can select a reinsurer code from the list.  
- Any errors encountered during the process are handled gracefully and an error message is displayed to the user.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Query to retrieve reinsurer codes:  
 ```sql  
 SELECT reinsurer\_code  
 FROM azbj\_reinsurer  
 WHERE REINSURANCE\_TYPE = NVL(:Control.rn\_type, 'AUTO')  
 AND product\_id = :control.cn\_product\_id;  
 ```  
  
- Query to retrieve cover codes and descriptions:  
 ```sql  
 SELECT B.COVER\_CODE, C.cover\_description  
 FROM AZBJ\_PACKAGE\_MASTER A,  
 AZBJ\_PACKAGE\_COVERS B,  
 CFG\_V\_PROD\_COVERS\_API C  
 WHERE a.package\_code = B.PACKAGE\_CODE  
 AND A.PRODUCT\_ID = :CONTROL.CN\_PRODUCT\_ID  
 AND A.PACKAGE\_CODE = :COVERHEAD.CH\_PACKAGE  
 AND A.PRODUCT\_ID = C.PRODUCT\_ID  
 AND B.COVER\_CODE = C.COVER\_CODE  
 ORDER BY b.cover\_code;  
 ```

# Back Button Functionality on Reinsurance Screen

Type: REINSURANCE

Title: Back Button Functionality on Reinsurance Screen  
  
Acceptance Criteria:  
1. When the "Refer to Reinsurer" option is enabled and selected:  
 - If the Reinsurer Code is not provided, display an error message: "Please select Reinsurer".  
 - If the Reinsurance Reason is not provided, display an error message: "Please select Reinsurance Reason".  
 - If the Underwriters' recommendation is not provided, display an error message: "Please select Underwriters recommendation".  
 - If any of the Underwriters' comments (Medical, Financial, Final) are not provided, display an error message: "Please enter Underwriters comments".  
 - If the Underwriters' recommendation is "COUNTER OFFER", ensure that at least one counter offer detail is provided. If not, display an error message: "Please enter the counter offer details".  
  
2. If the "Refer to Reinsurer" option is enabled but not selected, navigate to the Medical Underwriting screen and hide the Reinsurer Selection screen.  
  
3. If the "Refer to Reinsurer" option is enabled and selected, prompt the user with a confirmation message: "Do you want to refer this case to Re-insurer?". If the user confirms, set the form status to 'N', display a warning message to save and exit the proposal, navigate to the Medical Underwriting screen, and hide the Reinsurer Selection screen. If the user cancels, reset the "Refer to Reinsurer" option and raise a form trigger failure.  
  
Definition of Done:  
- The "Back" button functionality is implemented as per the acceptance criteria.  
- All necessary validations and error messages are displayed correctly.  
- The navigation flow works as expected based on user inputs and conditions.  
- The feature is tested and verified to ensure it meets the requirements.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include any direct database queries that can be executed independently of Oracle Forms constructs.

# Double-click on Cover Code to Display Relevant Covers

Type: REINSURANCE

Detailed description: As a user, I want to be able to double-click on the "Cover Code" field within the Reinsurance section to trigger a process that collects and displays relevant cover codes and descriptions from the database, so that I can easily select and view the details of the available covers.  
  
Acceptance criteria:  
1. When the user double-clicks on the "Cover Code" field, the system should:  
 - Navigate to the "COVERS" section and iterate through all records to collect non-null cover codes.  
 - Construct a query string to fetch cover codes and descriptions from the database based on the collected cover codes and other parameters.  
 - Populate a record group with the results of the query.  
 - Display a list of values (LOV) for the user to select from.  
 - Navigate back to the "REINSURANCE" section and set the focus on the "Cover Code" field.  
 - If the query string is not null, execute the query and handle any errors that occur during the process.  
 - Display the LOV to the user.  
 - Navigate to the "REINSURANCE" section and set the focus on the "Reinsurance Type" field.  
  
Definition of Done:  
- The process is triggered by a double-click on the "Cover Code" field.  
- The system successfully collects and displays the relevant cover codes and descriptions.  
- The LOV is displayed to the user, allowing them to select a cover code.  
- The system handles any errors gracefully and provides appropriate feedback to the user.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- The query to fetch cover codes and descriptions:  
 ```sql  
 SELECT B.COVER\_CODE, C.cover\_description  
 FROM AZBJ\_PACKAGE\_MASTER A, AZBJ\_PACKAGE\_COVERS B, CFG\_V\_PROD\_COVERS\_API C  
 WHERE A.PACKAGE\_CODE = B.PACKAGE\_CODE  
 AND A.PRODUCT\_ID = :CONTROL.CN\_PRODUCT\_ID  
 AND A.PACKAGE\_CODE = :COVERHEAD.CH\_PACKAGE\_CODE  
 AND A.PRODUCT\_ID = C.PRODUCT\_ID  
 AND B.COVER\_CODE = C.COVER\_CODE  
 ORDER BY B.COVER\_CODE;  
 ```

# Image Interaction and Manipulation

Type: BBU\_IMAGE

Title: Image Interaction and Manipulation  
  
Acceptance Criteria:  
1. The user should be able to view three images on the screen.  
2. Each image should have controls to navigate (left, right, up, down) and zoom in/out.  
3. The user should be able to reset the zoom level of each image.  
4. The user should be able to print each image.  
5. The user should be able to navigate between different sets of images using next and previous buttons.  
6. The user should be able to view and hide additional information related to the images.  
  
Definition of Done:  
1. The functionality to view, navigate, and manipulate images is implemented and tested.  
2. The navigation buttons (left, right, up, down) work as expected for each image.  
3. The zoom in/out buttons adjust the image size correctly.  
4. The reset zoom button restores the image to its original size.  
5. The print button initiates the print process for the selected image.  
6. The next and previous buttons allow the user to switch between different sets of images.  
7. The view and hide buttons toggle the visibility of additional information.  
8. All functionalities are tested and verified to be working as expected.  
9. The user interface is intuitive and user-friendly.  
10. The implementation is independent of any specific technology or platform.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Not applicable as there are no specific CRUD operations mentioned in the provided XML content.

# Zoom In/Out Functionality for Image in BBU\_IMAGE Section

Type: BBU\_IMAGE

Title: Zoom In/Out Functionality for Image in BBU\_IMAGE Section  
  
Acceptance Criteria:  
1. When the button labeled " - " is pressed, the system should call a function to zoom the image displayed in the BBU\_IMAGE section.  
2. The zoom function should take the current zoom factor and adjust it by multiplying it with -1, effectively toggling between zoom in and zoom out.  
3. The image should be displayed on the canvas named CAN\_BBU\_IMAGE within the window named WIN\_BBU\_IMAGE.  
  
Definition of Done:  
- The button should be functional and correctly trigger the zoom function.  
- The image should zoom in and out as expected when the button is pressed.  
- The functionality should be tested and verified to ensure it meets the acceptance criteria.

# Image Scrolling Functionality

Type: BBU\_IMAGE

Title: Image Scrolling Functionality  
  
Acceptance Criteria:  
1. When the user presses the button labeled ">>", the system should call a function to scroll the image.  
2. The function should take the following parameters:  
 - The image identifier.  
 - The current X and Y coordinates of the image.  
 - A predefined scroll factor.  
 - A zero value (indicating no additional offset).  
 - The current Z factor of the image.  
  
Definition of Done:  
1. The button labeled ">>" is present and functional in the BBU Image section.  
2. Pressing the button successfully scrolls the image based on the predefined parameters.  
3. The image scroll functionality is tested and verified to work as expected.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include any database CRUD operations.

# Image Scrolling Functionality

Type: BBU\_IMAGE

Title: Image Scrolling Functionality  
  
Acceptance Criteria:  
1. When the button labeled "V" is pressed, the system should call a function to scroll the image.  
2. The function should take the following parameters:  
 - The image identifier.  
 - The current X and Y coordinates of the image.  
 - A scroll factor to determine the amount of scroll.  
 - A Z factor to adjust the zoom level of the image.  
  
Definition of Done:  
1. The button should be visible and labeled "V".  
2. Pressing the button should trigger the image scrolling function.  
3. The image should scroll according to the specified parameters.  
4. The functionality should be 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 direct database CRUD operations.

# Scroll Image to the Left on Button Press

Type: BBU\_IMAGE

Title: Scroll Image to the Left on Button Press  
  
Acceptance Criteria:  
1. When the LEFT\_1 button is pressed, the image in the BBU\_IMAGE section should scroll to the left.  
2. The scroll factor should be determined by a predefined value multiplied by -1.  
3. The X and Y coordinates of the image should be updated accordingly.  
4. The Z factor of the image should remain unchanged.  
  
Definition of Done:  
- The LEFT\_1 button is functional and scrolls the image to the left as per the acceptance criteria.  
- The image's X and Y coordinates are updated correctly.  
- The feature is tested and verified to work as expected.  
- The implementation is independent of any specific technology or Oracle Forms terminology.

# Image Zoom Functionality

Type: BBU\_IMAGE

Title: Image Zoom Functionality  
  
Acceptance Criteria:  
1. When the user presses the zoom button, the application should call a function to zoom into the image.  
2. The zoom function should take the current image and zoom factor as parameters.  
3. The zoom factor should be adjustable based on predefined settings.  
  
Definition of Done:  
1. The zoom button is visible and clickable.  
2. Pressing the zoom button successfully calls the zoom function.  
3. The image is displayed with the new zoom factor.  
4. The zoom factor can be adjusted and is applied correctly to the image.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include any database CRUD operations.

# Image Scrolling Functionality

Type: BBU\_IMAGE

Title: Image Scrolling Functionality  
  
Acceptance Criteria:  
- When the button labeled "V" is pressed, the system should call a function to scroll the image.  
- The function should take the following parameters:  
 - The identifier of the image to be scrolled.  
 - The current X and Y coordinates of the image.  
 - A scroll direction parameter set to 0.  
 - A predefined scroll factor.  
 - The current Z factor of the image.  
  
Definition of Done:  
- The button labeled "V" is present and functional.  
- Pressing the button triggers the image scrolling function with the correct parameters.  
- The image scrolls as expected based on the predefined scroll factor and direction.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include any database CRUD operations.

# Zoom Functionality for Image in BBU\_IMAGE Section

Type: BBU\_IMAGE

Detailed description: As a user, I want to be able to zoom into an image displayed in the BBU\_IMAGE section by pressing a button, so that I can view the image in greater detail.  
  
Acceptance criteria:  
1. When the user presses the button labeled "+", the system should trigger a zoom function.  
2. The zoom function should take the current image displayed in the BBU\_IMAGE section and apply a zoom factor to it.  
3. The zoom factor should be determined by the current zoom factor value and a predefined global zoom factor.  
  
Definition of Done:  
1. The button labeled "+" is present in the BBU\_IMAGE section.  
2. Pressing the button successfully triggers the zoom function.  
3. The image is zoomed in according to the specified zoom factors.  
4. The zoomed image is displayed correctly within the BBU\_IMAGE section.  
  
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.

# Navigate to Previous Image in BBU\_IMAGE Section

Type: BBU\_IMAGE

Title: Navigate to Previous Image in BBU\_IMAGE Section  
  
Acceptance Criteria:  
1. When the button is pressed, the system should call a function to move to the previous image in the sequence.  
2. The function should update the current image file, cursor position, and other related parameters.  
3. The image coordinates (X, Y) should be reset to 0.  
4. The zoom factor should be set to a default value.  
  
Definition of Done:  
1. The button should be functional and allow navigation to the previous image.  
2. The image and related parameters should update correctly when the button is pressed.  
3. The coordinates and zoom factor should reset as specified.  
  
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.

# Image Scrolling Functionality

Type: BBU\_IMAGE

Title: Image Scrolling Functionality  
  
Acceptance Criteria:  
1. When the user presses the button labeled "^", the image in the BBU\_IMAGE section should scroll.  
2. The scrolling should be controlled by the current X and Y positions of the image and a predefined scroll factor.  
3. The image should move upwards by a distance determined by the scroll factor multiplied by -1.  
  
Definition of Done:  
1. The button labeled "^" is present and functional.  
2. Pressing the button causes the image to scroll upwards.  
3. The scrolling behavior is smooth and consistent with the predefined scroll factor.  
4. The feature is tested and verified to work as expected.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include any database queries.

# Scroll Image to the Left in BBU\_IMAGE Section

Type: BBU\_IMAGE

Title: Scroll Image to the Left in BBU\_IMAGE Section  
  
Acceptance Criteria:  
1. When the "<<" button is pressed, the image in the BBU\_IMAGE section should scroll to the left.  
2. The scroll factor should be determined by a predefined value multiplied by -1.  
3. The X and Y coordinates of the image should be updated accordingly.  
4. The Z factor of the image should remain unchanged.  
  
Definition of Done:  
- The "<<" button successfully scrolls the image to the left.  
- The image's X and Y coordinates are updated based on the scroll factor.  
- The Z factor of the image remains unchanged.  
- 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 XML content does not include any direct database CRUD operations.

# Image Scrolling Functionality

Type: BBU\_IMAGE

Title: Image Scrolling Functionality  
  
Acceptance Criteria:  
1. When the user presses the designated button, the application should scroll the image in the display section.  
2. The scrolling should be based on predefined factors and coordinates to ensure smooth navigation.  
3. The image should move vertically by a factor determined by the application logic.  
  
Definition of Done:  
1. The button is visible and clickable within the image display section.  
2. Pressing the button triggers the image to scroll vertically.  
3. The scrolling behavior is consistent with the predefined factors and coordinates.  
4. The feature is tested and verified to work as expected in the user interface.  
  
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.

# Implement Image Scrolling Functionality via Button Press

Type: BBU\_IMAGE

Title: Implement Image Scrolling Functionality via Button Press  
  
Acceptance Criteria:  
1. When the user presses the button labeled ">>", the image should scroll horizontally.  
2. The scrolling should be based on a predefined scroll factor.  
3. The image's X and Y coordinates should be updated accordingly to reflect the new position after scrolling.  
4. The Z factor of the image should remain unchanged during the scroll operation.  
  
Definition of Done:  
1. The button labeled ">>" is functional and triggers the scroll action.  
2. The image scrolls horizontally when the button is pressed.  
3. The image's coordinates are updated correctly based on the scroll factor.  
4. The feature is tested and verified to work as expected without any errors.  
  
SQL Query for Reference:  
- Not applicable as the provided XML content does not include any database CRUD operations.

# Image Scrolling Functionality

Type: BBU\_IMAGE

Title: Image Scrolling Functionality  
  
Acceptance Criteria:  
1. When the user presses the button labeled "<<", the system should call a function to scroll the image.  
2. The function should adjust the X and Y coordinates of the image based on a predefined scroll factor.  
3. The scroll factor should be multiplied by -1 to ensure the image scrolls in the correct direction.  
4. The Z factor of the image should remain unchanged during the scroll operation.  
  
Definition of Done:  
- The button labeled "<<" is present and functional.  
- Pressing the button triggers the image scroll function.  
- The image scrolls correctly based on the predefined scroll factor.  
- The X and Y coordinates of the image are updated appropriately.  
- The Z factor of the image remains unchanged.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include any database CRUD operations.

# Image Scrolling Functionality

Type: BBU\_IMAGE

Title: Image Scrolling Functionality  
  
Acceptance Criteria:  
1. When the user presses the button labeled ">>", the system should call a function to scroll the image displayed in the BBU\_IMAGE section.  
2. The function should take the current image coordinates (X, Y) and a predefined scroll factor to determine the new position of the image.  
3. The image should be scrolled horizontally by the predefined scroll factor.  
4. The function should also consider the Z factor for any additional adjustments needed for the image display.  
  
Definition of Done:  
- The button labeled ">>" is present in the BBU\_IMAGE section.  
- Pressing the button triggers the image scroll function.  
- The image scrolls horizontally by the predefined scroll factor.  
- The new image position is correctly calculated and displayed.  
- The feature is tested and verified to work as expected.

# Image Scrolling Functionality

Type: BBU\_IMAGE

Title: Image Scrolling Functionality  
  
Acceptance Criteria:  
1. When the button labeled "V" is pressed, the system should call a function to scroll the image.  
2. The function should take the following parameters:  
 - The identifier of the image to be scrolled.  
 - The current X and Y coordinates of the image.  
 - A scroll factor to determine the amount of scroll.  
 - A Z factor to adjust the zoom level of the image.  
  
Definition of Done:  
1. The button labeled "V" is visible and clickable within the BBU\_IMAGE section.  
2. Pressing the button triggers the image scroll function with the correct parameters.  
3. The image scrolls as expected based on the provided scroll and zoom factors.  
4. The functionality is tested and verified to work correctly without any errors.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include any database queries.

# Image Scrolling Functionality

Type: BBU\_IMAGE

Title: Image Scrolling Functionality  
  
Acceptance Criteria:  
1. When the user presses the designated button, the application should trigger a scroll action.  
2. The scroll action should adjust the image's X and Y coordinates based on a predefined scroll factor.  
3. The scroll factor should be multiplied by -1 to ensure the correct direction of scrolling.  
4. The image's Z factor should remain unchanged during the scroll action.  
  
Definition of Done:  
1. The button is visible and clickable within the image display section.  
2. Pressing the button successfully triggers the scroll action.  
3. The image's position updates correctly according to the scroll factor.  
4. 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 XML content does not include any direct database CRUD operations.

# Image Navigation Button Functionality

Type: BBU\_IMAGE

Title: Image Navigation Button Functionality  
  
Acceptance Criteria:  
1. When the button is pressed, the system should call a function to move to the next image in the sequence.  
2. The function should update the current image file, cursor position, and maximum image count.  
3. The image display settings (X, Y coordinates, and zoom factor) should be reset to their default values.  
  
Definition of Done:  
- The button successfully triggers the image navigation function.  
- The current image and related settings are updated correctly.  
- The image display settings are reset to default values after navigation.  
  
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.

# Image Navigation Button Functionality

Type: BBU\_IMAGE

Title: Image Navigation Button Functionality  
  
Acceptance Criteria:  
1. When the button is pressed, the system should call a function to move the image page.  
2. The function should update the current image file, cursor position, and other related parameters.  
3. The image coordinates (X, Y) should be reset to 0.  
4. The zoom factor should be set to a default value.  
  
Definition of Done:  
- The button should be functional and trigger the image navigation logic.  
- The image page should update correctly based on the function's parameters.  
- The image coordinates and zoom factor should reset as specified.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include any direct database queries.

# Zoom Functionality for Image in BBU\_IMAGE Section

Type: BBU\_IMAGE

Detailed description: As a user, I want to be able to zoom into an image displayed in the BBU\_IMAGE section by pressing a button, so that I can view the image in greater detail.  
  
Acceptance criteria:  
1. When the user presses the button labeled "-", the system should trigger a zoom function.  
2. The zoom function should take the current image displayed in the BBU\_IMAGE section and apply a zoom factor to it.  
3. The zoom factor should be dynamically calculated based on a predefined global zoom factor multiplied by -1.  
  
Definition of Done:  
1. The button labeled "-" is visible and clickable in the BBU\_IMAGE section.  
2. Pressing the button successfully triggers the zoom function.  
3. The image in the BBU\_IMAGE section is zoomed in or out based on the calculated zoom factor.  
4. 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 XML content does not include any database CRUD operations.

# Zoom Functionality for Image in BBU\_IMAGE Section

Type: BBU\_IMAGE

Title: Zoom Functionality for Image in BBU\_IMAGE Section  
  
Acceptance Criteria:  
1. When the user presses the button labeled "+", the system should trigger a zoom function.  
2. The zoom function should take the current image and apply a zoom factor to it.  
3. The zoom factor should be determined by the current zoom factor of the image and a predefined global zoom factor.  
  
Definition of Done:  
1. The button labeled "+" is visible and clickable.  
2. Pressing the button successfully triggers the zoom function.  
3. The image is zoomed in according to the specified zoom factors.  
4. The zoomed image is displayed correctly within the BBU\_IMAGE section.  
  
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.

# Image Navigation Button Functionality

Type: BBU\_IMAGE

Title: Image Navigation Button Functionality  
  
Acceptance Criteria:  
1. When the button labeled ">" is pressed, the system should call a function to move to the next image in the sequence.  
2. The function should update the current image file, the current image position, and the maximum number of images.  
3. The image coordinates (X, Y) should be reset to 0.  
4. The zoom factor should be set to a default value.  
  
Definition of Done:  
- The button should be functional and allow navigation to the next image.  
- The image details should be updated correctly as per the function logic.  
- The image coordinates and zoom factor should be reset as specified.  
  
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.

# Zoom Functionality for Image in BBU\_IMAGE Section

Type: BBU\_IMAGE

Title: Zoom Functionality for Image in BBU\_IMAGE Section  
  
User Story:  
As a user, I want to be able to zoom in and out of an image displayed in the BBU\_IMAGE section by pressing a button, so that I can view the image in different levels of detail.  
  
Acceptance Criteria:  
1. When the user presses the button labeled "-", the system should call a function to zoom the image.  
2. The zoom function should take the current zoom factor and multiply it by -1 to toggle between zooming in and out.  
3. The image to be zoomed is identified by the field 'BBU\_IMAGE.IMG\_3'.  
4. The zoom factor is determined by the value in 'BBU\_IMAGE.IMG\_3\_Z\_FACTOR' and a global zoom factor.  
  
Definition of Done:  
- The button press triggers the zoom function correctly.  
- The image zooms in and out based on the current zoom factor.  
- The user can see the image at different levels of detail after pressing the button.  
- The functionality is tested and verified to work as expected.  
  
