User Stories

# Manage Questionnaire Details

Type: AZBJ\_QUESTIONNAIRE

Title: Manage Questionnaire Details  
  
Acceptance Criteria:  
1. When a new record is created, the system should determine the strip number for the questionnaire based on the question ID and sub-question. If the strip number is valid and different from the previous strip number, the system should load the corresponding questionnaire image.  
2. When a mouse click occurs, if the clicked item is not a button, the system should navigate to the clicked item.  
  
Definition of Done:  
1. The form should display fields for entering answers, question descriptions, member names, and other related details.  
2. The form should include buttons for entering answers and populating questions.  
3. The form should handle navigation and display of records correctly.  
4. The form should load the appropriate questionnaire image based on the strip number when a new record is created.  
5. The form should navigate to the clicked item if it is not a button.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Not applicable as the provided XML content does not include specific database queries.

# Enter Answers Button Functionality

Type: AZBJ\_QUESTIONNAIRE

Title: Enter Answers Button Functionality  
  
Acceptance Criteria:  
1. When the user presses the "Enter Answers" button:  
 - If the question ID is 6, the sub-question is 0, and the correct value is 'Y', the system should set the status to 'Y'.  
 - If the question ID is 6, the sub-question is 0, and the correct value is 'N', the system should set the status to 'N'.  
 - If the question ID is 6, the sub-question is 0, and the correct value is neither 'N' nor 'Y', the system should set the status to 'I'.  
  
Definition of Done:  
- The "Enter Answers" button should be functional and trigger the appropriate logic based on the user's input.  
- The system should correctly update the status based on the conditions specified.  
- 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.

# Handle Input for Questionnaire Answer Field (ANS1)

Type: AZBJ\_QUESTIONNAIRE

Detailed description: As a user, I want the system to handle the input for the questionnaire answer field (ANS1) in a way that ensures data integrity and proper navigation based on specific conditions.  
  
Acceptance criteria:  
1. When the user navigates to the ANS1 field:  
 - If the `question\_id` is 4 and `sub\_question` is not in (0, 1, 2, 6), and the variable `v\_isalch` is not 'Y', the system should automatically move to the next record.  
2. When the user changes the value in the ANS1 field:  
 - If `question\_id` is 4 and `sub\_question` is 2:  
 - If ANS1 is 'Y', set `v\_isalch` to 'Y'.  
 - If ANS1 is 'N', set `v\_isalch` to 'N'.  
 - If ANS1 is neither 'N' nor 'Y', set `v\_isalch` to 'I'.  
  
Definition of Done:  
- The system correctly navigates to the next record based on the specified conditions.  
- The variable `v\_isalch` is updated correctly based on the value of ANS1 when `question\_id` is 4 and `sub\_question` is 2.  
- The functionality is tested and verified to ensure it meets the acceptance criteria.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- No direct database queries are provided in the XML content for CRUD operations. The logic primarily involves conditional checks and variable assignments.

# Populate Questionnaire Details Based on Conditions and User Inputs

Type: AZBJ\_QUESTIONNAIRE

Title: Populate Questionnaire Details Based on Conditions and User Inputs  
  
Acceptance Criteria:  
1. When the "Populate Questions" button is pressed, the system should check if the 'FREEZE\_QUEST' flag is enabled.  
2. If the 'FREEZE\_QUEST' flag is enabled, the system should ensure that the fields 'form\_question\_no' and 'question\_desc' are editable and insertable.  
3. Depending on the value of 'v\_grp\_product', the system should call the appropriate procedure to populate questions based on user inputs such as gender, part ID, and name.  
4. If the 'FREEZE\_QUEST' flag is enabled, the system should count the number of online activities related to the application number and log this information.  
5. If online activities are found, the system should make the fields 'form\_question\_no', 'question\_desc', 'CORRECT\_VALUE', and 'ANS\_DESC' non-editable and non-insertable.  
6. The system should handle any exceptions by logging an error message.  
  
Definition of Done:  
- The "Populate Questions" button functionality is implemented and tested.  
- The system correctly checks and updates field properties based on the 'FREEZE\_QUEST' flag.  
- The appropriate procedures are called based on the 'v\_grp\_product' value.  
- Online activity counts are correctly logged.  
- Fields are made non-editable and non-insertable when online activities are found.  
- Exception handling is implemented and tested.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- SELECT COUNT (1) INTO v\_online\_cnt FROM azbj\_nbtab\_activity\_dtls WHERE application\_no = :correct\_value.appln\_no AND ROWNUM = 1 AND module\_flag = 'INSTAB';  
- SELECT COUNT (1) INTO v\_online\_cnt FROM azbj\_phub\_tracker WHERE application\_no = :correct\_value.appln\_no AND remarks LIKE '%ONLINE%';  
  
These queries are used to count the number of online activities related to the application number.