SQL Query for Reference:  
N/A

# Navigate Through Images in BBU\_IMAGE Section

Type: BBU\_IMAGE

Title: Navigate Through Images in BBU\_IMAGE Section  
  
Acceptance Criteria:  
1. When the button labeled ">" is pressed, the system should call a function to move to the next image in the sequence.  
2. The function should update the current image file, cursor position, and other related parameters.  
3. The image coordinates (X, Y) should be reset to 0.  
4. The zoom factor should be set to a default value.  
  
Definition of Done:  
- The button should be functional and allow navigation to the next image.  
- The image and related parameters should update correctly upon button press.  
- The coordinates and zoom factor should reset as specified.  
  
SQL Query for Reference:  
- Not applicable as the provided XML content does not include any direct database CRUD operations.  
  
Block Name: BBU\_IMAGE

# View BBU Documents

Type: BBU\_IMAGE

Title: View BBU Documents  
  
Acceptance Criteria:  
1. When the "View Questions" button is pressed, the system should:  
 - Populate the BBU questions.  
 - Navigate to the BBU Image Details section.  
 - Display the BBU Image window.  
  
Definition of Done:  
- The "View Questions" button is functional and triggers the appropriate actions.  
- The BBU Image Details section is navigated to upon button press.  
- The BBU Image window is displayed correctly with the relevant details.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include specific database queries.

# Navigate to Previous Image in Gallery

Type: BBU\_IMAGE

Title: Navigate to Previous Image in Gallery  
  
User Story:  
As a user, I want to navigate to the previous image in the image gallery so that I can view the earlier images in the sequence.  
  
Acceptance Criteria:  
1. When the "Previous" button is pressed, the system should call a function to move to the previous image in the gallery.  
2. The function should update the current image file, the current image index, and other related image properties.  
3. The image coordinates (X, Y) should be reset to (0, 0).  
4. The zoom factor should be set to the default zoom level.  
5. The system should handle the image file type appropriately.  
6. The system should ensure that the image properties such as X, Y coordinates, and zoom factor are updated correctly.  
  
Definition of Done:  
- The "Previous" button is functional and correctly navigates to the previous image.  
- The image properties and coordinates are updated as specified.  
- The zoom factor is reset to the default value.  
- The functionality is tested and verified to work as expected.  
- The image file type is handled correctly.  
- All acceptance criteria are met without any Oracle Forms terminology.

# Print Image Functionality

Type: BBU\_IMAGE

Title: Print Image Functionality  
  
Acceptance Criteria:  
1. When the "Print" button is pressed, the system should call a function to print the image associated with the BBU\_IMAGE section.  
2. The function should use the current image name stored in a global variable to identify which image to print.  
  
Definition of Done:  
1. The "Print" button is visible and correctly labeled.  
2. Clicking the "Print" button triggers the image printing function.  
3. The image is successfully printed using the name stored in the global variable.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include any CRUD operations.

# Reset Zoom Button for Image

Type: BBU\_IMAGE

Title: Reset Zoom Button for Image  
  
Acceptance Criteria:  
- When the "Reset" button is pressed, the image should automatically adjust to fit within the specified area.  
- The image should be resized proportionally to ensure that it fits within the designated space without distortion.  
  
Definition of Done:  
- The "Reset" button is visible and functional.  
- Pressing the "Reset" button adjusts the image to fit within the designated area.  
- The image is resized proportionally and is not distorted.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include any database CRUD operations.

# Print Image Functionality

Type: BBU\_IMAGE

Title: Print Image Functionality  
  
Acceptance Criteria:  
- When the "Print" button is clicked, the system should call a function to print the image associated with the BBU\_IMAGE section.  
- The function should use the current image name stored in a global variable to identify which image to print.  
  
Definition of Done:  
- The "Print" button is visible and correctly labeled.  
- Clicking the "Print" button triggers the image printing function.  
- The image is printed using the name stored in the global variable.  
  
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.

# Reset Image Zoom

Type: BBU\_IMAGE

Title: Reset Image Zoom  
  
Acceptance Criteria:  
- When the "Reset" button is pressed, the image displayed in the "BBU\_IMAGE" section should be adjusted to fit within its designated area.  
- The image should revert to its original size and position, ensuring that no part of the image is zoomed in or out.  
  
Definition of Done:  
- The "Reset" button is functional and correctly resets the zoom level of the image.  
- The image is displayed in its entirety within the designated area after the reset.  
- The functionality has been tested and verified to work as expected.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include any database CRUD operations.

# Print Image Functionality

Type: BBU\_IMAGE

Title: Print Image Functionality  
  
Acceptance Criteria:  
- When the "Print" button is pressed, the system should call a function to print the image associated with the item IMG\_3.  
- The function should use the name of the previous image stored in a global variable to identify which image to print.  
  
Definition of Done:  
- The "Print" button is visible and correctly labeled.  
- Pressing the "Print" button triggers the image printing function.  
- The image is successfully printed using the name stored in the global variable.  
  
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.

# Reset Image Zoom

Type: BBU\_IMAGE

Title: Reset Image Zoom  
  
Acceptance Criteria:  
- When the "Reset" button is pressed, the image displayed in the "BBU\_IMAGE" section should automatically adjust to fit within its designated area.  
- The image should revert to its original size and position, ensuring that it is fully visible within the specified boundaries.  
  
Definition of Done:  
- The "Reset" button is functional and correctly resets the zoom level of the image.  
- The image adjusts to fit within the designated area upon pressing the "Reset" button.  
- The feature is tested and verified to work as expected in various scenarios.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable for this user story as it involves UI interaction and does not directly interact with the database.

# Hide BBU Image Window and Navigate Based on Item Status

Type: BBU\_IMAGE

Title: Hide BBU Image Window and Navigate Based on Item Status  
  
Acceptance Criteria:  
1. When the "Hide Questions" button is pressed, the BBU Image window should be hidden.  
2. The system should navigate to the "AGENTS" block.  
3. If the specified item in the BBU Image block is enabled, the system should navigate to that item.  
4. If the specified item in the BBU Image block is not enabled, the system should navigate to a default item in the "COVERS" block.  
5. In case of any exceptions, the system should navigate to the specified item in the BBU Image block.  
  
Definition of Done:  
- The "Hide Questions" button hides the BBU Image window.  
- The system navigates to the "AGENTS" block.  
- The system correctly navigates to the specified item if it is enabled.  
- The system navigates to the default item in the "COVERS" block if the specified item is not enabled.  
- The system handles exceptions by navigating to the specified item in the BBU Image block.

# User Interface for Managing Item Values

Type: BBU\_IMAGE\_ITEMS

Title: User Interface for Managing Item Values  
  
Acceptance Criteria:  
1. The interface should display the existing item value and allow the user to input a new item value.  
2. There should be a checkbox to indicate whether the item value has been changed.  
3. A button should be available to save the new item value.  
4. The interface should include a label for the item.  
5. The system should ensure that changes are only allowed if the "CHANGE\_ALLOWED" condition is met.  
6. The "ITEM\_NAME" should be stored but not visible to the user.  
  
Definition of Done:  
1. The user interface displays the existing item value and allows the user to input a new item value.  
2. The checkbox for indicating changes is functional.  
3. The save button works and updates the item value.  
4. The item label is displayed correctly.  
5. The system checks the "CHANGE\_ALLOWED" condition before allowing changes.  
6. The "ITEM\_NAME" is stored in the system but remains hidden from the user.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- No specific DB queries provided in the XML content.

# Conditional Value Change for Item

Type: BBU\_IMAGE\_ITEMS

Title: Conditional Value Change for Item  
  
Acceptance Criteria:  
1. When the checkbox for the item is checked (value set to 'Y'), the system should verify if changes are allowed.  
2. If changes are not allowed (change\_allowed is 'N'), the system should display a warning message: "Cannot Change The Value For This Item".  
3. The checkbox should then be reset to unchecked (value set to 'N').  
4. The new item value should be set to the original item value.  
  
Definition of Done:  
- The system correctly checks the conditions when the checkbox is changed.  
- The warning message is displayed if changes are not allowed.  
- The checkbox is reset to its original state if changes are not allowed.  
- The new item value is updated to the original item value if changes are not allowed.  
- All acceptance criteria are met and tested successfully.

# Update Item Values in BBU\_IMAGE\_ITEMS Section

Type: BBU\_IMAGE\_ITEMS

Detailed description: As a user, I want to be able to update item values in the BBU\_IMAGE\_ITEMS section so that any changes marked as 'Y' are copied to the corresponding item names.  
  
Acceptance criteria:  
1. When the "Change" button is pressed, the system should navigate to the BBU\_IMAGE\_ITEMS section.  
2. The system should determine the total number of records in the BBU\_IMAGE\_ITEMS section.  
3. For each record in the BBU\_IMAGE\_ITEMS section:  
 - If the value\_change attribute is 'Y', the new\_item\_value should be copied to the item\_name.  
4. The system should return to the first record after processing all records.  
  
Definition of Done:  
- The "Change" button functionality is implemented and tested.  
- The system correctly navigates through all records in the BBU\_IMAGE\_ITEMS section.  
- The new\_item\_value is copied to the item\_name for records where value\_change is 'Y'.  
- The system returns to the first record after processing.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided logic does not include direct database CRUD operations.

# Manage Image Details for Proposals and Applications

Type: IMAGE\_DET

Title: Manage Image Details for Proposals and Applications  
  
Acceptance Criteria:  
1. The user should be able to view all images associated with a proposal or application.  
2. The user should be able to view individual images.  
3. The user should be able to see details such as Proposal Number, Application Number, Image Type, Scan Time, and Image Size.  
4. The user should be able to access additional functionalities like viewing web sales proposal forms and OMNI images.  
5. The user should be able to mark an image as having a signature mismatch.  
  
Definition of Done:  
1. The user interface should display all relevant image details.  
2. Buttons for viewing images, viewing all images, accessing web sales proposal forms, and OMNI images should be functional.  
3. The checkbox for marking signature mismatches should be operational.  
4. All functionalities 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 database queries or operations.

# View All Images Sequentially

Type: IMAGE\_DET

Title: View All Images Sequentially  
  
Acceptance Criteria:  
1. When the "View All Image" button is pressed, the system should navigate to the image details section.  
2. The system should iterate through all image records, loading each image one by one.  
3. The system should start from the first image record and continue until the last image record is reached.  
  
Definition of Done:  
- The "View All Image" button is functional and triggers the process of loading and displaying all images sequentially.  
- The system correctly navigates through all image records and displays each image without errors.  
- The feature has been tested and verified to work as expected.

# View OMNI Images for Insured Person

Type: IMAGE\_DET

Detailed description: As a user, I want to view the OMNI Images related to an insured person by clicking a button, so that I can access the relevant documents quickly and efficiently.  
  
Acceptance criteria:  
1. When the user clicks the "OMNI Images" button, the system should generate a URL using the `azbj\_encrypt\_dms\_link` function.  
2. The URL should be constructed using the insured person's verification number or sign card number.  
3. If the generated URL is not null, the system should open the URL in a web browser to display the documents.  
  
Definition of Done:  
- The "OMNI Images" button is present and clickable.  
- Clicking the button triggers the generation of a URL based on the insured person's details.  
- The URL is opened in a web browser if it is valid and not null.  
- The functionality is tested and verified to ensure that the correct documents are displayed.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Not applicable as the provided logic does not include direct CRUD operations on the database.

# Access Web Sales Proposal Form

Type: IMAGE\_DET

Detailed description: As a user, I want to access the Web Sales Proposal Form by clicking a button, so that I can view the proposal details online.  
  
Acceptance criteria:  
1. When the button is pressed, the system should retrieve the transaction ID (`trans\_id`) from the `azbj\_cq\_trans\_data` table based on the `application\_no` which is derived from either `ip\_verf\_no` or `ip\_sign\_card\_no` of the insured person.  
2. If no data is found for the transaction ID, it should be set to 0. If any other error occurs, the transaction ID should be set to NULL.  
3. The system should then retrieve the system description (`sys\_desc`) from the `azbj\_system\_constants` table where `sys\_type` is 'WEB\_SALES' and `sys\_code` is 'PROPOSAL'.  
4. The URL for the web sales proposal should be constructed by concatenating the system description and the transaction ID.  
5. If the constructed URL is not NULL, the system should open the URL in a web browser.  
  
Definition of Done:  
- The button is visible and clickable on the user interface.  
- The transaction ID is correctly retrieved and handled based on the conditions specified.  
- The system description is correctly retrieved.  
- The URL is correctly constructed and opened in a web browser if it is not NULL.  
- All functionalities are tested and verified to work as expected.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Retrieve `trans\_id` from `azbj\_cq\_trans\_data`:  
 ```sql  
 SELECT trans\_id  
 INTO v\_trans\_id  
 FROM azbj\_cq\_trans\_data  
 WHERE application\_no = TO\_CHAR(NVL(:insured\_person.ip\_verf\_no, :insured\_person.ip\_sign\_card\_no));  
 ```  
  
- Retrieve `sys\_desc` from `azbj\_system\_constants`:  
 ```sql  
 SELECT sys\_desc  
 INTO v\_sys\_desc  
 FROM azbj\_system\_constants  
 WHERE sys\_type = 'WEB\_SALES' AND sys\_code = 'PROPOSAL';  
 ```

# View All Images Functionality

Type: IMAGE\_DET

Title: View All Images Functionality  
  
Acceptance Criteria:  
1. When the "View All Images" button is pressed, the system should create a directory on the client machine under `C:\temp\` with a subfolder named after the insured person's verification or sign card number.  
2. The system should navigate to the `IMAGE\_DET` block and iterate through all records.  
3. For each record, if the `image\_path` is not null, the system should:  
 - Copy the image from the server to the client machine's directory created in step 1.  
 - If the image is a PDF, open it using `explorer.exe`.  
 - If the image is not a PDF, open it using the default image viewer.  
4. After processing all records, the system should hide the `WEBUTIL\_CANVAS` and set focus back to the `IMAGE\_DET.VIEW\_IMAGE` item.  
  
Definition of Done:  
- The "View All Images" button functionality is implemented and tested.  
- The system successfully creates the directory and copies images as specified.  
- The system opens the images in the appropriate viewer based on the file type.  
- The system handles exceptions gracefully without crashing.  
- The functionality is documented and reviewed.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Not applicable as the provided logic does not include direct CRUD operations on the database tables.

# View Image Functionality

Type: IMAGE\_DET

Title: View Image Functionality  
  
Acceptance Criteria:  
1. When the "View Image" button is pressed, the system should create a directory on the client machine at `C:\temp\` followed by the proposal number.  
2. The system should construct the client path using the proposal number and image name.  
3. If the application version is 10.x, the system should transfer the image from the server to the client with a progress indicator.  
4. The system should check the file size of the transferred image. If the file size is zero, an error message "Image is corrupted. Cannot load image" should be displayed.  
5. If the application version is not 10.x and the form name is 'NEW\_BBU' with a non-eligible flag set to 'N', the system should transfer the image using a URL format.  
6. If the image name contains 'VIDEO' and 'PIVC', the system should make the PIVC video item visible and enabled, then navigate to the PIVC video item and execute its associated trigger.  
7. If the image name contains 'DIGITAL' and 'PIVC', the system should set a flag indicating a digital PIVC view.  
8. If the image is in PDF format, the system should open it using `explorer.exe`. For other formats, it should use `rundll32.exe` to open the image in fullscreen mode.  
9. The system should log relevant actions and validations, including age calculations and product definitions, to ensure compliance with business rules.  
  
Definition of Done:  
- The "View Image" button functionality is implemented and tested.  
- The system correctly handles image transfers and displays appropriate error messages for corrupted images.  
- The system supports both PDF and other image formats.  
- The system logs all necessary actions and validations.  
- The PIVC video functionality is correctly enabled and triggered when applicable.  
- The feature is tested and verified to work on the client machine.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Not applicable as the provided XML content does not include direct CRUD operations on database tables.

# Input and Manage Minor Life Details

Type: MLQ

Detailed description: As a user, I want to input and manage minor life details, including class/standard, father's annual income, and mother's annual income, so that I can maintain accurate and up-to-date information for each minor.  
  
Acceptance criteria:  
1. The user should be able to input the class/standard of the minor.  
2. The user should be able to input the father's annual income.  
3. The user should be able to input the mother's annual income.  
4. The user should be able to navigate back to the previous screen using a "Back" button.  
  
Definition of Done:  
1. The input fields for class/standard, father's annual income, and mother's annual income are present and functional.  
2. The "Back" button is present and functional, allowing the user to return to the previous screen.  
3. The input fields should accept and store the data correctly.  
4. The user interface should be intuitive and user-friendly.  
5. The changes should be saved and retrievable for future reference.  
  
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 can select a class or standard from a predefined list

Type: MLQ

Title: User can select a class or standard from a predefined list  
  
Acceptance Criteria:  
- When the user double-clicks on the "Class \ Std" field, a list of values (LOV) should be displayed.  
- The LOV should be named 'ag\_qual' and should contain relevant class or standard options.  
- The user should be able to select a value from the LOV, and the selected value should be populated in the "Class \ Std" field.  
  
Definition of Done:  
- The "Class \ Std" field should be interactive and allow double-click actions.  
- The LOV should be correctly configured and display the appropriate options.  
- The selected value from the LOV should be correctly populated in the "Class \ Std" field.  
- The functionality should be 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.

# Navigate Back from Minor Life Details Screen

Type: MLQ

Title: Navigate Back from Minor Life Details Screen  
  
User Story:  
As a user, I want to navigate back to the previous screen from the "Minor Life Details" screen by clicking a button, so that I can easily return to the previous section without manually navigating through the application.  
  
Acceptance Criteria:  
1. When the "Back" button is pressed, the application should navigate to the previous screen.  
2. The value of the variable `v\_IP\_MINOR\_QST\_FLAG` should be set to the value of `IP\_MINOR\_QST\_FLAG` from the "INSURED\_PERSON" section.  
  
Definition of Done:  
1. The "Back" button is visible and functional on the "Minor Life Details" screen.  
2. Pressing the "Back" button successfully navigates the user to the previous screen.  
3. The variable `v\_IP\_MINOR\_QST\_FLAG` is correctly assigned the value from `IP\_MINOR\_QST\_FLAG` in the "INSURED\_PERSON" section.  
4. All navigation and variable assignments are tested and verified to work as expected.  
  
SQL Query for Reference:  
```sql  
-- Navigate to the previous screen  
Go\_ITEM('CONTROL.PB\_MLQ\_DTLS');  
  
-- Assign the value of IP\_MINOR\_QST\_FLAG to v\_IP\_MINOR\_QST\_FLAG  
pk\_vars.v\_IP\_MINOR\_QST\_FLAG := :INSURED\_PERSON.IP\_MINOR\_QST\_FLAG;  
```  
  
Block Name: MLQ

# Manage Life Insurance Policy Details

Type: MLQ\_DTLS

Title: Manage Life Insurance Policy Details  
  
Acceptance Criteria:  
1. The user should be able to input and update the relationship with the main life insured.  
2. The user should be able to input and update the company name.  
3. The user should be able to input and update the sum assured, which should be a numeric value.  
4. The user should be able to input and update the premium amount, which should be a numeric value.  
  
Definition of Done:  
1. The user interface should allow for the input and update of the relationship with the main life insured, company name, sum assured, and premium amount.  
2. The data should be stored in the database and be retrievable for display and further updates.  
3. The input fields should have appropriate validation to ensure data integrity (e.g., numeric fields for sum assured and premium).  
4. The user interface should be intuitive and user-friendly, with clear prompts and labels for each input field.  
  
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 Relationship Details with ML

Type: MLQ\_DTLS

Title: Manage Relationship Details with ML  
  
Acceptance Criteria:  
1. The system should allow the user to insert new relationship details with ML.  
2. The system should allow the user to update existing relationship details with ML.  
3. The relationship details should be displayed in a list format with a maximum length of 25 characters.  
4. The relationship details should be visually distinct with a white background and bold prompt text.  
5. The relationship details should be accessible under the "New Business" section.  
  
Definition of Done:  
1. The user can successfully add new relationship details with ML.  
2. The user can successfully update existing relationship details with ML.  
3. The relationship details are displayed correctly in the specified format.  
4. The relationship details are accessible under the "New Business" section and are visually distinct as specified.  
  
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.

# Input and Save Signing Date and Witness Name

Type: BLOCK\_MWP\_ACT

Title: Input and Save Signing Date and Witness Name  
  
Acceptance Criteria:  
1. The "SIGN\_DATE" field should allow the user to input a date in the format "dd/mm/yyyy".  
2. The "WITNESS\_NAME" field should allow the user to input the name of the witness and support multiline text.  
3. There should be a "Save" button that, when clicked, saves the entered data.  
  
Definition of Done:  
1. The "SIGN\_DATE" field is present and accepts date input in the specified format.  
2. The "WITNESS\_NAME" field is present and accepts multiline text input.  
3. The "Save" button is present and functional, saving the entered data when clicked.  
4. The form layout is user-friendly and visually consistent with the specified design.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include specific database queries.

# Implement Witness Name Input Field

Type: BLOCK\_MWP\_ACT

Title: Implement Witness Name Input Field  
  
Acceptance Criteria:  
1. The text field for the witness name should be clearly visible and positioned appropriately within the form.  
2. The text field should allow multiline input to accommodate longer names or additional details.  
3. Upon pressing the designated key to move to the next item, the system should automatically navigate to the save button to facilitate quick saving of the entered witness name.  
  
Definition of Done:  
1. The text field for the witness name is implemented and visible in the form.  
2. The text field supports multiline input.  
3. The navigation logic to move to the save button upon pressing the next item key is implemented and tested.  
4. The feature is tested and verified to ensure it meets the acceptance criteria.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include any specific database queries.