# Validate and Process Questionnaire Responses

Type: AZBJ\_QUESTIONNAIRE

User Story: Validate and Process Questionnaire Responses  
  
Detailed description:   
As a user, I want the system to validate and process responses to specific questionnaire questions so that the data integrity is maintained and appropriate actions are taken based on the responses.  
  
Acceptance criteria:  
1. If the question ID is 6 and the sub-question is 5, and the response is null, an error message "Husband Insurance cannot be null" should be displayed.  
2. If the question ID is 6 and the sub-question is 0:  
 - If the response is 'Y', set the pregnancy status to 'Y'.  
 - If the response is 'N', set the pregnancy status to 'N'.  
 - If the response is neither 'N' nor 'Y', set the pregnancy status to 'I'.  
3. If the question ID is 6 and the sub-question is 1:  
 - If the pregnancy status is 'I', display an error message "Please check pregnancy status."  
 - If the pregnancy status is 'N' and the response is not null, display an error message "She is not pregnant... Invalid Entry."  
 - If the pregnancy status is 'Y':  
 - If the response is not between 1 and 12, display an error message "The Pregnancy months can be between 1 and 12."  
 - If the response is null, display an error message "Please enter the month of pregnancy."  
4. If the question ID is 4 and the sub-question is 2:  
 - If the response is 'Y', set the alcoholism status to 'Y'.  
 - If the response is 'N', set the alcoholism status to 'N'.  
 - If the response is neither 'N' nor 'Y', set the alcoholism status to 'I'.  
5. If the question ID is 4 and the sub-question is 3, 4, or 5:  
 - If the alcoholism status is 'I', display an error message "Please check Alcoholism Question."  
 - If the alcoholism status is 'N' and the response is not null, display an error message "The person does not drink alcohol... Invalid answer."  
 - If the alcoholism status is 'Y' and the response is null, display an error message "Please give in details of the drinking habits."  
  
Definition of Done:  
- The system correctly validates the responses based on the specified conditions.  
- Appropriate error messages are displayed when validation fails.  
- The system updates the status variables (pregnancy and alcoholism) based on the responses.  
- The system processes the next record or item as required after validation.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Not applicable as the provided XML content does not include direct database CRUD operations that can be executed independently of Oracle Forms constructs.

# Validation and Navigation Logic for ANS\_DESC Field

Type: AZBJ\_QUESTIONNAIRE

Detailed description: As a user, I want to ensure that the description field for a questionnaire answer is validated correctly based on specific conditions, so that the data entered is consistent and accurate.  
  
Acceptance criteria:  
1. If the `correct\_value` field is not 'Y' and the `ANS\_DESC` field is not null, and the `question\_id` is not 85 and `sub\_question` is not in (68, 69), then an error message should be displayed stating, "Answer to the question is N. Hence cannot enter description."  
2. When navigating downwards through the records:  
 - If the current record is the last one and the proposal status is 'D2':  
 - If the proposal type is 'N', load the questionnaire image with ID 46.  
 - If the proposal type is 'O', load the questionnaire image with ID 14.  
 - Move to the `proposal\_sign\_date` field.  
 - If the proposal status is not 'D2', load the questionnaire image with ID 1 and move to the save field.  
 - If the current record is not the last one, move to the next record and then to the `correct\_value` field.  
3. When navigating upwards through the records, move to the previous record and then to the `correct\_value` field.  
4. When navigating to the next item:  
 - If the current record is the last one, load the questionnaire image with ID 1, move to the save field, and set the next item to the save field.  
 - If the current record is not the last one, move to the next record, then to the `correct\_value` field, and set the next item to the `correct\_value` field.  
  
Definition of Done:  
- The validation logic for the `ANS\_DESC` field is implemented and tested.  
- Navigation logic for moving up and down through records is implemented and tested.  
- The error message is displayed correctly under the specified conditions.  
- The correct questionnaire images are loaded based on the proposal type and status.  
- The navigation to the next item works 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.

# Validate and Process Questionnaire Responses

Type: AZBJ\_QUESTIONNAIRE

User Story: Validate and Process Questionnaire Responses  
  
Detailed description:   
As a user, I want the system to validate and process responses to specific questionnaire questions so that the data integrity is maintained and appropriate actions are taken based on the responses.  
  