# Save Beneficiary and Trustee Information

Type: BLOCK\_MWP\_ACT

Title: Save Beneficiary and Trustee Information  
  
Acceptance Criteria:  
1. When the save button is pressed, the system should delete any existing records for the given contract ID from the `azbj\_beneficiary\_trustee\_rep` table.  
2. The system should then check if the mandatory fields (`BENEFICIARY\_NAME`, `TRUSTEE\_NAME`, and `WITNESS\_NAME`) are filled:  
 - If `BENEFICIARY\_NAME` is null, display a warning message "At least one beneficiary is mandatory" and stop the process.  
 - If `TRUSTEE\_NAME` is null, display a warning message "At least one Trustee is mandatory" and stop the process.  
 - If `WITNESS\_NAME` is null, display a warning message "Witness name cannot be null" and stop the process.  
3. If all mandatory fields are filled, the system should insert the new records into the `azbj\_beneficiary\_trustee\_rep` table with the provided details.  
4. After successful insertion, display a message "Saved Successfully" and clear the form fields.  
  
Definition of Done:  
- The save functionality should be implemented and tested.  
- All mandatory fields should be validated before saving.  
- Appropriate warning messages should be displayed if any mandatory field is missing.  
- Data should be correctly inserted into the `azbj\_beneficiary\_trustee\_rep` table.  
- The form should be cleared after successful save.  
  
DB queries for Table reference CRUD operations only(With Usage):  
```sql  
-- Delete existing records for the given contract ID  
DELETE FROM customer.azbj\_beneficiary\_trustee\_rep WHERE contract\_id = :control.cn\_contract\_id;  
COMMIT;  
  
-- Insert new records into the table  
INSERT INTO customer.azbj\_beneficiary\_trustee\_rep (  
 BEN\_NO,  
 BEN\_NAME,  
 BEN\_DOB,  
 BEN\_SHARE,  
 BEN\_RELATION,  
 top\_indicator,  
 contract\_id,  
 TRUSTEE\_NAME,  
 TRUSTEE\_ADDRESS,  
 SIGN\_DATE,  
 WITNESS\_NAME  
) VALUES (  
 :i,  
 :MWP\_ACT.BENEFICIARY\_NAME,  
 :MWP\_ACT.DOB,  
 :MWP\_ACT.SHARE,  
 :MWP\_ACT.RELATIONSHIP,  
 'Y',  
 :control.cn\_contract\_id,  
 :MWP\_ACT.TRUSTEE\_NAME,  
 :MWP\_ACT.address,  
 :BLOCK\_MWP\_ACT.SIGN\_DATE,  
 :BLOCK\_MWP\_ACT.WITNESS\_NAME  
);  
COMMIT;  
```

# Manage Beneficiary and Trustee Information

Type: MWP\_ACT

Title: Manage Beneficiary and Trustee Information  
  
Acceptance Criteria:  
1. The system should display a list of beneficiaries with the following details:  
 - Serial Number (Read-only)  
 - Beneficiary Name  
 - Date of Birth (in dd/mm/yyyy format)  
 - Share (percentage)  
 - Relationship (dropdown with predefined options)  
2. The system should allow the user to input trustee information with the following details:  
 - Serial Number (Read-only)  
 - Trustee Name (mandatory)  
 - Address  
  
Definition of Done:  
- The user can view and input beneficiary and trustee information as specified.  
- The system should validate the mandatory fields and ensure the correct format for dates.  
- The user interface should be intuitive and user-friendly, with appropriate field labels and input controls.  
  
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.

# Automatic Navigation from Address to Witness Name Field

Type: MWP\_ACT

Title: Automatic Navigation from Address to Witness Name Field  
  
Acceptance Criteria:  
- When the user enters an address and presses the key to move to the next item, the system should automatically focus on the witness name field.  
  
Definition of Done:  
- The address field is functional and allows user input.  
- Upon pressing the key to move to the next item, the system successfully navigates to the witness name field.  
- The navigation logic is implemented and tested to ensure it works as expected.

# Validate Beneficiary Name Field

Type: MWP\_ACT

Title: Validate Beneficiary Name Field  
  
Acceptance Criteria:  
- If the beneficiary name field is left blank, the system should display a warning message stating "Beneficiary name cannot be null".  
- The system should prevent the user from proceeding until a valid beneficiary name is entered.  
  
Definition of Done:  
- The beneficiary name field is validated to ensure it is not empty.  
- A warning message is displayed if the field is left blank.  
- The user is unable to proceed without entering a valid beneficiary name.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include any specific database queries.

# Validate Date of Birth against Opus Date

Type: MWP\_ACT

Title: Validate Date of Birth against Opus Date  
  
Acceptance Criteria:  
1. When the user enters a DOB, the system should validate it against the opus date.  
2. If the DOB is the same as or greater than the opus date, the system should display a warning message: "The Date of birth cannot be same as or greater than opus date".  
3. The system should prevent the user from proceeding if the DOB is invalid by raising an error.  
  
Definition of Done:  
1. The DOB field is present and functional in the user interface.  
2. The validation logic for DOB against the opus date is implemented and tested.  
3. The warning message is displayed correctly when the validation fails.  
4. The user is prevented from proceeding with an invalid DOB.  
5. All acceptance criteria are met and verified through testing.

# Validation for Trustee Name Field

Type: MWP\_ACT

Title: Validation for Trustee Name Field  
  
Acceptance Criteria:  
1. The "Trustee Name" field must not be left empty.  
2. If the "Trustee Name" field is empty, the system should display a warning message: "Trustee name cannot be null".  
3. The system should prevent the form from being submitted if the "Trustee Name" field is empty.  
  
Definition of Done:  
1. The "Trustee Name" field is marked as required.  
2. A validation check is implemented to ensure the "Trustee Name" field is not empty.  
3. A warning message is displayed if the "Trustee Name" field is empty.  
4. The form submission is blocked if the "Trustee Name" field is empty.  
5. All acceptance criteria are met and tested successfully.

# User can select a relationship type from a predefined list in MWP\_ACT module

Type: MWP\_ACT

Detailed description: As a user, I want to select a relationship type from a predefined list when interacting with the MWP\_ACT module, so that I can accurately categorize the relationship within the system.  
  
Acceptance criteria:  
1. The relationship type should be selectable from a list containing three predefined options.  
2. The list should be displayed in a user-friendly manner, with clear and readable text.  
3. The selection should be optional, meaning the user is not required to make a selection to proceed.  
4. The list should be positioned appropriately within the user interface for easy access.  
  
Definition of Done:  
1. The list of relationship types is implemented and contains exactly three options.  
2. The list is displayed with a white background and black text for readability.  
3. The list is positioned at the specified coordinates within the user interface.  
4. The user interface is tested to ensure the list is accessible and functional.  
5. 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 specific database queries or operations.

# User can enter and save a reason for exiting

Type: BLK\_SV\_EX\_RSN

Detailed description: As a user, I want to be able to enter and save a reason for exiting a particular process, so that I can document the reason for exiting and ensure it is recorded for future reference.  
  
Acceptance criteria:  
1. The user should be able to enter a reason for exiting in a multiline text field.  
2. The text field should allow a maximum of 4000 characters.  
3. There should be a "Save" button that, when clicked, saves the entered reason.  
4. There should be an "Exit" button that, when clicked, exits the form without saving.  
5. The form should display a confirmation window when the "Exit" button is clicked, asking the user to confirm their action.  
  
Definition of Done:  
1. The user can successfully enter and save a reason for exiting.  
2. The reason is stored in the database and can be retrieved later.  
3. The "Save" and "Exit" buttons function as expected.  
4. The confirmation window appears when the "Exit" button is clicked, and the user can confirm or cancel the exit action.  
  
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.

# Exit Button Functionality

Type: BLK\_SV\_EX\_RSN

Detailed description: As a user, I want to be able to exit the current screen and navigate to the "End Movement" screen when I press the exit button, ensuring that certain conditions are met before the transition.  
  
Acceptance criteria:  
1. If the field `sv\_ex\_rsn` in the current block is empty, the flag `sv\_ex\_rsn\_flg` should be set to 'N'.  
2. The current window and canvas should be hidden.  
3. The system should navigate to the "End Movement" screen and focus on the "Action" item within that screen.  
4. Any exceptions during this process should be handled gracefully without causing the application to crash.  
  
Definition of Done:  
- The exit button functionality is implemented and tested.  
- The conditions for setting the flag and hiding the window/canvas are verified.  
- Navigation to the "End Movement" screen is confirmed.  
- Exception handling is in place and tested.  
  
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.

# Save Reason for Exit

Type: BLK\_SV\_EX\_RSN

Detailed description: As a user, I want to be able to save a reason for exiting a specific process so that I can document the reason and proceed with the next steps.  
  
Acceptance criteria:  
1. When the save button is pressed, the system should check if the reason field is empty.  
 - If the reason field is empty, display an error message: "Please enter reason for Save & Exit and proceed."  
2. If the reason field is not empty, the system should:  
 - Retrieve the maximum event number associated with the current contract ID and increment it by one.  
 - Insert a new record into the comments table with the incremented event number, contract ID, policy number, user ID, current date, the entered reason, and a flag set to 'Y'.  
 - Clear the reason field.  
 - Set a flag indicating that the reason has been saved.  
 - Display a success message: "Reason saved successfully, please continue."  
 - Navigate to the next block and item for further actions.  
3. If any database errors occur during the process, appropriate error handling should be in place to log the error and display a relevant message.  
  
Definition of Done:  
- The save button functionality is implemented and tested.  
- The system correctly handles empty reason fields by displaying an error message.  
- The system successfully inserts a new record into the comments table when a reason is provided.  
- The reason field is cleared, and the success message is displayed upon successful insertion.  
- The system navigates to the next block and item after saving the reason.  
- Error handling is implemented to log and display relevant error messages.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Retrieve the maximum event number:  
 ```sql  
 SELECT NVL (MAX (event\_no), 0) + 1  
 INTO v\_event\_no  
 FROM azbj\_uw\_comments  
 WHERE contract\_id = :control.cn\_contract\_id;  
 ```  
  
- Insert a new comment record:  
 ```sql  
 INSERT INTO azbj\_uw\_comments  
 (event\_no, contract\_id, policy\_no, user\_id, comment\_date, comments, flag)  
 VALUES (v\_event\_no, :control.cn\_contract\_id, :control.cn\_policy\_ref, :boiler.userid, SYSDATE, :blk\_sv\_ex\_rsn.sv\_ex\_rsn, 'Y');  
 ```

# Automatic Update of Lifestyle Group Status Based on Product Conditions

Type: LIFE\_STYLE\_RAD

Title: Automatic Update of Lifestyle Group Status Based on Product Conditions  
  
Acceptance Criteria:  
1. When the product ID is 345 and the primary lifestyle group status is 'P', the system should automatically change the primary lifestyle group status to 'N'.  
2. When the product ID is 345 and the secondary lifestyle group status is 'P', the system should automatically change the secondary lifestyle group status to 'N'.  
3. The system should log the changes made to the lifestyle group status, including contract ID, verification number or sign card number, and the updated statuses.  
  
Definition of Done:  
- The system correctly updates the lifestyle group status based on the specified conditions.  
- The changes are logged with the necessary details.  
- The functionality is tested and verified to ensure accuracy and reliability.  
  
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.

# Display and Interact with Lifestyle Radio Group

Type: LIFE\_STYLE\_RAD

Title: Display and Interact with Lifestyle Radio Group  
  
Acceptance Criteria:  
- The radio group should be displayed on the "LIFE\_STYLE\_DESC" tab.  
- The radio group should have a default value of "S".  
- The radio group should be read-only, meaning users cannot insert or update values.  
- The radio group should be positioned at coordinates (75, 81) on the tab.  
- The radio group should have a width of 90 and a height of 18.  
- The radio group should have a gray background and black foreground color.  
  
Definition of Done:  
- The radio group is visible on the "LIFE\_STYLE\_DESC" tab.  
- The radio group displays the default value "S".  
- The radio group is read-only.  
- The radio group is correctly positioned and sized.  
- 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 specific database queries.

# Display and Interact with Lifestyle Radio Group on New Business Form

Type: LIFE\_STYLE\_RAD

Title: Display and Interact with Lifestyle Radio Group on New Business Form  
  
Acceptance Criteria:  
1. The radio group should be displayed on the "New Business" form.  
2. The radio group should have a default value of "S".  
3. The radio group should be read-only and not allow insert or update operations.  
4. The radio group should be positioned at coordinates (513, 77) on the form.  
5. The radio group should have a width of 90 and a height of 18.  
6. The radio group should have a gray background and black foreground color.  
  
Definition of Done:  
1. The radio group is visible on the "New Business" form.  
2. The radio group displays the default value "S".  
3. The radio group is read-only and does not allow insert or update operations.  
4. The radio group is correctly positioned and sized on the form.  
5. 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 specific database queries.

# Manage Lifestyle Questions Interface

Type: LIFE\_STYLE

Title: Manage Lifestyle Questions Interface  
  
Acceptance Criteria:  
1. The interface should display fields for entering a question ID, question description, value, and result.  
2. The question ID should be a numeric field.  
3. The question description should allow up to 2000 characters.  
4. The value fields should be restricted to uppercase input and allow up to 2000 characters.  
5. The result fields should accept 'Y' or 'N' values.  
6. The interface should be visually organized with clear prompts for each field.  
  
Definition of Done:  
1. The user interface is implemented and displays all required fields.  
2. The fields adhere to the specified data types and input restrictions.  
3. The interface is tested for usability and correctness.  
4. The interface is integrated into the existing system and is functional.  
  
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.

# Update form status based on specific conditions in the Value field

Type: LIFE\_STYLE

Detailed description: As a user, I want the system to automatically set the form status to 'N' when specific conditions are met after entering a value in the "Value" field within the "Lifestyle" section, so that the form status is correctly updated based on the input.  
  
Acceptance criteria:  
- When the user enters a value in the "Value" field and the associated question ID is either 16 or 17, the form status should be set to 'N'.  
- The form status should not change for any other question IDs.  
  
Definition of Done:  
- The form status is correctly updated to 'N' when the conditions are met.  
- The form status remains unchanged for all other conditions.  
- 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.

# Manage Underwriting Questions Section

Type: AZBJ\_UW\_QUESTIONS

Detailed description: As a user, I want to manage underwriting questions within a dedicated section so that I can efficiently input and review question details.  
  
Acceptance criteria:  
1. The section should display a list of underwriting questions with the following fields:  
 - Question ID: A unique identifier for each question.  
 - Question Description: A detailed description of the question.  
 - Select questions: A checkbox to mark the question as selected.  
  
2. The section should allow for the following functionalities:  
 - Display up to 10 records at a time.  
 - Scroll through the list of questions if there are more than 10 records.  
 - Ensure that the Question ID and Question Description fields are editable.  
 - The checkbox should allow users to mark a question as selected (checked) or not selected (unchecked).  
  
Definition of Done:  
- The section is implemented and displays the underwriting questions as specified.  
- Users can scroll through the list if there are more than 10 questions.  
- Users can edit the Question ID and Question Description fields.  
- Users can select or deselect questions using the checkbox.  
- The section is tested and verified to meet the acceptance criteria.  
  
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.

# CRM Comments Section

Type: CRM\_COMMENTS

Title: CRM Comments Section  
  
Acceptance Criteria:  
1. The section should display up to 8 records at a time.  
2. Each record should have fields for:  
 - Question (text, up to 1000 characters)  
 - Comment (text, up to 1000 characters)  
 - User (text, up to 30 characters)  
 - Date (date format)  
3. The interface should be visually organized with appropriate spacing and alignment for ease of use.  
4. The background color of the section should be gray, and the text fields should have a white background with black text.  
5. The section should include a scrollbar to navigate through the records if there are more than 8.  
  
Definition of Done:  
- The section is implemented and integrated into the application.  
- The section meets all the acceptance criteria.  
- The section is tested and verified to ensure it functions correctly and displays the correct information.  
- User feedback is collected and any necessary adjustments are made.  
  
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.

# Display Reason for Manual BBU in Underwriting Comments

Type: RES\_MAN\_BBU

User Story: Display Reason for Manual BBU in Underwriting Comments  
  
Detailed Description:  
As an underwriter, I want to see the reason for pushing a case to manual BBU displayed in the underwriting comments, so that I can understand the context and rationale behind the decision.  
  
Acceptance Criteria:  
1. When a new block instance is created, the system should fetch the reason for pushing the case to manual BBU from the database.  
2. If the reason is related to 'QC REQUIREMENTS' and the rule message is not 'QC Requirements raised.', the system should fetch the detailed question description from the database and display it.  
3. If the reason is not related to 'QC REQUIREMENTS' or the rule message is 'QC Requirements raised.', the system should display the rule message as is.  
4. The system should handle any exceptions during the database fetch operations gracefully and ensure that the underwriting comments are updated accordingly.  
  
Definition of Done:  
- The reason for manual BBU is correctly displayed in the underwriting comments.  
- The system handles different scenarios for 'QC REQUIREMENTS' and other reasons appropriately.  
- All database fetch operations are executed without errors, and exceptions are handled gracefully.  
- The feature is tested and verified by the QA team.  
  
DB Queries for Table Reference CRUD Operations:  
- Fetching reasons and rule messages from `bbu\_trans` and `bbu\_trans\_dtls` tables.  
- Fetching question descriptions from `azbj\_qc\_questions` table.  
  
```sql  
-- Example query to fetch reasons and rule messages  
SELECT   
 CASE  
 WHEN source\_flag = 'BBU' THEN param\_val\_string  
 ELSE 'Rule Configurator'  
 END AS param\_type,  
 rule\_message,   
 a.trans\_id  
FROM   
 bbu\_trans a,   
 bbu\_trans\_dtls b  
WHERE   
 a.trans\_id = b.trans\_id   
 AND action\_id = 2  
 AND a.appl\_no = '1100298672';  
  
-- Example query to fetch question descriptions  
SELECT   
 question\_desc  
FROM   
 azbj\_qc\_questions  
WHERE   
 question\_id = (  
 SELECT   
 SUBSTR(rule\_message, 37)  
 FROM   
 bbu\_trans\_dtls  
 WHERE   
 trans\_id = :trans\_id  
 AND param\_val\_string = 'QC REQUIREMENTS'  
 AND action\_id = 2  
 AND rule\_message != 'QC Requirements raised.'  
 AND rule\_message = :rule\_message  
 );  
```  
  
Note: The above queries are provided for reference and should be adapted as per the actual database schema and requirements.

# Exit Button Functionality

Type: RES\_MAN\_BBU

User Story: Exit Button Functionality  
  
Detailed description: As a user, I want to be able to exit the current view and navigate to a different section of the application when I press the "Exit" button, so that I can efficiently manage my workflow.  
  
Acceptance criteria:  
1. When the "Exit" button is pressed, the system should navigate to the 'CLIENT\_ENV.RES\_FOR\_MAN' section.  
2. The current view and window named 'RES\_MAN\_BBU' should be hidden.  
3. The 'NBTABS' view should be displayed.  
  
Definition of Done:  
- The "Exit" button should be functional and perform the specified navigation and view changes.  
- The system should hide the current view and window and display the new view without any errors.  
- The functionality should be tested and verified to ensure it meets the acceptance criteria.  
  
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 that can be executed independently of Oracle Forms constructs.

# Validate Beneficial Owner Record Details

Type: CFT

Title: Validate Beneficial Owner Record Details  
  
Acceptance Criteria:  
1. When a beneficial owner record is saved, the system should validate that the following fields are not null:  
 - Beneficial Owner Name  
 - Shares  
 - Identification Proof  
 - Identification Document  
 - Identification Date  
 - Address Proof  
 - Address ID  
 - Address Date  
 - Address  
 - Date of Birth  
2. If any of the above fields are null, the system should display an error message: "Please enter all details."  
  
Definition of Done:  
- The system performs the validation check when a beneficial owner record is saved.  
- An error message is displayed if any required field is missing.  
- The validation logic is tested and confirmed to work as expected.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- The system should use the following query to fetch identification proof details:  
 ```sql  
 SELECT document\_type, proof\_desc  
 FROM azbj\_aml\_proof\_master  
 WHERE partner\_type = :control.cn\_partner\_type;  
 ```  
- The system should use the following query to fetch address proof details:  
 ```sql  
 SELECT document\_type, proof\_desc  
 FROM azbj\_aml\_proof\_master  
 WHERE partner\_type = :control.cn\_partner\_type;  
 ```

# Address Proof LOV Selection

Type: CFT

Title: Address Proof LOV Selection  
  
Acceptance Criteria:  
1. When the user double-clicks on the address proof field, the system should check if the document type is not null.  
2. The system should then execute a query to fetch the document type and proof description from the `azbj\_aml\_proof\_master` table where the partner type matches the current partner type.  
3. If the query execution is successful, the system should display the LOV with the fetched values.  
4. If the query execution fails, the system should disable the validation from the list for the address proof field.  
5. The system should handle any exceptions and display an appropriate error message.  
  
Definition of Done:  
- The address proof field should display a LOV when double-clicked, provided the document type is not null.  
- The LOV should be populated with document types and proof descriptions based on the partner type.  
- Proper error handling should be in place to manage query execution failures and other exceptions.  
  
DB queries for Table reference CRUD operations only(With Usage):  
```sql  
SELECT document\_type, proof\_desc  
FROM azbj\_aml\_proof\_master  
WHERE partner\_type = :control.cn\_partner\_type;  
```

# Identification Proof Selection

Type: CFT

Title: Identification Proof Selection  
  
Acceptance Criteria:  
1. When the user double-clicks on the identification proof field, the system should check if the document type is not null.  
2. If the document type is not null, the system should query the `azbj\_aml\_proof\_master` table to retrieve the document type and proof description based on the partner type.  
3. The system should populate the list of values with the retrieved data.  
4. If the query is successful, the identification proof field should be validated against the list of values.  
5. If the query fails, the identification proof field should not be validated against the list of values.  
6. The list of values should be displayed to the user for selection.  
  
Definition of Done:  
- The identification proof field should display a list of values when double-clicked.  
- The list of values should be populated based on the document type and partner type.  
- The identification proof field should be validated against the list of values if the query is successful.  
- The identification proof field should not be validated against the list of values if the query fails.  
- The user should be able to select a value from the list of values.  
  
DB queries for Table reference CRUD operations only(With Usage):  
```sql  
SELECT document\_type, proof\_desc  
FROM azbj\_aml\_proof\_master  
WHERE partner\_type = :control.cn\_partner\_type;  
```

# View and Manage Risk Score Details

Type: AZBJ\_RISK\_SCORE\_DTLS

Title: View and Manage Risk Score Details  
  
Acceptance Criteria:  
1. The user should be able to view a list of risk score details with the following fields:  
 - Parameters  
 - Value in Proposal  
 - Branch Risk Score  
 - DE-QC Risk Score  
 - Total Risk Score  
2. Each field should have appropriate labels and be displayed in a user-friendly manner.  
3. The user should be able to insert new records, update existing records, and scroll through the list of records.  
4. The interface should be visually appealing with consistent font styles and colors.  
  
Definition of Done:  
1. The user interface displays the risk score details as specified.  
2. The user can successfully insert, update, and scroll through records.  
3. The interface meets the visual design requirements.  
4. All acceptance criteria are met and tested.  
  
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 Incomplete Proposal Details

Type: AZBJ\_INCMPLT\_PROP\_DTLS

Title: Manage Incomplete Proposal Details  
  
Acceptance Criteria:  
1. The system should display a list of proposal form fields in a dropdown menu, populated from a predefined list of values.  
2. The user should be able to add comments to each proposal record.  
3. The user should be able to mark whether the details have been received using a dropdown menu.  
4. The user should be able to save the current record.  
5. The user should be able to add a new row for additional proposal details.  
6. The user should be able to delete a selected row by checking a checkbox and clicking the delete button.  
7. The user should be able to navigate back to the previous screen.  
  
Definition of Done:  
1. The dropdown menu for proposal form fields is populated with values from the database.  
2. The comments field allows text input up to 1000 characters.  
3. The details received field is a dropdown with predefined values.  
4. The save button saves the current record to the database.  
5. The add row button inserts a new blank row for additional proposal details.  
6. The delete row button removes the selected row from the database.  
7. The back button navigates the user to the previous screen.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Populate dropdown menu for proposal form fields:  
 ```sql  
 SELECT screen\_value  
 FROM inf\_dnm\_poplists  
 WHERE poplist\_code = 'PROPOSAL\_FORM\_FIELDS' AND LANGUAGE = 'US'  
 ORDER BY SCREEN\_VALUE ASC;  
 ```

# Delete Multiple Records from Incomplete Property Details

Type: AZBJ\_INCMPLT\_PROP\_DTLS

Title: Delete Multiple Records from Incomplete Property Details  
  
Acceptance Criteria:  
1. When the delete button is pressed, the system should navigate to the incomplete property details section.  
2. The system should iterate through all records in the incomplete property details section.  
3. For each record, if the delete checkbox is selected, the record should be deleted.  
4. The process should continue until all records have been checked.  
5. The process should stop if the last record is reached.  
  
Definition of Done:  
- The delete button is functional and deletes all selected records from the incomplete property details section.  
- The system correctly iterates through all records and deletes only those with the delete checkbox selected.  
- The process stops correctly 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.

# Navigation from Incomplete Property Details to Further Requirements

Type: AZBJ\_INCMPLT\_PROP\_DTLS

Title: Navigation from Incomplete Property Details to Further Requirements  
  
Acceptance Criteria:  
1. When the "BACK" button is pressed, the current view of the incomplete property details section should be hidden.  
2. The system should navigate to the further requirements section.  
3. The first record in the further requirements section should be displayed.  
4. If any error occurs during this process, it should be handled gracefully without any visible impact to the user.  
  
Definition of Done:  
- The "BACK" button is functional and performs the described actions.  
- The navigation between sections is seamless and does not cause any disruption to the user experience.  
- Error handling is implemented to ensure no unhandled exceptions occur during the navigation process.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include any specific database queries.

# Selection Field for Details Received

Type: AZBJ\_INCMPLT\_PROP\_DTLS

Title: Selection Field for Details Received  
  
Acceptance Criteria:  
- The user should see a prompt labeled "Details Received" at the top of the selection field.  
- The selection field should have two options: "Yes" and "No".  
- The default value of the selection field should be "No".  
- The selection field should be clearly visible and accessible on the screen.  
  
Definition of Done:  
- The selection field for "Details Received" is implemented and visible on the screen.  
- The prompt "Details Received" is correctly aligned and positioned above the selection field.  
- The selection field defaults to "No" when the form is initialized.  
- The selection field allows the user to choose between "Yes" and "No".  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include any specific database queries.

# Display and Interact with Proposal Form Field

Type: AZBJ\_INCMPLT\_PROP\_DTLS

Title: Display and Interact with Proposal Form Field  
  
Acceptance Criteria:  
1. The proposal form field should be displayed with a label "Proposal Form Field" at the top.  
2. The field should be positioned at coordinates (16, 92) within the incomplete proposal details section.  
3. The field should have a width of 231 units and a height of 22 units.  
4. The field should use the "Tahoma" font with a size of 8 points and a demilight weight.  
5. The field should have a white background and black foreground color.  
6. The field should be a dropdown list item that allows the user to select from a predefined list of values.  
7. The field should not allow direct insertion or updates by the user.  
8. The field should be associated with a list of values named "AZBJ\_PROP\_FORM\_FIELD".  
  
Definition of Done:  
1. The proposal form field is displayed correctly within the incomplete proposal details section.  
2. The field meets all specified positioning, size, and styling requirements.  
3. The field functions as a dropdown list, displaying the predefined list of values.  
4. The field does not allow direct insertion or updates by the user.  
5. The field label "Proposal Form Field" is displayed correctly above the field.  
6. All acceptance criteria are met and verified through testing.

# Add New Record to Incomplete Proposal Form Details

Type: AZBJ\_INCMPLT\_PROP\_DTLS

Title: Add New Record to Incomplete Proposal Form Details  
  
Acceptance Criteria:  
1. When the "Add Row" button is pressed, an alert should be displayed asking for confirmation to add a new record.  
2. If the user confirms the action, the system should navigate to the Incomplete Proposal Form Details section.  
3. The system should iterate through all existing records in the Incomplete Proposal Form Details section and set specific fields to be non-insertable and non-updatable.  
4. If any error occurs during the process, an appropriate error message should be displayed.  
  
Definition of Done:  
- The "Add Row" button should trigger the confirmation alert.  
- Upon confirmation, the system should navigate to the Incomplete Proposal Form Details section and process the records as specified.  
- All specified fields in the existing records should be set to non-insertable and non-updatable.  
- Error handling should be in place to display relevant error messages if any issues arise during the process.  
  
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.

# Save Incomplete Proposal Form Details

Type: AZBJ\_INCMPLT\_PROP\_DTLS

Title: Save Incomplete Proposal Form Details  
  
Acceptance Criteria:  
1. When the "Save" button is clicked, a confirmation alert should appear asking, "Do you really want to Save Incomplete Proposal Form Details?"  
2. If the user confirms the action, the system should set a flag indicating that the incomplete proposal details have been saved.  
3. If the user cancels the action, the system should not save the incomplete proposal details and should raise an error to halt the process.  
  
Definition of Done:  
- The "Save" button is functional and triggers the confirmation alert.  
- The system correctly handles user confirmation and cancellation, setting the appropriate flags and raising errors as needed.  
- The feature is tested and verified to work as expected without any Oracle Forms-specific dependencies.  
  
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 and Manage Old Policy Details

Type: OLD\_POLICY\_DTLS

Title: View and Manage Old Policy Details  
  
Acceptance Criteria:  
1. The system should display the following fields for old policy details:  
 - Previous Policy Number  
 - Name of Policyholder  
 - Partner ID  
 - PAN Number  
2. Each field should be clearly labeled and positioned for easy readability.  
3. There should be a button labeled "Exit" that allows the user to close the old policy details section.  
  
Definition of Done:  
1. The old policy details section is implemented and displays the required fields.  
2. The fields are labeled and positioned as specified.  
3. The "Exit" button is functional and closes the old policy details section.  
4. The feature is tested and verified to ensure it meets the acceptance criteria.  
  
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.

# Navigation from Old Policy Details to New Business Section

Type: OLD\_POLICY\_DTLS

Title: Navigation from Old Policy Details to New Business Section  
  
Acceptance Criteria:  
1. When the "Exit" button is pressed, the system should navigate to the "Previous Policy Details" section.  
2. The "Old Policy Details" section should be hidden.  
3. The "New Business" section should be displayed.  
  
Definition of Done:  
- The "Exit" button successfully navigates the user to the "Previous Policy Details" section.  
- The "Old Policy Details" section is hidden from view.  
- The "New Business" section is displayed to the user.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include any database queries.

# Display Partner Details in Read-Only Format

Type: BLK\_SUS\_CP\_DTLS

Title: Display Partner Details in Read-Only Format  
  
Acceptance Criteria:  
1. The partner details should include the following fields:  
 - Partner Type  
 - First Name  
 - Second Name  
 - Third Name  
 - Fourth Name  
 - Alias Name  
 - Date of Birth  
 - City  
 - Unique ID  
 - List Type  
2. All fields should be displayed in a read-only format.  
3. The Date of Birth field should be formatted as "dd/mm/yyyy".  
4. The fields should be aligned and displayed in a user-friendly manner.  
  
Definition of Done:  
1. The partner details are displayed as per the acceptance criteria.  
2. The fields are read-only and properly formatted.  
3. The Date of Birth is displayed in the correct format.  
4. The user interface is tested for usability and correctness.  
5. The feature is reviewed and approved by the stakeholders.

# Manage Duplicate Contact Information

Type: DUPLICATE\_CONTACT\_NO

Detailed description: As a user, I want to view and manage duplicate contact information, including policy numbers, agent codes, names, and contact details, so that I can ensure data accuracy and avoid redundancy.  
  
Acceptance criteria:  
1. The system should display a list of duplicate contact records with the following fields:  
 - Policy Number/Agent Code  
 - Name  
 - Contact  
 - IP Number  
 - Participant ID  
2. The Policy Number/Agent Code and Name fields should be read-only.  
3. The system should provide a checkbox labeled "CHKSUP" to mark records for further review or action.  
4. The system should allow scrolling through the list of duplicate contact records.  
  
Definition of Done:  
- The user can view a list of duplicate contact records with the specified fields.  
- The Policy Number/Agent Code and Name fields are read-only.  
- The user can mark records using the "CHKSUP" checkbox.  
- The user can scroll through the list of records.  
  
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.

# Display Fund Names and Percentages

Type: BLOCK\_FUNDS

Title: Display Fund Names and Percentages  
  
Acceptance Criteria:  
1. The interface should display a list of funds with their names and corresponding percentages.  
2. The fund names and percentages should be read-only and not editable by the user.  
3. The interface should be visually appealing, with clear labels and alignment for fund names and percentages.  
4. The interface should include a scrollbar to navigate through the list if it exceeds the display area.  
  
Definition of Done:  
1. The user interface displays the fund names and percentages as specified.  
2. The fund names and percentages are non-editable.  
3. The interface is visually aligned and user-friendly.  
4. The scrollbar is functional and allows users to navigate through the list.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Not applicable as the provided XML content does not include any database-specific queries or operations.

# Signature Confidence Details Screen

Type: BLK\_SIGN\_CONFID

Title: Signature Confidence Details Screen  
  
Acceptance Criteria:  
1. The screen should display the document name and signature confidence percentage.  
2. The document name and signature confidence percentage fields should be read-only.  
3. There should be an "Exit" button that allows the user to close the screen.  
4. There should be a "Master Signature" button that allows the user to access the master signature details.  
  
Definition of Done:  
1. The Signature Confidence Details screen is implemented and displays the document name and signature confidence percentage.  
2. The document name and signature confidence percentage fields are read-only.  
3. The "Exit" button successfully closes the screen.  
4. The "Master Signature" button successfully navigates to the master signature details.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include any specific database queries.

# Exit Button Functionality

Type: BLK\_SIGN\_CONFID

Title: Exit Button Functionality  
  
Acceptance Criteria:  
- When the "Exit" button is clicked, the system should navigate to the "Agents" screen.  
- The "Exit" button should be clearly labeled and easily identifiable.  
- The button should be positioned at coordinates (266, 269) on the screen.  
- The button should have a width of 110 and a height of 20.  
- The button should have a gray background and black text.  
- The button should use the "Tahoma" font with a size of 10 and a plain style.  
  
Definition of Done:  
- The "Exit" button is implemented and functional.  
- The button navigates to the "Agents" screen when clicked.  
- The button meets all specified design and positioning criteria.  
- The feature is tested and verified to work as expected.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include any database CRUD operations.

# Generate and Open Secure URL for Master Signature

Type: BLK\_SIGN\_CONFID

Title: Generate and Open Secure URL for Master Signature  
  
Acceptance Criteria:  
1. When the "Master Signature" button is pressed, the system should generate a secure URL using the `azbj\_encrypt\_dms\_link` function.  
2. The URL should be generated based on the following parameters:  
 - 'NB' as the first parameter.  
 - 'MASTER\_SIGN' as the second parameter.  
 - The verification number of the insured person (`IP\_VERF\_NO`) or, if not available, the sign card number (`IP\_SIGN\_CARD\_NO`).  
3. If the generated URL is not null, the system should open the URL in a web browser.  
  
Definition of Done:  
- The "Master Signature" button should be functional and trigger the URL generation process.  
- The URL should be generated securely and accurately based on the provided parameters.  
- The URL should open in a web browser if it is not null.  
- The functionality should be tested and verified to ensure it meets the acceptance criteria.

# View and Verify Alert Details

Type: BLOCK\_ALERT

Title: View and Verify Alert Details  
  
Acceptance Criteria:  
1. The alert details should include the following fields:  
 - Dating Back  
 - Date of Commencement  
 - Name  
 - Father's/Husband's Name  
 - Address Line 1  
 - Address Line 2  
 - Place/Area  
 - District  
 - State/Country  
 - PIN  
 - Name as per receipt  
 - Name as per CP  
 - Total premium received + SR  
 - Total Premium  
 - Excess BOP  
 - Confirmed Excess Premium  
 - Mobile Number  
 - E-mail ID  
2. The alert details should also include radio group options for:  
 - Backdating  
 - Dispatch  
 - Receipt  
 - Premium  
 - Rider  
 - Excess Premium  
 - Funds  
 - Mobile  
3. All fields should be read-only and not allow any updates or insertions.  
4. The layout should be user-friendly, with proper alignment and spacing for each field.  
  
Definition of Done:  
- The alert details are displayed in a structured and readable format.  
- All fields are non-editable and accurately reflect the data.  
- The user can view all necessary information without any missing details.  
- The layout is visually appealing and easy to navigate.  
  
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.

# User selection via radio group in alert list window

Type: BLOCK\_ALERT

Title: User selection via radio group in alert list window  
  
Acceptance Criteria:  
1. The radio group should be initialized with a default value of "X".  
2. The alert list window should be modal, ensuring that the user must interact with it before returning to the main application.  
3. The alert list window should have specific dimensions and positions to ensure it is displayed correctly on the screen.  
  
Definition of Done:  
1. The radio group is displayed within the alert list window.  
2. The radio group is initialized with the default value "X".  
3. The alert list window is modal and has the specified dimensions and positions.  
4. 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 XML content does not include any database-specific operations.

# Validation and Navigation for Alert Block Options

Type: BLOCK\_ALERT

Title: Validation and Navigation for Alert Block Options  
  
Acceptance Criteria:  
1. The system should validate that the options for backdating, dispatch, receipt, premium, rider, excess premium, and mobile are either 'N' or 'Y'.  
2. The system should check if the product ID is not in the list ('Y', 'P') or if it is in the list and the funds option is 'Y'.  
3. If the above conditions are met, the proceed button should be enabled and navigable.  
4. If the conditions are not met, the proceed button should be disabled and non-navigable.  
5. When the premium option is selected as 'Y', the system should navigate to the excess premium field.  
6. If the premium option is not 'Y', the system should clear the block, set the action to 'W', update the policy status, navigate to the insured person tab, and hide the alert list window.  
  
Definition of Done:  
- The proceed button is enabled or disabled based on the specified conditions.  
- The system navigates to the appropriate fields or blocks based on the selected options.  
- The block is cleared, action is set, policy status is updated, and the alert list window is hidden when the premium option is not 'Y'.  
- All functionalities are tested and verified to work as expected.

# Enable/Disable Proceed Button Based on Radio Button Selections and Conditions

Type: BLOCK\_ALERT

Title: Enable/Disable Proceed Button Based on Radio Button Selections and Conditions  
  
Acceptance Criteria:  
1. The "Proceed" button should be enabled and navigable if the following conditions are met:  
 - The values of the radio buttons for backdating, dispatch, receipt, premium, rider, and mobile are either 'N' or 'Y'.  
 - The value of the "Proceed" button control is 'X' or the value of the excess premium radio button is 'N' or 'Y'.  
 - The product ID is not linked to 'Y' or 'P'.  
 - Alternatively, if the product ID is linked to 'Y' or 'P' and the funds radio button is 'Y'.  
2. The "Proceed" button should be disabled and non-navigable if the above conditions are not met.  
3. When the funds radio button is changed to 'Y', the focus should move to the mobile radio button.  
4. If the funds radio button is not 'Y', the system should clear the current block, set specific control values, navigate to the insured person tab, and hide the alert list window.  
  
Definition of Done:  
- The "Proceed" button's state (enabled/disabled) and navigability should be correctly updated based on the specified conditions.  
- The focus should correctly move to the mobile radio button when the funds radio button is 'Y'.  
- The system should correctly clear the block, set control values, navigate to the insured person tab, and hide the alert list window when the funds radio button is not 'Y'.  
- All functionalities should be tested and verified to ensure they work as expected.

# Enable/Disable Proceed Button Based on Alert Options

Type: BLOCK\_ALERT

Title: Enable/Disable Proceed Button Based on Alert Options  
  
Acceptance Criteria:  
1. The "Proceed" button should be enabled and navigable if all the following conditions are met:  
 - The values of `RAD\_BACKDATION`, `RADIO\_DISPATCH`, `RAD\_RECEIPT`, `RAD\_PREMIUM`, `RAD\_RIDDER`, `RAD\_EXCESS\_PRE`, and `RAD\_MOB` are either 'N' or 'Y'.  
 - The value of `V\_PRCD\_BTN` is 'X' or `RAD\_EXCESS\_PRE` is 'N' or 'Y'.  
 - The function `unitlink` with the parameter `cn\_product\_id` does not return 'Y' or 'P'.  
 - Alternatively, if the function `unitlink` with the parameter `cn\_product\_id` returns 'Y' or 'P' and the value of `RAD\_FUNDS` is 'Y'.  
  
2. The "Proceed" button should be disabled and non-navigable if any of the above conditions are not met.  
  
3. When the `RAD\_RECEIPT` option is changed to 'Y', the focus should move to the `RAD\_PREMIUM` option.  
  
4. If the `RAD\_RECEIPT` option is not 'Y', the system should:  
 - Clear the current block.  
 - Set the action to 'W'.  
 - Update the policy status.  
 - Navigate to the "Insured Person" tab.  
 - Hide the alert window.  
  
Definition of Done:  
- The "Proceed" button's state (enabled/disabled) is correctly updated based on the specified conditions.  
- The focus correctly shifts to the `RAD\_PREMIUM` option when `RAD\_RECEIPT` is 'Y'.  
- The system performs the specified actions when `RAD\_RECEIPT` is not 'Y'.  
- All functionalities are tested and verified to work 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.

# Validation and Navigation for RADIO\_DISPATCH Option

Type: BLOCK\_ALERT

Title: Validation and Navigation for RADIO\_DISPATCH Option  
  
Acceptance Criteria:  
1. The system should validate the "RADIO\_DISPATCH" option when it is selected.  
2. The "Proceed" button should be enabled and navigable if the following conditions are met:  
 - The values of "RAD\_BACKDATION", "RADIO\_DISPATCH", "RAD\_RECEIPT", "RAD\_RIDDER", "RAD\_PREMIUM", "RAD\_EXCESS\_PRE", and "RAD\_MOB" are either 'N' or 'Y'.  
 - The value of "V\_PRCD\_BTN" is 'X' or "RAD\_EXCESS\_PRE" is 'N' or 'Y'.  
 - The function `unitlink` with the parameter `cn\_product\_id` does not return 'Y' or 'P'.  
 - Alternatively, if the function `unitlink` with the parameter `cn\_product\_id` returns 'Y' or 'P' and the value of "RAD\_FUNDS" is 'Y'.  
3. The "Proceed" button should be disabled and non-navigable if the above conditions are not met.  
4. When the "RADIO\_DISPATCH" option is changed to 'Y', the system should navigate to the "RAD\_RECEIPT" field.  
5. When the "RADIO\_DISPATCH" option is changed to any value other than 'Y', the system should:  
 - Clear the current block.  
 - Set the action to 'W'.  
 - Update the policy status.  
 - Navigate to the "INSURED\_PERSON" tab and block.  
 - Hide the alert window.  
  