Acceptance criteria:  
1. If the question ID is 6 and the sub-question is 5, and the response is null, an error message "Husband Insurance cannot be null" should be displayed.  
2. If the question ID is 6 and the sub-question is 0:  
 - If the response is 'Y', set the pregnancy status to 'Y'.  
 - If the response is 'N', set the pregnancy status to 'N'.  
 - If the response is neither 'N' nor 'Y', set the pregnancy status to 'I'.  
3. If the question ID is 6 and the sub-question is 1:  
 - If the pregnancy status is 'I', display an error message "Please check pregnancy status."  
 - If the pregnancy status is 'N' and the response is not null, display an error message "She is not pregnant... Invalid Entry."  
 - If the pregnancy status is 'Y':  
 - If the response is not between 1 and 12, display an error message "The Pregnancy months can be between 1 and 12."  
 - If the response is null, display an error message "Please enter the month of pregnancy."  
4. If the question ID is 4 and the sub-question is 2:  
 - If the response is 'Y', set the alcoholism status to 'Y'.  
 - If the response is 'N', set the alcoholism status to 'N'.  
 - If the response is neither 'N' nor 'Y', set the alcoholism status to 'I'.  
5. If the question ID is 4 and the sub-question is 3, 4, or 5:  
 - If the alcoholism status is 'I', display an error message "Please check Alcoholism Question."  
 - If the alcoholism status is 'N' and the response is not null, display an error message "The person does not drink alcohol... Invalid answer."  
 - If the alcoholism status is 'Y' and the response is null, display an error message "Please give in details of the drinking habits."  
  
Definition of Done:  
- The system correctly validates the responses based on the specified conditions.  
- Appropriate error messages are displayed when validation fails.  
- The system updates the status variables (e.g., pregnancy status, alcoholism status) based on the responses.  
- The system navigates to the next record or item as required by the logic.  
- All acceptance criteria are met and tested.  
  
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 Loading of Questionnaire Images Based on Proposal Type and Question ID

Type: AZBJ\_QUESTIONNAIRE

Title: Automatic Loading of Questionnaire Images Based on Proposal Type and Question ID  
  
Acceptance Criteria:  
1. The system should determine the proposal type and the number of pages based on the application number.  
2. If the proposal type is 'N' and the number of pages is 50, the system should load specific images based on the question ID and sub-question values.  
3. If the proposal type is 'O' and the number of pages is 27, the system should load different specific images based on the question ID and sub-question values.  
4. The system should navigate to the next or previous record based on the user's key press (KEY-DOWN or KEY-UP).  
5. The system should update the `v\_isalch` variable based on the answer to question ID 4, sub-question 2.  
6. The system should skip to the next record if certain conditions are met when a new item instance is created.  
  
Definition of Done:  
- The system correctly identifies the proposal type and number of pages.  
- The appropriate image is loaded based on the question ID and sub-question values.  
- Navigation between records works as expected with the correct images being displayed.  
- The `v\_isalch` variable is updated correctly based on the user's input.  
- The system skips to the next record when the specified conditions are met.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- No direct database queries are provided in the XML content that can be executed without modification. The logic involves procedural calls and conditional checks that are specific to the Oracle Forms environment.

# Validation of 'Correct Value' Field in Questionnaire

Type: AZBJ\_QUESTIONNAIRE

Title: Validation of 'Correct Value' Field in Questionnaire  
  
Acceptance Criteria:  
1. If the question ID is 6 and the sub-question is 5, and the 'Correct Value' is null, an error message 'Husband Insurance cannot be null' should be displayed.  
2. If the question ID is 6 and the sub-question is 0:  
 - If the 'Correct Value' is 'Y', set the pregnancy status to 'Y'.  
 - If the 'Correct Value' is 'N', set the pregnancy status to 'N'.  
 - If the 'Correct Value' is neither 'N' nor 'Y', set the pregnancy status to 'I'.  
3. If the question ID is 6 and the sub-question is 1:  
 - If the pregnancy status is 'I', display an error message 'Please check pregnancy status.'  
 - If the pregnancy status is 'N' and the 'Correct Value' is not null, display an error message 'She is not pregnant... Invalid Entry.'  
 - If the pregnancy status is 'Y':  
 - If the 'Correct Value' is not between 1 and 12, display an error message 'The Pregnancy months can be between 1 and 12.'  
 - If the 'Correct Value' is null, display an error message 'Please enter the month of pregnancy.'  
4. If the question ID is 4 and the sub-question is 2:  
 - If the 'Correct Value' is 'Y', set the alcoholism status to 'Y'.  
 - If the 'Correct Value' is 'N', set the alcoholism status to 'N'.  
 - If the 'Correct Value' is neither 'N' nor 'Y', set the alcoholism status to 'I'.  
5. If the question ID is 4 and the sub-question is 3, 4, or 5:  
 - If the alcoholism status is 'I', display an error message 'Please check Alcoholism Question.'  
 - If the alcoholism status is 'N' and the 'Correct Value' is not null, display an error message 'The person does not drink alcohol... Invalid answer.'  
 - If the alcoholism status is 'Y' and the 'Correct Value' is null, display an error message 'Please give in details of the drinking habits.'  
  