Definition of Done:  
- The validation logic for the "RADIO\_DISPATCH" option is implemented and tested.  
- The "Proceed" button's state (enabled/disabled) is correctly updated based on the specified conditions.  
- The navigation logic for the "RADIO\_DISPATCH" option change is implemented and tested.  
- The system correctly handles the clearing of the block, setting the action, updating the policy status, navigating to the "INSURED\_PERSON" tab, and hiding the alert window.  
- All acceptance criteria are met and verified through testing.

# Enable/Disable Proceed Button Based on Radio Button Conditions

Type: BLOCK\_ALERT

Title: Enable/Disable Proceed Button Based on Radio Button Conditions  
  
Acceptance Criteria:  
1. The "Proceed" button should be enabled and navigable if the following conditions are met:  
 - The values of the radio buttons for backdating, dispatch, receipt, premium, and rider are either 'N' or 'Y'.  
 - The value of the "Proceed" button control is 'X' or the value of the excess premium radio button is either 'N' or 'Y'.  
 - The value of the mobile radio button is either 'N' or 'Y'.  
 - The product ID is not 'Y' or 'P'.  
2. Alternatively, the "Proceed" button should be enabled and navigable if:  
 - The values of the radio buttons for dispatch, receipt, premium, and rider are either 'N' or 'Y'.  
 - The value of the "Proceed" button control is 'X' or the value of the excess premium radio button is either 'N' or 'Y'.  
 - The value of the mobile radio button is either 'N' or 'Y'.  
 - The product ID is 'Y' or 'P'.  
 - The value of the funds radio button is 'Y' or null.  
3. If the above conditions are not met, the "Proceed" button should be disabled and non-navigable.  
4. When the mobile radio button is changed to 'Y', the "Proceed" button should be enabled, navigable, and the focus should move to it.  
5. If the mobile radio button is not 'Y', the system should clear the current block, set specific control values, navigate to the "Insured Person" tab, and hide the alert window.  
  
Definition of Done:  
- The "Proceed" button's state (enabled/disabled) and navigability should be correctly updated based on the specified conditions.  
- The system should handle the change in the mobile radio button value as described.  
- The system should navigate and update control values as specified when the mobile radio button is not 'Y'.  
- All functionalities should be tested and verified to ensure they work as expected without any Oracle Forms-specific terminology or dependencies.

# Validate Backdation Option and Enable/Disable Proceed Button

Type: BLOCK\_ALERT

Title: Validate Backdation Option and Enable/Disable Proceed Button  
  
Acceptance Criteria:  
1. The system should validate the "Backdation" option along with other related options such as "Dispatch", "Receipt", "Premium", "Rider", "Excess Premium", and "Mobile".  
2. If all the options are either 'N' or 'Y' and the product ID is not 'Y' or 'P', or if certain conditions involving the product ID and funds are met, the "Proceed" button should be enabled and navigable.  
3. If the conditions are not met, the "Proceed" button should be disabled and non-navigable.  
4. When the "Backdation" option is changed, the system should check the policy's inception date against a specific date and perform database operations such as deleting and inserting records in the "azbj\_chk\_backdation" table.  
5. If the inception date is earlier than the specific date, a warning message should be displayed, and the user should be redirected to update the date of commencement.  
  
Definition of Done:  
- The "Proceed" button's state (enabled/disabled) is correctly updated based on the specified conditions.  
- The system performs the necessary database operations when the "Backdation" option is changed.  
- Appropriate warning messages are displayed when required, and the user is redirected to the correct screen for updates.  
- All functionalities are tested and verified to work as expected.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- DELETE FROM azbj\_chk\_backdation WHERE policy\_ref = :control.cn\_policy\_ref;  
- INSERT INTO azbj\_chk\_backdation (policy\_ref, save\_flag, user\_id, time\_stamp) VALUES (:control.cn\_policy\_ref, 'Y', USER, TRUNC(SYSDATE));  
- INSERT INTO azbj\_chk\_backdation (policy\_ref, save\_flag, user\_id, time\_stamp) VALUES (:control.cn\_policy\_ref, t\_flag, USER, TRUNC(SYSDATE));

# Enable Proceed Button Based on Radio Button Selections and Product Types

Type: BLOCK\_ALERT

Title: Enable Proceed Button Based on Radio Button Selections and Product Types  
  
Acceptance Criteria:  
1. The "Proceed" button should be enabled and navigable if the following conditions are met:  
 - The values of the radio buttons for backdating, dispatch, receipt, premium, rider, and mobile are either 'N' or 'Y'.  
 - The value of the "Proceed" button control is 'X' or the value of the "Excess Premium" radio button is 'N' or 'Y'.  
 - The product ID does not belong to the 'Y' or 'P' categories.  
 - Alternatively, if the product ID belongs to the 'Y' or 'P' categories and the "Funds" radio button is 'Y' or null, the "Proceed" button should also be enabled and navigable.  
2. If the above conditions are not met, the "Proceed" button should be disabled and non-navigable.  
3. When the "Excess Premium" radio button is changed to 'Y', the system should navigate to the "Funds" radio button if the product ID belongs to the 'Y' or 'P' categories; otherwise, it should navigate to the "Mobile" radio button.  
4. If the "Excess Premium" radio button is not 'Y', the system should clear the current block, set specific control values, and navigate to the "Insured Person" tab, hiding the alert window.  
  
Definition of Done:  
- The "Proceed" button's state (enabled/disabled and navigable/non-navigable) is correctly updated based on the specified conditions.  
- The system correctly navigates to the appropriate radio button or block based on the "Excess Premium" radio button's value.  
- The alert window is hidden, and the "Insured Person" tab is displayed when the "Excess Premium" radio button is not 'Y'.  
- All conditions and navigations are thoroughly tested and verified to ensure they work as expected.

# Financial Eligibility and Premium Payment Capacity Input and Display

Type: FINANCIAL\_UW\_ELIGI

Title: Financial Eligibility and Premium Payment Capacity Input and Display  
  
Acceptance Criteria:  
1. The system should display fields for financial eligibility and premium payment capacity for both primary and secondary entities.  
2. The fields should include:  
 - Financial Eligibility (Primary and Secondary)  
 - Premium Payment Capacity (Primary and Secondary)  
 - Annual Premium  
 - Total Premium Liability (Annual)  
 - Income Covers  
 - Declined Lives  
 - Rated Up  
3. The fields should be formatted correctly, with appropriate justification and color coding.  
4. The system should allow navigation between fields using the keyboard.  
5. The system should display additional information such as female category for both primary and secondary entities.  
  
Definition of Done:  
1. All specified fields are displayed and functional.  
2. Fields are correctly formatted and color-coded.  
3. Navigation between fields using the keyboard is implemented.  
4. Additional information such as female category is displayed.  
5. The feature is tested and verified to meet the acceptance criteria.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Not applicable as the provided XML content does not include any direct database operations or queries.

# Validate Income Covers Field

Type: FINANCIAL\_UW\_ELIGI

Title: Validate Income Covers Field  
  
Acceptance Criteria:  
1. When a value is entered in the "Income Covers" field, it should be validated to ensure it is numeric.  
2. If the value is not numeric, an error message should be displayed: "Please Enter valid Numeric Value".  
3. Upon successful validation, the focus should move to the next field, "PPC Eligibility".  
  
Definition of Done:  
1. The "Income Covers" field accepts only numeric values.  
2. An appropriate error message is displayed for invalid entries.  
3. The focus moves to the "PPC Eligibility" field after successful validation.  
  
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.

# Validate Financial Eligibility Phone Number Input

Type: FINANCIAL\_UW\_ELIGI

Title: Validate Financial Eligibility Phone Number Input  
  
Acceptance Criteria:  
1. When the user enters a value in the financial eligibility phone number field, the system should check if the value is numeric.  
2. If the value is not numeric, the system should display an error message: "Please Enter valid Numeric Value".  
3. The system should prevent navigation to the next field if the entered value is not numeric.  
  
Definition of Done:  
1. The financial eligibility phone number field should validate the input to ensure it is numeric.  
2. An error message should be displayed if the input is not numeric.  
3. The user should not be able to navigate to the next field if the input is invalid.  
4. The functionality should be tested and verified to work as expected.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Not applicable as the provided XML content does not include any direct database operations.

# Validate Annual Premium PH Field

Type: FINANCIAL\_UW\_ELIGI

Title: Validate Annual Premium PH Field  
  
Acceptance Criteria:  
1. When a value is entered in the "Annual Premium PH" field, it should be validated to ensure it is numeric.  
2. If the value is not numeric, an error message should be displayed: "Please Enter valid Numeric Value".  
3. Upon successful validation, the focus should move to the next field, "PPC\_ELIG\_IP".  
  
Definition of Done:  
1. The "Annual Premium PH" field accepts only numeric values.  
2. An appropriate error message is displayed for invalid entries.  
3. The focus moves to the "PPC\_ELIG\_IP" field after successful validation.  
  
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.

# Validate Financial Eligibility Input Field

Type: FINANCIAL\_UW\_ELIGI

Title: Validate Financial Eligibility Input Field  
  
Acceptance Criteria:  
1. When a user enters a value in the "Financial Eligibility" input field, the system should validate that the input is numeric.  
2. If the input is not numeric, an error message should be displayed: "Please Enter valid Numeric Value".  
3. Upon successful validation, the user should be able to navigate to the next input field seamlessly.  
  
Definition of Done:  
1. The "Financial Eligibility" input field accepts only numeric values.  
2. An appropriate error message is displayed for non-numeric inputs.  
3. The user can navigate to the next input field without any issues after entering a valid numeric value.  
4. All validations and error messages are tested and confirmed to be working as expected.  
  
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.

# Validate Numeric Input for Rated Up Field

Type: FINANCIAL\_UW\_ELIGI

Detailed description: As a user, I want to ensure that the "Rated Up" field in the financial eligibility form only accepts valid numeric values so that data integrity is maintained.  
  
Acceptance criteria:  
1. When a user enters a value in the "Rated Up" field, the system should validate that the input is numeric.  
2. If the input is not numeric, the system should display an error message: "Please Enter valid Numeric Value".  
3. Upon successful validation, the system should allow the user to navigate to the next field, "PPC Eligibility".  
  
Definition of Done:  
1. The "Rated Up" field accepts only numeric values.  
2. An error message is displayed for non-numeric inputs.  
3. The user can navigate to the "PPC Eligibility" field after entering a valid numeric value in the "Rated Up" field.  
4. All validations and navigations are tested and confirmed to be working as expected.  
  
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.

# Validate Income Coverage Field and Navigate to Next Field

Type: FINANCIAL\_UW\_ELIGI

Title: Validate Income Coverage Field and Navigate to Next Field  
  
Acceptance Criteria:  
1. When a value is entered in the income coverage field, it should be validated to ensure it is numeric.  
2. If the value is not numeric, an error message should be displayed: "Please Enter valid Numeric Value".  
3. Upon successful validation, the system should automatically navigate to the next field.  
  
Definition of Done:  
1. The income coverage field accepts only numeric values.  
2. An error message is displayed for non-numeric values.  
3. The system navigates to the next field upon successful validation.  
4. All acceptance criteria are met and tested.  
  
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.

# Validate Numeric Input for Declined Lives Field

Type: FINANCIAL\_UW\_ELIGI

Title: Validate Numeric Input for Declined Lives Field  
  
Acceptance Criteria:  
1. When a user enters a value in the "Declined Lives" field, the system should check if the value is numeric.  
2. If the value is not numeric, the system should display an error message: "Please Enter valid Numeric Value".  
3. When the user navigates away from the "Declined Lives" field, the system should validate the numeric value again and move the cursor to the next field if the value is valid.  
4. If the value is not valid, the system should display the error message and prevent the user from moving to the next field.  
  
Definition of Done:  
- The "Declined Lives" field should only accept numeric values.  
- Error messages should be displayed for invalid entries.  
- The user should be able to navigate to the next field only after entering a valid numeric value.  
- All validations and error messages should be tested and verified.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include any direct database operations.

# Validate and Navigate Total Premium Liability Annual Field

Type: FINANCIAL\_UW\_ELIGI

Detailed description: As a user, I want to ensure that the "Total Premium Liability Annual" field only accepts valid numeric values and navigates to the next field upon validation, so that data integrity is maintained and the user experience is streamlined.  
  
Acceptance criteria:  
1. When a value is entered in the "Total Premium Liability Annual" field, it should be validated to ensure it is a numeric value.  
2. If the value is not numeric, an error message should be displayed: "Please Enter valid Numeric Value".  
3. Upon successful validation, the system should automatically navigate to the next field, "PPC\_ELIG\_IP".  
  
Definition of Done:  
- The "Total Premium Liability Annual" field should only accept numeric values.  
- An error message should be displayed for non-numeric values.  
- The system should navigate to the "PPC\_ELIG\_IP" field after successful validation.  
- All acceptance criteria are met and tested.  
  
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.

# Validate and Navigate Total Premium Liability per Policy Holder

Type: FINANCIAL\_UW\_ELIGI

Detailed description: As a user, I want to ensure that the "Total Premium Liability per Policy Holder" field only accepts valid numeric values and navigates to the next field upon validation, so that data integrity is maintained and the user experience is smooth.  
  
Acceptance criteria:  
1. When a value is entered in the "Total Premium Liability per Policy Holder" field, it should be validated to ensure it is numeric.  
2. If the value is not numeric, an error message should be displayed: "Please Enter valid Numeric Value".  
3. Upon successful validation, the system should automatically navigate to the next field, "PPC Eligibility per Policy Holder".  
  
Definition of Done:  
- The "Total Premium Liability per Policy Holder" field accepts only numeric values.  
- An error message is displayed for non-numeric values.  
- The system navigates to the "PPC Eligibility per Policy Holder" field after successful validation.  
- All acceptance criteria are met and tested.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include any direct database operations.

# Validate PPC Eligibility Field for Numeric Input

Type: FINANCIAL\_UW\_ELIGI

Title: Validate PPC Eligibility Field for Numeric Input  
  
Acceptance Criteria:  
1. When the user enters a value in the "PPC Eligibility" field and moves to the next field, the system should check if the entered value is numeric.  
2. If the entered value is not numeric, the system should display an error message: "Please Enter valid Numeric Value".  
3. The validation should occur both when the user attempts to validate the item and when the user navigates to the next item.  
  
Definition of Done:  
- The "PPC Eligibility" field should only accept numeric values.  
- An error message should be displayed if a non-numeric value is entered.  
- The validation should be triggered both on item validation and on navigation to the next item.  
- The error message should prevent the user from proceeding until a valid numeric value is entered.  
  
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.

# Validate Numeric Input for PPC\_ELIG\_PH Field

Type: FINANCIAL\_UW\_ELIGI

Detailed description: As a user, I want to ensure that the input for the "PPC\_ELIG\_PH" field is validated to be a numeric value so that incorrect data is not entered into the system.  
  
Acceptance criteria:  
1. When the user enters a value in the "PPC\_ELIG\_PH" field and moves to the next item, the system should check if the value is numeric.  
2. If the value is not numeric, the system should display an error message: "Please Enter valid Numeric Value".  
3. The validation should occur both when the user attempts to validate the item and when the user navigates to the next item.  
  
Definition of Done:  
1. The "PPC\_ELIG\_PH" field should only accept numeric values.  
2. An error message should be displayed if a non-numeric value is entered.  
3. The validation logic should be triggered both on item validation and on navigation to the next item.  
4. The error message should prevent the user from proceeding until a valid numeric value is entered.  
  
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.

# Validate and Navigate 'Annual Premium' Field in Financial Eligibility Form

Type: FINANCIAL\_UW\_ELIGI

Description:   
As a user, I want to ensure that the "Annual Premium" field in the financial eligibility form only accepts valid numeric values so that data integrity is maintained and the form navigation is seamless.  
  
Acceptance Criteria:  
1. When a value is entered into the "Annual Premium" field, it should be validated to ensure it is numeric.  
2. If the entered value is not numeric, an error message should be displayed: "Please Enter valid Numeric Value".  
3. Upon successful validation, the cursor should move to the "PPC Eligibility" field in the form.  
  
Definition of Done:  
1. The "Annual Premium" field accepts only numeric values.  
2. An appropriate error message is displayed for invalid entries.  
3. The cursor navigates to the "PPC Eligibility" field after successful validation.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Not applicable as the provided XML content does not include any direct database operations.

# Validate Numeric Input for Rated Up Percentage

Type: FINANCIAL\_UW\_ELIGI

Title: Validate Numeric Input for Rated Up Percentage  
  
Acceptance Criteria:  
1. When a user enters a value in the "Rated Up Percentage" field, the system should validate that the input is numeric.  
2. If the input is not numeric, the system should display an error message: "Please Enter valid Numeric Value".  
3. Upon successful validation, the system should allow the user to navigate to the next field, "PPC Eligibility Percentage".  
  
Definition of Done:  
1. The "Rated Up Percentage" field accepts only numeric values.  
2. An error message is displayed if the input is not numeric.  
3. The user can navigate to the "PPC Eligibility Percentage" field after entering a valid numeric value.  
  
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.

# Validate Numeric Input for Declined Life PH Field

Type: FINANCIAL\_UW\_ELIGI

Title: Validate Numeric Input for Declined Life PH Field  
  
Acceptance Criteria:  
1. When a value is entered into the "Declined Life PH" field, it should be checked to ensure it is numeric.  
2. If the value is not numeric, an error message should be displayed stating, "Please Enter valid Numeric Value".  
3. When navigating to the next item, the system should validate the current input and move to the next field if the input is valid.  
4. If the input is invalid, the system should display an error message and prevent navigation to the next field.  
  
Definition of Done:  
- The input validation for the "Declined Life PH" field is implemented and tested.  
- Error messages are displayed correctly when invalid input is entered.  
- Navigation to the next field works as expected when the input is valid.  
- The feature is tested and verified to ensure it meets the acceptance criteria.

# Manage Liquid Investments

Type: LIQUID\_INVESTMENT

Title: Manage Liquid Investments  
  
Acceptance Criteria:  
1. The system should allow the user to input values for the following types of liquid investments:  
 - Fixed-term deposits  
 - Mutual funds  
 - Equity shares  
 - Fund values of UL policies  
 - Bank balances  
 - One-time incomes  
 - Other liquid investments  
2. The system should automatically calculate the total investment by summing up all the individual investment values.  
3. The total investment should be displayed in a specific format (e.g., 99,99,99,99,999).  
  
Definition of Done:  
1. The user can successfully input values for each type of liquid investment.  
2. The total investment is correctly calculated and displayed in the specified format.  
3. Navigation between input fields is intuitive and allows for easy data entry.  
4. The user interface is user-friendly and visually 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.

# Manage Financial Underwriting Details

Type: FINANCIAL\_UW

Detailed description: As a user, I want to manage financial underwriting details, including financial years, proof types, income details, deductions, and tax information, so that I can accurately calculate total income and net profit.  
  
Acceptance criteria:  
1. The system should automatically populate the financial year based on the start and end dates from the system constants.  
2. If the financial year is not provided or matches the previous financial year, it should be set to the calculated value.  
3. If the user is a premium payer and the gross income is not provided, it should be populated from the insured person's or policy holder's annual income, and the proof type should be set to 'PF'.  
4. The total income should be calculated as the sum of gross income, exempted income, and one-time income.  
5. The net profit should be calculated as the total income minus the sum of deductions and tax.  
6. The system should display average values for ITR, computation, total income, and net profit.  
7. The system should handle navigation and record instance events, ensuring proper data population and validation.  
  
Definition of Done:  
- The financial underwriting details are correctly managed and calculated.  
- The system automatically populates and calculates values as per the acceptance criteria.  
- The user can navigate through records without errors.  
- All calculations and data population are validated and tested.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- The query to fetch start and end dates from the system constants table:  
 ```sql  
 SELECT TO\_NUMBER(TO\_CHAR(START\_DATE, 'YYYY')), TO\_NUMBER(TO\_CHAR(END\_DATE, 'YY'))  
 INTO v\_start, v\_end  
 FROM azbj\_system\_constants  
 WHERE SYS\_type LIKE 'FIN\_YEAR';  
 ```

# Implement Proof Type Dropdown in Financial Underwriting Section

Type: FINANCIAL\_UW

Title: Implement Proof Type Dropdown in Financial Underwriting Section  
  
Acceptance Criteria:  
1. The proof type should be selectable from a dropdown list.  
2. The dropdown list should be positioned appropriately on the screen.  
3. The proof type field should be clearly labeled as "Proof".  
4. The proof type field should be located within the financial underwriting section of the form.  
5. The proof type field should be displayed on the "New Business" tab.  
6. The proof type field should have a white background and be styled with a specific font and size for the label.  
  
Definition of Done:  
1. The proof type dropdown list is implemented and functional.  
2. The proof type field is correctly positioned and labeled on the form.  
3. The proof type field is part of the financial underwriting section.  
4. The proof type field is displayed on the "New Business" tab.  
5. The proof type field adheres to the specified styling requirements.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include any specific database queries.

# Manage Financial Information

Type: CFT\_FIN

Title: Manage Financial Information  
  
Acceptance Criteria:  
1. The system should allow the user to input the owner of the financial data.  
2. The system should allow the user to input the number of shares.  
3. The system should automatically calculate and display the total sum of shares based on the inputted number of shares.  
  
Definition of Done:  
1. The user can successfully input the owner of the financial data.  
2. The user can successfully input the number of shares.  
3. The system automatically calculates and displays the total sum of shares.  
4. The user interface is intuitive and user-friendly.  
5. The functionality is tested and verified to ensure accuracy and reliability.  
  
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 Spouse Financial Details

Type: SPOUSE\_FIN\_DTLS

Title: Manage Spouse Financial Details  
  
Acceptance Criteria:  
1. The system should allow the user to input the following financial details for a spouse:  
 - Proof Type  
 - Gross Income  
 - Exempted Income  
 - One-Time Income  
 - Deductions  
 - Tax  
2. The system should automatically calculate the Total Income as the sum of Gross Income, Exempted Income, and One-Time Income.  
3. The system should automatically calculate the Net Income as the Total Income minus the sum of Deductions and Tax.  
4. The system should display the calculated Total Income and Net Income to the user.  
  
Definition of Done:  
1. The user can input and save the financial details of a spouse.  
2. The system correctly calculates and displays the Total Income and Net Income based on the provided inputs.  
3. The user interface is intuitive and user-friendly.  
4. All calculations are accurate and reflect the correct financial status of the spouse.  
5. The feature is tested and verified to ensure it meets the acceptance criteria.  
  
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.

# Select Proof Type for Spouse Financial Details

Type: SPOUSE\_FIN\_DTLS

Title: Select Proof Type for Spouse Financial Details  
  
Acceptance Criteria:  
1. The proof type field should display a list of predefined proof types.  
2. The list should be easily accessible and selectable by the user.  
3. The selected proof type should be saved and associated with the spouse financial details.  
  
Definition of Done:  
1. The proof type field is implemented and displays a list of predefined proof types.  
2. Users can select a proof type from the list.  
3. The selected proof type is saved correctly and associated with the spouse financial details.  
4. 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 XML content does not include any specific database queries.

# Implement Proof Type Selection for Spouse Financial Details

Type: SPOUSE\_FIN\_DTLS

Title: Implement Proof Type Selection for Spouse Financial Details  
  
Acceptance Criteria:  
1. The proof type field should display a list of predefined proof types.  
2. The list should be easily accessible and selectable by the user.  
3. The selected proof type should be saved and associated with the spouse financial details.  
  
Definition of Done:  
1. The proof type field is implemented and displays a list of predefined proof types.  
2. Users can select a proof type from the list.  
3. The selected proof type is saved correctly and associated with the spouse financial details.  
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.

# Automatic Calculation of Spouse's Net Profit

Type: SPOUSE\_FIN\_DTLS

Title: Automatic Calculation of Spouse's Net Profit  
  
Acceptance Criteria:  
1. When the premium payer is identified as the spouse (value 3), the system should calculate the spouse's net profit by subtracting the spouse's net income from the average net profit.  
2. The calculated net profit should be stored in the appropriate field for further processing.  
  
Definition of Done:  
1. The system correctly identifies when the premium payer is the spouse.  
2. The system accurately calculates the spouse's net profit based on the given formula.  
3. The calculated net profit is stored and visible in the appropriate field.  
4. The functionality is tested and verified to ensure accuracy.  
  
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.

# View and Manage Doctor Details in Mobile App Interface

Type: BLK\_MOB\_APP

Detailed description: As a user, I want to view and manage doctor details and related information within a mobile application interface, so that I can efficiently access and update relevant data.  
  
Acceptance criteria:  
1. The interface should display the following fields:  
 - Doctor Code (editable text field)  
 - Partner Name (editable text field with a maximum length of 200 characters)  
 - Age of Client (display-only field)  
 - Gender (display-only field)  
 - Test Conducted (display-only field)  
 - Time Stamp (display-only field with datetime format)  
 - Longitude (editable text field with a maximum length of 100 characters)  
 - Latitude (editable text field with a maximum length of 100 characters)  
2. The interface should include the following buttons:  
 - Exit (to close the interface)  
 - View Images (to view related images)  
 - View Location (to view the location on a map)  
3. The interface should be user-friendly and visually organized, with appropriate labels and prompts for each field.  
4. The interface should be displayed within a window titled "Mobile Apps Doctor Details" and should not allow resizing, minimizing, or maximizing.  
  
Definition of Done:  
- The interface is implemented and displays all specified fields and buttons.  
- The fields and buttons function as described in the acceptance criteria.  
- The interface is visually organized and user-friendly.  
- The window title is "Mobile Apps Doctor Details" and it adheres to the specified window properties.  
  
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 Location Button Functionality

Type: BLK\_MOB\_APP

Title: View Location Button Functionality  
  
Acceptance Criteria:  
1. When the "View Location" button is pressed, the system should construct a URL using the longitude and latitude values of the mobile app entry.  
2. The constructed URL should be in the format: `http://maps.google.com/?q=<longitude>,<latitude>`.  
3. If the URL is successfully constructed, the system should open the URL in a web browser to display the location on Google Maps.  
  
Definition of Done:  
- The "View Location" button is functional and correctly constructs the URL using the longitude and latitude values.  
- The URL opens in a web browser and displays the correct location on Google Maps.  
- The feature is tested and verified to work 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 CRUD operations.

# Exit Button Functionality

Type: BLK\_MOB\_APP

Title: Exit Button Functionality  
  
Acceptance Criteria:  
1. When the "Exit" button is pressed, the system should navigate to the "Mobile App Information" screen.  
2. The "Exit" button should be clearly labeled and easily identifiable.  
3. The button should be positioned at the specified location on the screen and have the specified visual properties (e.g., color, font).  
  
Definition of Done:  
1. The "Exit" button is implemented and functional.  
2. Pressing the "Exit" button successfully navigates the user to the "Mobile App Information" screen.  
3. The button meets the specified design and usability criteria.  
4. The feature is tested and verified to work as expected.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include any database queries.

# View Images Button Functionality

Type: BLK\_MOB\_APP

Detailed description: As a user, I want to be able to view images related to a specific insured person by clicking a button, so that I can easily access and review the relevant images.  
  
Acceptance criteria:  
1. When the user clicks the "View Images" button, the system should generate a URL using the insured person's verification number or sign card number.  
2. The URL should be encrypted for security purposes.  
3. If the URL is successfully generated, the system should open the URL in a web browser to display the images.  
4. If the URL is not generated, no action should be taken.  
  
Definition of Done:  
- The "View Images" button is present on the user interface.  
- Clicking the button triggers the URL generation process.  
- The URL is encrypted and includes the insured person's verification number or sign card number.  
- The system successfully opens the URL in a web browser to display the images if the URL is generated.  
- No action is taken if the URL is not generated.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided logic does not include direct database CRUD operations.

# User selects a valid doctor code to display associated client names

Type: BLK\_MOB\_APP

Title: User selects a valid doctor code to display associated client names  
  
Acceptance Criteria:  
1. When a valid doctor code is selected, the system should:  
 - Clear the list of partner names.  
 - Reset the fields for gender, age of client, test, and timestamp to null.  
 - Populate the partner name list with client names associated with the selected doctor code and application number.  
 - Display a warning message if the doctor code is invalid.  
  
2. The system should fetch client names from the database where the user ID matches the selected doctor code and the application number matches either the verification number or the sign card number of the insured person.  
  
Definition of Done:  
- The user can select a doctor code and see the corresponding client names in the partner name list.  
- The fields for gender, age of client, test, and timestamp are reset when a new doctor code is selected.  
- A warning message is displayed if an invalid doctor code is selected.  
- The functionality is tested and verified to work as expected.  
  
DB queries for Table reference CRUD operations only(With Usage):  
```sql  
SELECT CLIENT\_NAME, ROWNUM sr\_no  
FROM (  
 SELECT DISTINCT CLIENT\_NAME  
 FROM AZBJ\_DOCT\_CUST\_DETAILS  
 WHERE USER\_ID = :BLK\_MOB\_APP.DOCTOR\_CODE  
 AND application\_no = TO\_CHAR(NVL(:insured\_person.ip\_verf\_no, :insured\_person.ip\_sign\_card\_no))  
);  
```

# Retrieve Partner Details Based on Partner Name

Type: BLK\_MOB\_APP

Title: Retrieve Partner Details Based on Partner Name  
  
Acceptance Criteria:  
1. When the user inputs a partner's name, the system should clear any existing values for gender, age, test number, and timestamp.  
2. If the partner's name is not empty, the system should fetch the gender, age, test number, and timestamp from the database using the provided partner name, doctor code, and application number.  
3. If the database query fails or no matching records are found, the system should clear the values for gender, age, test number, and timestamp.  
  
Definition of Done:  
- The user can input a partner's name in the designated field.  
- The system automatically retrieves and displays the partner's gender, age, test number, and timestamp if the partner's name is valid and matches the database records.  
- The system handles cases where no matching records are found by clearing the relevant fields.  
- The feature is tested and verified to work as expected.  
  
DB queries for Table reference CRUD operations only(With Usage):  
```sql  
SELECT GENDER,  
 AGE,  
 TEST\_NO,  
 UPD\_DATE  
 INTO :BLK\_MOB\_APP.GENDER,  
 :BLK\_MOB\_APP.AGE\_OF\_CLIENT,  
 :BLK\_MOB\_APP.TEST,  
 :BLK\_MOB\_APP.TIMESTAMP  
 FROM AZBJ\_DOCT\_CUST\_DETAILS  
 WHERE USER\_ID = :BLK\_MOB\_APP.DOCTOR\_CODE  
 AND CLIENT\_NAME = :BLK\_MOB\_APP.PARTNER\_NAME  
 AND application\_no = TO\_CHAR(NVL(:insured\_person.ip\_verf\_no, :insured\_person.ip\_sign\_card\_no));  
```  
  
This query is used to fetch the gender, age, test number, and timestamp based on the provided partner name, doctor code, and application number.

# Display Insurance Coverage Details

Type: BLK\_COVER

Detailed description: As a user, I want to view the details of insurance coverage in a structured format so that I can easily understand the key attributes of each coverage.  
  
Acceptance criteria:  
1. The system should display the following fields for each insurance coverage:  
 - Cover Code  
 - Rider Name  
 - Paying Term  
 - Sum Assured  
 - Benefit Term  
2. The fields should be displayed in a read-only format, except for the "Benefit Term" which should be editable.  
3. The fields should be aligned and formatted consistently for better readability.  
4. The "Sum Assured" and "Paying Term" fields should be right-justified and should only accept numeric values.  
5. The "Sum Assured" field should not accept values less than zero.  
  
Definition of Done:  
- The user interface displays the insurance coverage details as specified.  
- All fields, except "Benefit Term," are read-only.  
- The "Benefit Term" field is editable.  
- Numeric fields are right-justified and validated for numeric input.  
- The "Sum Assured" field does not accept negative values.  
- The layout is consistent and user-friendly.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- No specific database queries are provided in the XML content.

# Display and Manage PAN, Name, Status, Partner, and Document Type Information

Type: CTRL

Title: Display and Manage PAN, Name, Status, Partner, and Document Type Information  
  
Acceptance Criteria:  
1. The form should display the following fields:  
 - PAN number  
 - Name  
 - Status  
 - Partner (as a list with three options)  
 - Document type  
2. All fields should be read-only and not allow updates or insertions.  
3. The form should be organized within a tabbed interface.  
4. The form should have a scrollbar for navigation if the content exceeds the display area.  
  
Definition of Done:  
- The form is implemented and displays the required fields.  
- All fields are read-only.  
- The form is organized within a tabbed interface.  
- A scrollbar is present and functional for navigation.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include any database-specific queries or operations.

# Partner Selection Field on New Business Form

Type: CTRL

Title: Partner Selection Field on New Business Form  
  
Acceptance Criteria:  
1. The partner selection field should be displayed on the "New Business" form.  
2. The partner selection field should be a dropdown list containing three predefined options.  
3. The partner selection field should be read-only, meaning users cannot insert or update the list items.  
4. The partner selection field should be positioned at the specified coordinates on the form.  
5. The form should have a tabbed interface with the partner selection field located on the "AML" tab.  
6. The form should have a consistent visual style, with the partner selection field having a white background and the tabbed interface having a gray background.  
  
Definition of Done:  
- The partner selection field is visible and correctly positioned on the "New Business" form.  
- The dropdown list contains exactly three predefined options.  
- Users are unable to modify the contents of the partner selection field.  
- The form's visual attributes match the specified design requirements.  
- The form is tested and verified to ensure the partner selection field functions as expected.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include any database-specific queries or operations.

# Manage Surrogate Proof Details

Type: SIP\_CONTROL

Title: Manage Surrogate Proof Details  
  
Acceptance Criteria:  
1. The system should display fields for surrogate proof type, proof description, field value, document date, derived income, and derived Tasa value.  
2. The derived income and derived Tasa value fields should be read-only and not allow user input.  
3. The fields should be displayed in a user-friendly manner with appropriate labels and alignment.  
4. The system should ensure that the data entered in these fields is stored correctly in the database.  
  
Definition of Done:  
1. The user interface displays all required fields with appropriate labels.  
2. The derived income and derived Tasa value fields are read-only.  
3. Data entered in the fields is saved correctly in the database.  
4. The user interface is tested for usability and correctness.  
5. All acceptance criteria are met and verified through testing.  
  
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 Proof Type Dropdown

Type: SIP\_CONTROL

Title: Implement Proof Type Dropdown  
  
Acceptance Criteria:  
1. The proof type field should display a list of available proof types for selection.  
2. The list should be displayed in a dropdown format.  
3. The proof type field should be clearly labeled as "Surrogate Proof Type".  
4. The field should be positioned appropriately within the form for easy access.  
5. The form should have a consistent look and feel, with appropriate font styles and colors for readability.  
  
Definition of Done:  
1. The proof type field is implemented and displays a dropdown list of available proof types.  
2. The field is labeled "Surrogate Proof Type" and is positioned correctly within the form.  
3. The form's appearance, including font styles and colors, is consistent and user-friendly.  
4. The functionality is tested and verified to ensure the correct proof type can be selected and recorded.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include specific database queries.

# Validate and Calculate Entry Age for Insured Person and Policyholder

Type: SIP\_CONTROL

Title: Validate and Calculate Entry Age for Insured Person and Policyholder  
  
Acceptance Criteria:  
1. When the user enters a value in the field, the system should:  
 - Calculate the entry age of the insured person using their date of birth and the inception date or effective date.  
 - Calculate the entry age of the policyholder using their date of birth and the inception date or effective date.  
 - Determine the final age to be used based on the calculated ages:  
 - If the insured person's entry age is less than 18, use the policyholder's entry age.  
 - Otherwise, use the insured person's entry age.  
 - Use the determined age, proof type, and field value for further processing.  
  
2. When the user navigates to the next item, the system should:  
 - Perform the same age calculations and determinations as described above.  
 - Proceed to the next record.  
  
Definition of Done:  
- The system correctly calculates and determines the age based on the provided logic.  
- The system uses the determined age, proof type, and field value for further processing.  
- The system handles exceptions and displays appropriate messages if any errors occur during the process.  
- The system allows navigation to the next record after processing the current field value.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided logic does not include direct CRUD operations on the database.

# Manage Child Cover Details

Type: BLK\_CHILD\_COVERS

Detailed description: As a user, I want to manage child cover details within a dedicated section so that I can add, view, and delete child cover information efficiently.  
  
Acceptance criteria:  
1. The section should display the following fields for each child cover:  
 - Child Name  
 - Child Date of Birth (DOB)  
 - Child Age  
 - Percentage of Sum Assured  
 - A checkbox to mark for deletion  
2. The section should include buttons for the following actions:  
 - Add Child: Allows the user to add a new child cover.  
 - Delete: Allows the user to delete a selected child cover.  
 - Exit: Allows the user to exit the section.  
3. The Child Age field should be read-only and automatically calculated based on the Child DOB.  
4. The section should be visually organized and user-friendly, with appropriate labels and prompts for each field.  
  
Definition of Done:  
- The section is implemented and accessible within the application.  
- All fields and buttons are functional and meet the acceptance criteria.  
- The Child Age field is correctly calculated and displayed as read-only.  
- The user interface is intuitive and visually appealing.  
- The section 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 specific database queries.

# Add Child Record Functionality

Type: BLK\_CHILD\_COVERS

Detailed description: As a user, I want to add a new child record to the child covers section so that I can manage and update child cover details efficiently.  
  
Acceptance criteria:  
1. When the "Add Child" button is pressed, the system should navigate to the child covers section.  
2. The system should then move to the last record in the child covers section.  
3. A new record should be created in the child covers section for the user to input new child details.  
  
Definition of Done:  
- The "Add Child" button should be functional and trigger the described actions.  
- The system should correctly navigate to the child covers section and create a new record.  
- The user should be able to input new child details in the newly created record.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include specific database queries.

# Automatic Child Age Calculation Based on DOB

Type: BLK\_CHILD\_COVERS

Title: Automatic Child Age Calculation Based on DOB  
  
Acceptance Criteria:  
1. When the user enters a valid date of birth (DOB) for the child, the system should automatically calculate the child's age.  
2. The age calculation should be based on the entered DOB and the current date.  
3. If the DOB is not entered, the age should not be calculated.  
4. The calculated age should be displayed in the appropriate field.  
5. If any error occurs during the age calculation, the age field should be cleared.  
  
Definition of Done:  
- The user can enter the child's DOB in the designated field.  
- The system calculates and displays the child's age based on the entered DOB and the current date.  
- The age field is cleared if an error occurs during the calculation.  
- The functionality is tested and verified to ensure accuracy and reliability.  
  
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.

# Exit Button Functionality in Nominee/Child Cover Details

Type: BLK\_CHILD\_COVERS

User Story:  
As a user, I want to be able to exit the "Nominee/Child Cover Details" section easily, so that I can return to the main "Child Covers" section without any hassle.  
  
Acceptance Criteria:  
1. When the "Exit" button is pressed, the system should refresh the child covers data.  
2. The "Nominee/Child Cover Details" view should be hidden.  
3. The "Nominee/Child Cover Details" window should be hidden.  
4. The focus should be set back to the main "Child Covers" section.  
  
Definition of Done:  
- The "Exit" button is functional and performs the required actions.  
- The "Nominee/Child Cover Details" view and window are hidden upon pressing the "Exit" button.  
- The focus is correctly set back to the main "Child Covers" section.  
- All actions are performed without any errors or issues.  
  
Block Name: BLK\_CHILD\_COVERS

# Delete Multiple Child Cover Records

Type: BLK\_CHILD\_COVERS

Title: Delete Multiple Child Cover Records  
  
Acceptance Criteria:  
1. When the delete button is pressed, the system should navigate to the child cover records section.  
2. The system should count the total number of child cover records.  
3. The system should iterate through each child cover record.  
4. If a record is marked for deletion (indicated by a specific field being set to 'Y'), the system should delete that record.  
5. The system should continue to the next record until all records have been checked.  
6. The system should return to the first record after the deletion process is complete.  
  
Definition of Done:  
- The delete button functionality is implemented and tested.  
- The system correctly navigates through the child cover records.  
- Records marked for deletion are successfully deleted.  
- The system returns to the first record after processing all records.  
- 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 logic is specific to Oracle Forms and cannot be executed directly in the database without modification.

# Display and Interaction of Child Age Field

Type: BLK\_CHILD\_COVERS

Title: Display and Interaction of Child Age Field  
  
Acceptance Criteria:  
1. The "Child Age" field should be displayed with a label "Age" in the "Nominee/Child Cover Details" section.  
2. The field should be non-editable, meaning users cannot insert or update the value.  
3. The field should be positioned at coordinates (220, 56) within the section.  
4. The field should have a gray background and black text.  
5. The font for the field and its prompt should be Tahoma, with a size of 10 and a medium weight.  
6. When the user attempts to navigate away from the "Child Age" field, the focus should automatically move to the "Child Sum Percentage" field.  
  
Definition of Done:  
- The "Child Age" field is displayed correctly with the specified properties.  
- The field is non-editable.  
- The navigation from the "Child Age" field to the "Child Sum Percentage" field works as specified.  
- The user interface matches the design specifications in terms of positioning, colors, and fonts.  
  
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 IIB Details section

Type: BLK\_IIB\_CTRL

Title: View and interact with IIB Details section  
  
Acceptance Criteria:  
1. The IIB Details section should display a policy number field that is not editable.  
2. There should be a "Back" button that allows the user to navigate back to the previous screen.  
3. The layout should be user-friendly, with the policy number and back button positioned appropriately for easy access.  
  
Definition of Done:  
1. The policy number field is visible and non-editable.  
2. The "Back" button is functional and returns the user to the previous screen.  
3. The user interface is tested for usability and accessibility.  
4. All functionalities are verified to work as expected without any Oracle Forms-specific dependencies.  
  
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.

# Navigate Back from Current Screen

Type: BLK\_IIB\_CTRL

Title: Navigate Back from Current Screen  
  
Acceptance Criteria:  
- When the "Back" button is pressed, the system should navigate to the item identified as 'AG\_AGENT\_CODE'.  
- The current view identified as 'CAN\_IIB\_DET' should be hidden upon pressing the "Back" button.  
  
Definition of Done:  
- The "Back" button is visible and functional on the screen.  
- Pressing the "Back" button successfully navigates to the 'AG\_AGENT\_CODE' item.  
- The 'CAN\_IIB\_DET' view is hidden when the "Back" button is pressed.  
  
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 Insurance Policy Details

Type: BLK\_IIB\_DET

Title: Manage Insurance Policy Details  
  
Acceptance Criteria:  
1. The system should display the following fields for each policy:  
 - Transaction Id  
 - Quest Policy Date  
 - Not Match  
 - Status  
 - Error Description  
 - Remarks  
 - Update Date  
 - Inserted Date  
 - Quest Policy Status  
 - Quest Cause Of Death  
 - Quest Company Number  
 - Input Proposal Policy No  
 - Questdbno  
 - Input Matching Parameter  
 - Quest Dop Doc  
 - Quest Sum Assured  
 - Quest Policy Status  
 - Quest Date Of Exit  
 - Quest Date Of Death  
 - Quest Record Last Updated  
 - Quest Cause Of Death  
 - Quest Reason For Declined  
 - Quest Entity Caution Status  
 - Quest Interme Caution Status  
 - Quest Company Number  
 - Is Negative Match  
  
2. Each field should be enabled and editable, allowing users to input or update information as needed.  
  
3. The system should support scrolling through records vertically, displaying up to 5 records at a time.  
  
4. The system should allow navigation between records using a "Change Record" style.  
  
Definition of Done:  
- The user interface displays all the specified fields for each policy.  
- Users can input and update information in each field.  
- The system supports vertical scrolling and displays up to 5 records at a time.  
- Users can navigate between records using the "Change Record" style.  
- All changes are saved and reflected in the system.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Not applicable as the provided XML content does not include specific database queries or operations.

# View and Manage Previous Underwriting Decisions

Type: PREV\_DET

Title: View and Manage Previous Underwriting Decisions  
  
Acceptance Criteria:  
1. The system should display a list of previous underwriting decisions with the following details:  
 - Serial Number  
 - User ID  
 - Previous Underwriting Decision  
 - Activities  
 - Underwriting Comments  
  
2. The list should be displayed in a grid format with a maximum of 5 records visible at a time and a scrollbar to navigate through additional records.  
  
3. Each field should have the following properties:  
 - Serial Number: Numeric, read-only.  
 - User ID: Text, read-only.  
 - Previous Underwriting Decision: Dropdown list, read-only.  
 - Activities: Text, read-only.  
 - Underwriting Comments: Text, read-only, with a maximum length of 500 characters.  
  
4. The grid should be displayed on a canvas named "Previous Policy Details" within a window named "Previous Policy Window".  
  
Definition of Done:  
- The user can view a grid with the specified fields and properties.  
- The grid allows scrolling to view additional records beyond the first 5.  
- All fields are displayed as read-only.  
- The grid is displayed within the specified canvas and window.  
  
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.

# Display and Select Previous Underwriting Decision

Type: PREV\_DET

Title: Display and Select Previous Underwriting Decision  
  
Acceptance Criteria:  
1. The list of previous underwriting decisions should be displayed in a dropdown menu.  
2. The dropdown menu should be enabled and allow selection.  
3. The selected decision should be displayed in a designated area on the screen.  
4. The dropdown menu should be styled with a specific font and color scheme for readability.  
5. The dropdown menu should be positioned correctly on the screen as per the design specifications.  
  
Definition of Done:  
1. The dropdown menu for previous underwriting decisions is implemented and functional.  
2. The dropdown menu is styled and positioned as per the design specifications.  
3. The dropdown menu allows users to select a previous underwriting decision.  
4. The selected decision is displayed correctly on the screen.  
5. All acceptance criteria are met and tested successfully.

# Manage and View Unit-Linked Funds in Portfolio

Type: SOL\_UL\_AT\_PORTFOL\_2

Title: Manage and View Unit-Linked Funds in Portfolio  
  
Acceptance Criteria:  
1. The system should display a list of unit-linked funds within a portfolio.  
2. Each fund should have a percentage allocation that can be viewed and edited.  
3. The system should automatically calculate and display the total value of all unit-linked funds in the portfolio.  
4. The total value should be a sum of the percentage allocations of all individual funds.  
  
Definition of Done:  
1. The user can view a list of unit-linked funds within a portfolio.  
2. The user can edit the percentage allocation for each fund.  
3. The total value of all unit-linked funds is automatically calculated and displayed.  
4. The functionality is tested and verified to ensure accuracy and reliability.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include specific database queries.

# Manage Fund Information

Type: SOL\_SSO\_FUND

Title: Manage Fund Information  
  
Acceptance Criteria:  
1. The user should be able to select a fund from a predefined list of funds.  
2. The user should be able to enter a percentage allocation for the selected fund.  
3. The system should automatically calculate and display the total fund value based on the entered percentage allocations.  
4. The fund list should be filtered based on specific criteria, such as product ID, date range, and cover code.  
  
Definition of Done:  
1. The fund selection list is populated with valid fund names.  
2. The user can enter and save the percentage allocation for each fund.  
3. The total fund value is correctly calculated and displayed.  
4. The fund list is filtered according to the specified criteria in the database query.  
  
DB queries for Table reference CRUD operations only(With Usage):  
```sql  
SELECT fund\_short\_name, fund\_full\_name   
FROM AZBJ\_TFV\_FUND\_DEFINITION a, azbj\_cover\_funds b   
WHERE a.fund\_short\_name = b.fund\_name  
AND b.product\_id = :control.cn\_product\_id  
AND :susac.sa\_daterecd > NVL(start\_date, '01-apr-2005')  
AND :susac.sa\_daterecd < NVL(end\_date, '01-jan-3000')  
AND a.fund\_id <> 'FVFD000025'  
AND cover\_code = :covers.cv\_cover\_code;  
```

# Validate Cash Plus Pension Fund Apportionment

Type: SOL\_SSO\_FUND

Title: Validate Cash Plus Pension Fund Apportionment  
  
Acceptance Criteria:  
1. When the product ID is one of the following: 31, 32, 33, 34, 49, or 50, and the fund ID is 'NCPPF', the apportionment percentage should not exceed 20%.  
2. If the apportionment percentage exceeds 20% under the specified conditions, an error message should be displayed: "Apportionment for Cash Plus Pension Fund cannot be more than 20".  
3. The form status should be updated to 'Y' after validation.  
  
Definition of Done:  
1. The system correctly identifies the specified product IDs and fund ID.  
2. The system enforces the apportionment percentage limit and displays the appropriate error message when the limit is exceeded.  
3. The form status is updated as required after validation.  
4. The functionality is tested and verified to ensure it meets the acceptance criteria.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include any direct database queries.

# User can select a fund from the list on double-clicking 'Fund Name' field

Type: SOL\_SSO\_FUND

Title: User can select a fund from the list on double-clicking 'Fund Name' field  
  
Acceptance Criteria:  
1. When the user double-clicks on the 'Fund Name' field, the system should check if the global loading status is 'F'.  
2. If the global loading status is 'F', the system should set the form status to 'Y'.  
3. The system should then navigate to the 'covers' section and select the first record.  
4. The system should navigate back to the 'SSO\_FUND' section.  
5. The system should execute a query to fetch the list of funds based on the product ID, date range, and cover code.  
6. The query should exclude the fund with ID 'FVFD000025'.  
7. The system should populate the list of values (LOV) with the results of the query.  
8. The system should display the LOV to the user.  
9. The user should be able to select a fund from the LOV.  
10. The system should navigate to the 'SSO\_FUND.Fundapor' field after the user selects a fund.  
  
Definition of Done:  
- The user can double-click on the 'Fund Name' field to open a list of available funds.  
- The list of funds is filtered based on the product ID, date range, and cover code.  
- The user can select a fund from the list, and the system navigates to the 'SSO\_FUND.Fundapor' field.  
  
DB queries for Table reference CRUD operations only (With Usage):  
```sql  
SELECT fund\_short\_name, fund\_full\_name  
FROM AZBJ\_TFV\_FUND\_DEFINITION a, azbj\_cover\_funds b  
WHERE a.fund\_short\_name = b.fund\_name  
AND b.product\_id = :control.cn\_product\_id  
AND :susac.sa\_daterecd > NVL(start\_date, '01-apr-2005')  
AND :susac.sa\_daterecd < NVL(end\_date, '01-jan-3000')  
AND a.fund\_id <> 'FVFD000025'  
AND cover\_code = :covers.cv\_cover\_code  
ORDER BY fund\_short\_name;  
```

# View Previous Policies Details

Type: PREV\_POLICY

Title: View Previous Policies Details  
  
Acceptance Criteria:  
1. When the user navigates to the previous policies section, the system should display a list of previous policies with the following details:  
 - Policy Number  
 - Customer Name  
 - Status  
 - Reason for the policy status  
 - Comments  
2. The system should fetch and display the policy details based on the customer's ID.  
3. The system should handle any exceptions during data retrieval gracefully and ensure that the user is informed if any data is missing or cannot be retrieved.  
  
Definition of Done:  
1. The user can see a list of previous policies with all the specified details.  
2. The data is fetched correctly based on the customer's ID.  
3. The system handles exceptions and informs the user appropriately.  
4. The feature is tested and verified to work as expected.  
  
DB queries for Table reference CRUD operations only(With Usage):  
```sql  
-- Fetch policy details based on customer ID  
SELECT AZBJ\_PK0\_ACC.GET\_POLICY\_REF (A.CONTRACT\_ID) AS POLICY\_REF,  
 B.NAME,  
 C.CONTRACT\_STATUS AS STATUS  
 FROM OCP\_INTERESTED\_PARTIES A,  
 CP\_PARTNERS B,  
 OCP\_POLICY\_VERSIONS C,  
 OCP\_POLICY\_COVERS E,  
 OCP\_POLICY\_BASES F,  
 AZBJ\_POLICY\_CONTRACT\_EXT H,  
 CUSTOMER.AZBJ\_AUTO\_CP\_MERGE\_DETAILS I  
 WHERE A.PARTNER\_ID = :INSURED\_PERSON.IP\_PART\_ID  
 AND A.PARTNER\_ID = B.PART\_ID  
 AND A.CONTRACT\_ID = C.CONTRACT\_ID  
 AND A.CONTRACT\_ID = C.CONTRACT\_ID  
 AND A.TOP\_INDICATOR = 'Y'  
 AND A.ACTION\_CODE <> 'D'  
 AND C.TOP\_INDICATOR = 'Y'  
 AND E.CONTRACT\_ID = C.CONTRACT\_ID  
 AND A.CONTRACT\_ID = E.CONTRACT\_ID  
 AND E.ACTION\_CODE <> 'D'  
 AND E.TOP\_INDICATOR = 'Y'  
 AND E.COVER\_CODE LIKE 'L%'  
 AND F.CONTRACT\_ID = C.CONTRACT\_ID  
 AND A.CONTRACT\_ID = F.CONTRACT\_ID  
 AND E.CONTRACT\_ID = F.CONTRACT\_ID  
 AND F.ACTION\_CODE <> 'D'  
 AND F.TOP\_INDICATOR = 'Y'  
 AND H.CONTRACT\_ID = C.CONTRACT\_ID  
 AND A.CONTRACT\_ID = H.CONTRACT\_ID  
 AND H.CONTRACT\_ID = F.CONTRACT\_ID  
 AND H.ACTION\_CODE <> 'D'  
 AND A.IP\_NO = 1  
 AND A.PARTNER\_ID = I.PART\_ID  
 AND I.PART\_ID = B.PART\_ID  
 AND (C.CHANGE\_DESCRIPTION IN ('CI', 'DECLINED')  
 OR C.CHANGE\_DESCRIPTION LIKE '%POST%'  
 OR C.CHANGE\_DESCRIPTION LIKE 'PST%');  
  
-- Fetch reason and comments for a policy  
SELECT REQUEST\_REASON, USER\_COMMENT  
 INTO V\_REASON, V\_UW\_COMMENTS  
 FROM AZBJ\_COMPLAINTS  
 WHERE POLICY\_REF = :PREV\_POLICY.PREV\_POL\_NO  
 AND ROWNUM < 2;  
```

# Manage Unit-Linked Portfolio

Type: SOL\_UL\_AT\_PORTFOL

Detailed description: As a user, I want to manage the unit-linked portfolio by entering and viewing fund details, including the percentage allocation and the total fund value, so that I can effectively monitor and adjust my investments.  
  
Acceptance criteria:  
1. The user should be able to enter the percentage allocation for each fund in the portfolio.  
2. The system should automatically calculate and display the total fund value based on the entered percentages.  
3. The total fund value should be a sum of all the percentage allocations entered by the user.  
  
Definition of Done:  
1. The user interface allows for the entry of percentage allocations for each fund.  
2. The total fund value is correctly calculated and displayed based on the entered percentages.  
3. The functionality is tested and verified to ensure accuracy and usability.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include specific database queries.

# Display Approval Status, Substatus, and Remarks for Video Call Detail

Type: VIDEO\_CALLING\_DET

Title: Display Approval Status, Substatus, and Remarks for Video Call Detail  
  
Acceptance Criteria:  
1. The system should display the approval status, approval substatus, and remarks fields.  
2. The approval status and approval substatus should be displayed in uppercase.  
3. The fields should be read-only and not editable by the user.  
4. The fields should be clearly labeled with "Approval Status," "Approval Substatus," and "Remarks."  
  
Definition of Done:  
1. The approval status, approval substatus, and remarks fields are displayed on the screen.  
2. The fields are in uppercase and read-only.  
3. The labels for the fields are correctly displayed.  
4. The user can view the information without any errors.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include any database-specific queries or operations.

# Manage Fund Linkage to Units

Type: SOL\_UNIT\_LINKED

Title: Manage Fund Linkage to Units  
  
Acceptance Criteria:  
1. The system should allow the user to input and display the fund name and percentage allocation for each unit.  
2. The system should calculate and display the total fund value based on the percentage allocations entered.  
3. The system should validate the fund name against a predefined list of valid fund names.  
4. The system should ensure that the percentage allocation is a numeric value and initialize it to 0 if not provided.  
5. The system should sum the percentage allocations to calculate the total fund value.  
  
Definition of Done:  
1. The user can successfully input and display the fund name and percentage allocation for each unit.  
2. The total fund value is accurately calculated and displayed based on the percentage allocations.  
3. The fund name is validated against the predefined list.  
4. The percentage allocation is validated as a numeric value and initialized to 0 if not provided.  
5. The system is tested and verified to ensure all functionalities work as expected.  
  
DB queries for Table reference CRUD operations only(With Usage):  
```sql  
SELECT fund\_short\_name, fund\_full\_name  
 FROM AZBJ\_TFV\_FUND\_DEFINITION a, azbj\_cover\_funds b  
 WHERE a.fund\_short\_name = b.fund\_name  
 AND b.product\_id = :control.cn\_product\_id  
 AND :susac.sa\_daterecd >= NVL (start\_date, '01-apr-2005')  
 AND :susac.sa\_daterecd <= NVL (end\_date, '01-jan-3000')  
 AND cover\_code = :covers.cv\_cover\_code  
 AND 1 =  
 (CASE  
 WHEN :ALLOCATION.PORTFOLIO\_STRATEGY = 'SO'  
 AND a.FUND\_ID <> 'FVFD000025'  
 THEN  
 0  
 WHEN :ALLOCATION.PORTFOLIO\_STRATEGY <> 'SO'  
 THEN  
 1  
 ELSE  
 1  
 END);  
```

# Validate Cash Plus Pension Fund Apportionment

Type: SOL\_UNIT\_LINKED

Title: Validate Cash Plus Pension Fund Apportionment  
  
Acceptance Criteria:  
1. If the selected product ID is one of the following: 31, 32, 33, 34, 49, or 50, and the fund ID is 'NCPPF', then the apportionment percentage for the Cash Plus Pension Fund should not exceed 20%.  
2. If the apportionment percentage exceeds 20% under the above conditions, an error message should be displayed: "Apportionment for Cash Plus Pension Fund cannot be more than 20".  
3. The form status should be updated to 'Y' after validation.  
  
Definition of Done:  
- The system correctly identifies the specified product IDs and fund ID.  
- The system enforces the apportionment percentage limit of 20% for the Cash Plus Pension Fund.  
- An appropriate error message is displayed when the apportionment percentage exceeds the limit.  
- The form status is updated to 'Y' after the validation process is completed.

# Fund Selection on Double-Click

Type: SOL\_UNIT\_LINKED

Title: Fund Selection on Double-Click  
  
Acceptance Criteria:  
1. When the user double-clicks on the "Fund Name" field, a list of available funds should be displayed.  
2. The list of funds should be filtered based on the product ID and the date range.  
3. If the portfolio strategy is 'SO', only the fund with ID 'FVFD000025' should be displayed.  
4. The list should be populated dynamically based on the query results.  
  
Definition of Done:  
1. The user can double-click on the "Fund Name" field to open a list of available funds.  
2. The list of funds is correctly filtered based on the product ID, date range, and portfolio strategy.  
3. The user can select a fund from the list, and the selected fund is displayed in the "Fund Name" field.  
4. The functionality is tested and verified to work as expected.  
  
DB queries for Table reference CRUD operations only(With Usage):  
```sql  
SELECT fund\_short\_name, fund\_full\_name  
FROM AZBJ\_TFV\_FUND\_DEFINITION a, azbj\_cover\_funds b  
WHERE a.fund\_short\_name = b.fund\_name  
 AND b.product\_id = :control.cn\_product\_id  
 AND :susac.sa\_daterecd >= NVL(start\_date, '01-apr-2005')  
 AND :susac.sa\_daterecd <= NVL(end\_date, '01-jan-3000')  
 AND cover\_code = :covers.cv\_cover\_code  
 AND 1 = (CASE  
 WHEN :ALLOCATION.PORTFOLIO\_STRATEGY = 'SO' AND a.FUND\_ID <> 'FVFD000025' THEN 0  
 WHEN :ALLOCATION.PORTFOLIO\_STRATEGY <> 'SO' THEN 1  
 ELSE 1  
 END);  
```

# Manage PAN Details

Type: AZBJ\_PAN\_DET

Detailed description: As a user, I want to manage PAN details within the system, so that I can ensure accurate and up-to-date information for compliance and verification purposes.  
  
Acceptance criteria:  
1. The system should allow the user to input and display the following fields:  
 - PAN Number  
 - PAN Status  
 - Name Match  
 - Date of Birth (DOB) Match  
2. Each field should have a maximum length constraint:  
 - PAN Number: 100 characters  
 - PAN Status: 2000 characters  
 - Name Match: 2 characters  
 - DOB Match: 2 characters  
3. The fields should be displayed in a vertical orientation and should be aligned centrally.  
4. The user should be able to navigate through the records using a scrollbar.  
5. The system should display the fields on a tabbed canvas named "AML".  
  
Definition of Done:  
- The user can successfully input and view PAN details.  
- The fields adhere to the specified maximum length constraints.  
- The fields are displayed in a vertical orientation and are centrally aligned.  
- The user can navigate through the records using a scrollbar.  
- The fields are displayed on the "AML" tabbed canvas.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Not applicable as the provided XML content does not include specific database queries or operations.

# Manage Counter Offer Details in Digital Interface

Type: AZBJ\_CO\_DETAILS

Title: Manage Counter Offer Details in Digital Interface  
  
Acceptance Criteria:  
1. The interface should allow the user to:  
 - Select the type of counter offer from a predefined list.  
 - Enter the sum assured, premium term, benefit term, and premium amount.  
 - View calculated fields such as GST, deposit collected, net payable or excess paid, and rider details.  
 - Input additional details like policy reference, contract ID, product ID, and activity information.  
 - Display read-only fields for certain financial metrics and rider information.  
2. The interface should ensure that:  
 - The counter offer type is selectable from a list.  
 - Sum assured, premium term, benefit term, and premium are numeric fields.  
 - GST, deposit collected, net payable, and rider details are displayed but not editable.  
 - All input fields have appropriate labels and prompts for user guidance.  
 - The interface is visually organized with proper alignment and spacing for ease of use.  
  
Definition of Done:  
- The user interface for managing counter offers is implemented and accessible.  
- All specified fields and functionalities are available and working as per the acceptance criteria.  
- The interface is tested for usability and correctness of data handling.  
- Documentation is updated to reflect the new functionality.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Not applicable as the provided XML content does not include specific database queries or operations.

# Validate Premium Term Input

Type: AZBJ\_CO\_DETAILS

Title: Validate Premium Term Input  
  
Acceptance Criteria:  
1. The system should display an error message if the premium term entered is less than or equal to zero.  
2. The error message should be clear and indicate that the premium term cannot be less than or equal to zero.  
3. The system should not allow the user to proceed until a valid premium term is entered.  
  
Definition of Done:  
1. The validation logic for the premium term is implemented.  
2. The error message is displayed correctly when the premium term is invalid.  
3. The user is prevented from proceeding with an invalid premium term.  
4. 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 XML content does not include any direct database operations.

# Validate Benefit Term Field

Type: AZBJ\_CO\_DETAILS

Title: Validate Benefit Term Field  
  
Acceptance Criteria:  
1. The "Benefit Term" field should only accept numeric values greater than zero.  
2. If a user attempts to enter a value less than or equal to zero, an error message should be displayed stating, "Benefit Term cannot be less than or equal to zero."  
3. The error message should prevent the user from proceeding until a valid value is entered.  
  
Definition of Done:  
1. The "Benefit Term" field validation is implemented and tested.  
2. The error message is displayed correctly when invalid data is entered.  
3. The user is prevented from proceeding with invalid data in the "Benefit Term" field.  
4. All acceptance criteria are met and verified through testing.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include any direct database operations.

# Manage Counter Offer Type Based on Product and Group Status

Type: AZBJ\_CO\_DETAILS

Title: Manage Counter Offer Type Based on Product and Group Status  
  
Acceptance Criteria:  
1. When the checkbox for counter offer type is checked:  
 - If the product is not linked to a unit and the group product status is 'N':  
 - If the counter offer type is 'CO\_IP', the counter offer type field should be disabled for both insertion and update.  
 - If the counter offer type is not 'CO\_IP', the counter offer type field should be enabled for both insertion and update.  
 - If the product is linked to a unit or the group product status is not 'N':  
 - Display a warning message: "Cannot modify counter offer for ULIP/GROUP cases."  
 - Disable the counter offer type field for both insertion and update.  
 - Uncheck the checkbox for counter offer type.  
  
Definition of Done:  
- The system correctly enables or disables the counter offer type field based on the specified conditions.  
- The appropriate warning message is displayed when the conditions are not met.  
- The checkbox for counter offer type is unchecked when the conditions are not met.  
- The changes are navigated to the counter offer type field after processing.  
  
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.

# Calculate Total GST Amount Based on Sum Assured and Premium

Type: AZBJ\_CO\_DETAILS

Detailed description: As a user, I want the system to calculate the total GST amount based on the sum assured and premium values, so that I can ensure accurate tax calculations for insurance policies.  
  
Acceptance criteria:  
1. The system should calculate the total GST amount when both the sum assured and premium values are provided.  
2. The system should fetch the document date and service tax date based on the contract ID.  
3. The system should call the GST calculation procedure with the necessary parameters, including product ID, cover code, premium, effective date, policy year, mailing address pin code, service address pin code, sum assured, and event code.  
4. The system should handle any exceptions during the GST calculation process and set the total GST amount accordingly.  
5. The system should validate that the sum assured is greater than zero and display an error message if it is not.  
6. The system should ensure that the sum assured does not exceed a predefined maximum value for certain counter offer types and display an error message if it does.  
  
Definition of Done:  
1. The GST amount is accurately calculated and displayed based on the provided sum assured and premium values.  
2. Error messages are displayed for invalid sum assured values.  
3. The system handles exceptions gracefully during the GST calculation process.  
4. The user can proceed to the next item based on the counter offer type.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- The system should fetch the document date using the contract ID.  
- The system should fetch the service tax date using the contract ID and document date.  
- The system should call the GST calculation procedure with the necessary parameters to get the GST amount details.  
- The system should calculate the total GST amount by summing up the state GST, central GST, UT GST, and integrated GST values.

# Modify Counter Offer Type

Type: AZBJ\_CO\_DETAILS

Title: Modify Counter Offer Type  
  
Acceptance Criteria:  
1. If the product is of type 'ULIP' or 'GROUP', the system should display a warning message stating that counter offers cannot be modified for these product types.  
2. If the selected counter offer type is 'CO\_IP', the system should reset the selection and display an error message asking the user to select a different option.  
3. The system should check for duplicate counter offers. If a duplicate is found, the system should reset the selection and display an error message.  
4. If the selected counter offer type is valid and not a duplicate, the system should populate the counter offer details with predefined values and allow modifications based on the selected type:  
 - For 'CO\_SA', 'CO\_BTPT1', and 'CO\_BTPT2', specific fields should be populated and certain fields should be enabled or disabled for editing.  
5. If the selected counter offer type is invalid or a duplicate, the system should reset the selection and display an error message.  
  
Definition of Done:  
- The user can select a counter offer type from a list.  
- The system enforces business rules for modifying counter offers.  
- Appropriate messages are displayed based on the user's actions and selections.  
- The counter offer details are populated and editable based on the selected type.  
- The system handles invalid or duplicate selections by resetting the selection and displaying an error message.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include specific CRUD operations that can be directly executed in the database.

# GST Calculation for Premium

Type: AZBJ\_CO\_DETAILS

Title: GST Calculation for Premium  
  
Acceptance Criteria:  
1. When the user enters a valid premium amount and the sum assured is not null, the system should:  
 - Retrieve the document date associated with the contract.  
 - Fetch the service tax date based on the contract ID and document date.  
 - Retrieve the postal code of the insured person.  
 - Retrieve the agent code and branch code associated with the contract.  
 - Calculate the GST amount using the product ID, cover code, premium, effective date, postal codes, sum assured, and event code.  
 - Sum up the state GST, central GST, UT GST, and integrated GST to get the total GST amount.  
 - Display the total GST amount in the GST field.  
  
2. If the premium amount entered is less than or equal to zero, the system should display an error message stating "Premium cannot be less than or equal to zero."  
  
Definition of Done:  
- The GST calculation logic is implemented and tested.  
- The system correctly retrieves and uses the necessary data for GST calculation.  
- The error message for invalid premium amounts is displayed as specified.  
- The total GST amount is displayed in the GST field when valid data is entered.  
- All acceptance criteria are met and verified through testing.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Retrieve postal code:  
 ```sql  
 SELECT postcode  
 INTO v\_pin\_code  
 FROM cp\_addresses a, cp\_partners b  
 WHERE a.add\_id = b.add\_id AND b.part\_id = :insured\_person.ip\_part\_id;  
 ```  
  
- Retrieve agent code:  
 ```sql  
 SELECT agent\_code  
 INTO v\_agnt  
 FROM azbj\_policy\_agents\_rep  
 WHERE CONTRACT\_ID = :control.cn\_contract\_id AND top\_indicator = 'Y' AND action\_code <> 'D';  
 ```  
  
- Retrieve branch code:  
 ```sql  
 SELECT branch\_code  
 INTO v\_branch\_code  
 FROM azbj\_v\_agents  
 WHERE reference\_code = v\_agnt;  
 ```  
  
- Calculate GST amount:  
 ```sql  
 azbj\_pk\_gst.get\_gst\_amount (  
 p\_product\_id => :AZBJ\_CO\_DETAILS.PRODUCT\_ID,  
 p\_cover\_code => :AZBJ\_CO\_DETAILS.cover\_code,  
 p\_premium => :AZBJ\_CO\_DETAILS.PREMIUM,  
 p\_effectve\_date => NVL(v\_service\_tax\_date, pme\_api.opus\_date),  
 p\_policy\_year => 1,  
 p\_mailadd\_pin\_code => v\_pin\_code,  
 p\_servadd\_pin\_code => azbj\_pk\_gst.get\_coll\_branch\_pincode(v\_branch\_code),  
 p\_sumassured => :AZBJ\_CO\_DETAILS.SUM\_ASSURED,  
 p\_event\_code => 'PREMIUM',  
 p\_error => v\_error,  
 p\_gst\_dtl\_string => v\_azbj\_gst\_tab  
 );  
 ```  
  
- Calculate total GST:  
 ```sql  
 azbj\_pk\_gst.gst\_tax\_calc\_prc (  
 p\_gst\_dtls => v\_azbj\_gst\_tab,  
 p\_amount => v\_state\_gst  
 );  
 ```

# View and Select Reasons for Counter Offers

Type: AZBJ\_CO\_REASON

Title: View and Select Reasons for Counter Offers  
  
Acceptance Criteria:  
- The interface should display a list of reasons for counter offers.  
- Each reason should be displayed in a designated area with a maximum length of 2000 characters.  
- There should be a checkbox next to each reason, allowing the user to select or deselect the reason.  
- The checkbox should have two states: checked (Y) and unchecked (N).  
- The interface should be visually organized and easy to navigate, with a scrollbar if the list of reasons exceeds the display area.  
  
Definition of Done:  
- The user can see a list of reasons for counter offers.  
- The user can select or deselect reasons using checkboxes.  
- The interface is user-friendly and visually organized.  
- 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 XML content does not include any database-specific queries or operations.

# Limit Counter Offer Reasons Selection

Type: AZBJ\_CO\_REASON

Title: Limit Counter Offer Reasons Selection  
  
Acceptance Criteria:  
1. When a reason is selected, the system should count the number of reasons already selected.  
2. If the number of selected reasons exceeds three, the system should display an error message: "Cannot select more than 3 reasons".  
3. If the number of selected reasons is three or fewer, the selected reason should be appended to the list of reasons.  
4. The error message should be displayed using a predefined alert mechanism.  
  
Definition of Done:  
1. The user is able to select reasons for a counter offer.  
2. The system correctly counts the number of selected reasons.  
3. An error message is displayed if more than three reasons are selected.  
4. The selected reason is appended to the list if the total number of reasons is three or fewer.  
5. The error message is displayed using the predefined alert mechanism.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided logic does not involve direct CRUD operations on database tables.

# View Counter Offer Details

Type: AZBJ\_CO\_DET\_HIS

Title: View Counter Offer Details  
  
Acceptance Criteria:  
1. The user should be able to view the following attributes for each counter offer:  
 - Counter Offer Type  
 - Policy Reference  
 - Contract ID  
 - Product ID  
 - Sum Assured  
 - Benefit Term  
 - Premium Paying Term  
 - Premium  
 - GST  
 - Proposal Deposit Collected  
 - Net Payable/Excess Paid  
 - Rider Name  
 - Rider Cover  
 - Counter Offer Reason  
 - Activity Number  
 - Activity Date  
 - Event Type  
 - Package Code  
 - User ID  
 - Various percentage and amount fields (ML Perc, ML Amt, OC Perc, OC Amt, SR Perc, SR Amt, NRI Perc, NRI Amt)  
2. The information should be displayed in a user-friendly format with appropriate labels and alignment.  
3. The user should not be able to delete any records.  
4. The user should be able to navigate through the records using a vertical scrollbar.  
  
Definition of Done:  
- The user interface displays all the specified attributes for each counter offer.  
- The attributes are properly labeled and aligned for easy readability.  
- The vertical scrollbar allows the user to navigate through the records.  
- The delete functionality is disabled for the records.  
- The user can view all the required information without any errors or missing data.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- No specific DB queries provided in the XML content.

# User Interface for Counter Offer Actions

Type: AZBJ\_CO\_CONTROL

Detailed description: As a user, I want to validate, generate, and submit counter offers through a user interface that provides distinct buttons for each action, ensuring a streamlined and efficient process.  
  
Acceptance criteria:  
1. The interface should have a "VALIDATE" button that allows users to validate the counter offer.  
2. The interface should have a "GENERATE LETTER" button that is initially disabled and becomes enabled only after validation.  
3. The interface should have a "GENERATE BI" button that is always enabled.  
4. The interface should have a "SUBMIT" button that is initially disabled and becomes enabled only after the counter offer is generated.  
5. The interface should have a hidden "TEST" item for internal use, not visible to the user.  
  
Definition of Done:  
1. The "VALIDATE" button is functional and triggers the validation process.  
2. The "GENERATE LETTER" button is disabled by default and becomes enabled after successful validation.  
3. The "GENERATE BI" button is always enabled and functional.  
4. The "SUBMIT" button is disabled by default and becomes enabled after the counter offer is generated.  
5. The "TEST" item is present in the interface but not visible to the user.  
6. The interface is tested and verified for usability and functionality.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include specific database queries.

# Generate Counter Offer Letter

Type: AZBJ\_CO\_CONTROL

Title: Generate Counter Offer Letter  
  
Acceptance Criteria:  
1. The system should navigate to the 'Counter Offer Details' section and iterate through all records.  
2. For each record, if the counter offer type is specified and the checkbox is checked, the system should collect the relevant details based on the counter offer type (CO\_IP, CO\_SA, CO\_BTPT1, CO\_BTPT2).  
3. The system should then navigate to the 'Counter Offer Reason' section and ensure that at least one and no more than three reasons are selected.  
4. The system should collect the selected reasons and store them in a variable.  
5. The system should retrieve the contact details of the policyholder or insured person from the database.  
6. The system should determine the modification type based on the collected counter offer details.  
7. The system should create a JSON object with all the collected details for PDF generation.  
8. The system should call an external API to generate the PDF using the JSON object.  
9. If the PDF generation is successful, the system should enable the 'Submit' button.  
10. The system should display a success message indicating that the counter offer letter has been generated successfully.  
  
Definition of Done:  
- The counter offer letter is generated based on the provided details.  
- The system ensures that the correct number of reasons are selected.  
- The JSON object is created with all necessary details.  
- The PDF is generated successfully using the external API.  
- The 'Submit' button is enabled upon successful PDF generation.  
- A success message is displayed to the user.  
  
DB queries for Table reference CRUD operations only (With Usage):  
```sql  
SELECT NVL (CONTACT1, CONTACT2),  
 EMAIL,  
 SEX,  
 FIRST\_NAME,  
 SURNAME,  
 NATIONALITY --to be changed as city  
 INTO v\_mobile,  
 v\_email,  
 v\_gender,  
 v\_first\_name,  
 v\_last\_name,  
 v\_city  
 FROM cp\_partners  
 WHERE part\_id = NVL(:policy\_holder.ph\_part\_id, :insured\_person.ip\_part\_id);  
```

# Generate BI Report for Insurance Covers

Type: AZBJ\_CO\_CONTROL

Detailed description: As a user, I want to generate a Business Intelligence (BI) report for different types of insurance covers (CO\_IP, CO\_SA, CO\_BTPT1, CO\_BTPT2) by pressing a button, so that I can receive a new BI number from the NVEST API and update the relevant details in the system.  
  
Acceptance criteria:  
1. When the button is pressed, the system should navigate to the 'AZBJ\_CO\_DETAILS' section and iterate through all records.  
2. For each record, based on the type of cover (CO\_IP, CO\_SA, CO\_BTPT1, CO\_BTPT2), the system should populate the cover details and call the NVEST API to generate a new BI number.  
3. The system should handle different responses from the NVEST API:  
 - If the response indicates success, the system should update the BI details (e.g., new BI PDF, message, status, premium term, benefit term, new BI number, quote ID, premium, sum assured) in the 'AZBJ\_CO\_DETAILS' section.  
 - If the response indicates failure or an internal server error, the system should display an appropriate warning message and reset the visual attributes of the relevant fields.  
4. The system should calculate and update the GST amount and the balance of payment (BOP) based on the premium and other details.  
5. The system should log relevant information for each BI generation attempt.  
6. The system should enable or disable the 'PB\_GENERATE\_CO' button based on the error flag status.  
  
Definition of Done:  
- The BI report generation process is triggered by pressing the button.  
- The system correctly navigates through the records and calls the NVEST API for each cover type.  
- The system handles API responses appropriately and updates the relevant details.  
- The system calculates and updates the GST amount and BOP.  
- Relevant information is logged for each BI generation attempt.  
- The 'PB\_GENERATE\_CO' button is enabled or disabled based on the error flag status.  
- The process is tested and verified to work as expected without any Oracle Forms-specific terminology.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Select queries to fetch agent code, branch code, and postal code.  
- Insert logs into the `azbj\_new\_bbu\_utilities.bbu\_ins\_log` table.  
- Update the `carequote\_bi` table to fetch the premium based on the request ID.

# Submit Counter Offer Details

Type: AZBJ\_CO\_CONTROL

Title: Submit Counter Offer Details  
  
Acceptance Criteria:  
1. When the submit button is pressed, the system should navigate to the details section and count the number of records.  
2. The system should check if there are any existing offers for the given policy reference. If offers exist, it should update the top indicator to 'N'.  
3. For each record in the details section, if the counter offer type is not null and the check counter offer type flag is 'Y', the system should insert a new record into the modified offer table with the provided details.  
4. The system should commit the transaction and display a success message indicating that the counter offer was modified successfully.  
5. The form status should be updated to 'N', and the view should be hidden.  
  
Definition of Done:  
- The submit button functionality is implemented and tested.  
- The system correctly navigates, counts records, updates existing offers, inserts new records, commits the transaction, and displays the success message.  
- The form status is updated, and the view is hidden as specified.  
- All acceptance criteria are met without any errors.  
  
DB queries for Table reference CRUD operations only(With Usage):  
```sql  
-- Check if there are existing offers for the given policy reference  
SELECT COUNT(1) INTO V\_CNT FROM AZBJ\_MODIFIED\_OFFER WHERE policy\_ref = :control.cn\_policy\_ref;  
  
-- Update existing offers if they exist  
UPDATE AZBJ\_MODIFIED\_OFFER   
SET TOP\_INDICATOR = 'N'  
WHERE policy\_ref = :control.cn\_policy\_ref;  
  
-- Insert new record into the modified offer table  
INSERT INTO CUSTOMER.AZBJ\_MODIFIED\_OFFER (  
 policy\_ref, contract\_id, APPLICATION\_NO, activity\_no, activity\_date, event\_type, user\_id,   
 TOP\_INDICATOR, product\_id, package\_code, cover\_code, sum\_assured, benefit\_term, premium\_term,   
 premium, GST, DEPOSIT\_COLL, BOP, CHECK\_CO\_FLAG, multiplier, ml\_perc, ml\_amt, oc\_perc, oc\_amt,   
 sr\_perc, sr\_amt, nri\_perc, nri\_amt, CO\_REASON1, CO\_REASON2, CO\_REASON3, sign\_mismatch\_flg,   
 OLD\_BI\_NO, B\_URL, BI\_NO, B\_MESSAGE, B\_STEP, B\_STATUS, B\_PREMIUM\_TERM, B\_BENEFIT\_TERM, B\_PREM,   
 B\_SUM\_ASSURED, B\_QUOTEID, RIDER\_NAME, RIDER\_COVER  
) VALUES (  
 :control.cn\_policy\_ref, :control.cn\_contract\_id, TO\_CHAR(NVL(:INSURED\_PERSON.IP\_VERF\_NO, :INSURED\_PERSON.IP\_SIGN\_CARD\_NO)),   
 :AZBJ\_CO\_DETAILS.activity\_no, NVL(:AZBJ\_CO\_DETAILS.activity\_date, SYSDATE), :AZBJ\_CO\_DETAILS.co\_type, USER,   
 'Y', :AZBJ\_CO\_DETAILS.product\_id, :AZBJ\_CO\_DETAILS.COVER\_NAME, :AZBJ\_CO\_DETAILS.cover\_code,   
 :AZBJ\_CO\_DETAILS.sum\_assured, :AZBJ\_CO\_DETAILS.benefit\_term, :AZBJ\_CO\_DETAILS.premium\_term,   
 :AZBJ\_CO\_DETAILS.premium, :AZBJ\_CO\_DETAILS.GST, :AZBJ\_CO\_DETAILS.DEPOSIT\_COLL, :AZBJ\_CO\_DETAILS.BOP,   
 :AZBJ\_CO\_DETAILS.check\_co\_type, to\_number(:AZBJ\_CO\_DETAILS.multiplier), to\_number(:AZBJ\_CO\_DETAILS.ml\_perc),   
 to\_number(:AZBJ\_CO\_DETAILS.ml\_amt), to\_number(:AZBJ\_CO\_DETAILS.oc\_perc), to\_number(:AZBJ\_CO\_DETAILS.oc\_amt),   
 to\_number(:AZBJ\_CO\_DETAILS.sr\_perc), to\_number(:AZBJ\_CO\_DETAILS.sr\_amt), to\_number(:AZBJ\_CO\_DETAILS.nri\_perc),   
 to\_number(:AZBJ\_CO\_DETAILS.nri\_amt), pk\_vars.v\_co1, pk\_vars.v\_co2, pk\_vars.v\_co3, :coverhead.c\_offer\_sign,   
 :agents.bi\_no, :AZBJ\_CO\_DETAILS.NEW\_BI\_PDF, to\_number(:AZBJ\_CO\_DETAILS.NEW\_BI\_NO), :AZBJ\_CO\_DETAILS.B\_MESSAGE,   
 :AZBJ\_CO\_DETAILS.B\_STEP, :AZBJ\_CO\_DETAILS.B\_STATUS, to\_number(:AZBJ\_CO\_DETAILS.B\_PREMIUM\_TERM),   
 to\_number(:AZBJ\_CO\_DETAILS.B\_BENEFIT\_TERM), to\_number(:AZBJ\_CO\_DETAILS.B\_PREM), to\_number(:AZBJ\_CO\_DETAILS.B\_SUM\_ASSURED),   
 :AZBJ\_CO\_DETAILS.B\_QUOTEID, :AZBJ\_CO\_DETAILS.RIDER\_NAME, :AZBJ\_CO\_DETAILS.RIDER\_COVER  
);  
```

# Validate Counter Offer Details

Type: AZBJ\_CO\_CONTROL

User Story: Validate Counter Offer Details  
  
Detailed Description:  
As a user, I need to validate the counter offer details to ensure that all necessary conditions and validations are met before proceeding with the policy issuance. This includes checking the selection of counter offer reasons, ensuring the correct calculation of various parameters such as age, telephone numbers, and premium amounts, and validating the product-specific conditions.  
  
Acceptance Criteria:  
1. The system should navigate to the 'Counter Offer Reason' section and ensure that at least one reason is selected and no more than three reasons are selected.  
2. The system should concatenate the selected reasons and store them for further processing.  
3. The system should validate the beneficiary details and ensure they are correctly populated.  
4. The system should check the product ID and validate the counter offer types and their respective conditions.  
5. The system should calculate the insured person's age and joint life flag based on the date of birth and other relevant details.  
6. The system should calculate the telephone numbers for both the insured person and the policyholder.  
7. The system should validate the counter offer details for different types such as CO\_IP, CO\_SA, CO\_BTPT1, and CO\_BTPT2, ensuring all necessary fields are populated and conditions are met.  
8. The system should perform product-specific validations, including checking cover codes, sum assured, benefit term, premium term, and other relevant parameters.  
9. The system should log any errors encountered during the validation process and display appropriate error messages to the user.  
10. The system should enable or disable the 'Generate Counter Offer' button based on the validation results.  
  
Definition of Done:  
- The counter offer validation process is implemented and tested.  
- All acceptance criteria are met.  
- The system correctly handles and logs errors.  
- The 'Generate Counter Offer' button is enabled or disabled based on the validation results.  
- The user is able to proceed with the policy issuance if all validations are successful.  
  
DB Queries for Table Reference CRUD Operations:  
- Not applicable as the provided XML content does not include direct database queries that can be executed independently of Oracle Forms constructs.