Definition of Done:  
- The 'Correct Value' field is validated according to the specified conditions.  
- Appropriate error messages are displayed based on the validation rules.  
- The pregnancy and alcoholism statuses are correctly set based on the 'Correct Value' input.  
- The user is guided to the next appropriate field or record based on the validation outcomes.

# Validate Questionnaire Responses for Pregnancy and Alcoholism

Type: AZBJ\_QUESTIONNAIRE

Detailed description: As a user, I want to ensure that the system validates specific questionnaire responses related to pregnancy and alcoholism, so that the data entered is accurate and consistent with the business rules.  
  
Acceptance criteria:  
1. If the question ID is 6 and the sub-question is 5, and the correct value is null, the system should display an error message: "Husband Insurance cannot be null."  
2. If the question ID is 6 and the sub-question is 0:  
 - If the correct value is 'Y', set the pregnancy status to 'Y'.  
 - If the correct value is 'N', set the pregnancy status to 'N'.  
 - If the correct value is neither 'N' nor 'Y', set the pregnancy status to 'I'.  
3. If the question ID is 6 and the sub-question is 1:  
 - If the pregnancy status is 'I', display an error message: "Please check pregnancy status."  
 - If the pregnancy status is 'N' and the correct value is not null, display an error message: "She is not pregnant... Invalid Entry."  
 - If the pregnancy status is 'Y':  
 - If the correct value is not between 1 and 12, display an error message: "The Pregnancy months can be between 1 and 12."  
 - If the correct value is null, display an error message: "Please enter the month of pregnancy."  
4. If the question ID is 4 and the sub-question is 2:  
 - If the correct value is 'Y', set the alcoholism status to 'Y'.  
 - If the correct value is 'N', set the alcoholism status to 'N'.  
 - If the correct value is neither 'N' nor 'Y', set the alcoholism status to 'I'.  
5. If the question ID is 4 and the sub-question is 3, 4, or 5:  
 - If the alcoholism status is 'I', display an error message: "Please check Alcoholism Question."  
 - If the alcoholism status is 'N' and the correct value is not null, display an error message: "The person does not drink alcohol... Invalid answer."  
 - If the alcoholism status is 'Y' and the correct value is null, display an error message: "Please give in details of the drinking habits."  
  
Definition of Done:  
- The system correctly validates the questionnaire responses based on the specified business rules.  
- Appropriate error messages are displayed when validation fails.  
- The pregnancy and alcoholism statuses are correctly set based on the responses.  
- The user can proceed to the next item or record only if the validation passes.  
  
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.

# Handle Navigation and Validation of Questionnaire Responses

Type: AZBJ\_QUESTIONNAIRE

Detailed description: As a user, I want the system to handle the navigation and validation of questionnaire responses based on specific conditions, so that the correct images are loaded and the appropriate fields are updated or restricted.  
  
Acceptance criteria:  
1. When the user navigates to the next item:  
 - If the current value is not in the set {1, 2, 3, 4, 7, 8}, an error message "Invalid value entered" should be displayed.  
 - If the value of `correct` is 3 and the answers `ans1` and `ans\_de1` do not match, an error message "DE1 and DE2 Values Are Not Matching" should be displayed.  
 - If the value of `correct` is in {1, 2} and the answers `ans1` and `ans\_de1` match, an error message "Please select 3, 4, 7 or 8" should be displayed.  
 - If the value of `correct` is 3 or 2, the `correct\_value` should be set to `ans1`, and the `correct\_value` field should be made read-only.  
 - If the value of `correct` is 1, the `correct\_value` should be set to `ans\_de1`, and the `correct\_value` field should be made read-only.  
 - If the value of `correct` is in {4, 7, 8}, the `correct\_value` field should be made editable.  
  
2. When the user navigates to the next record:  
 - If the current value is 4, the focus should move to the `correct\_value` field.  
 - If the last record is reached, the focus should move to the `POPULATE\_QUESTIONS` field.  
 - Otherwise, the focus should move to the `correct\_value` field, and the next item trigger should be executed.  
  
Definition of Done:  
- The system correctly handles navigation and validation based on the specified conditions.  
- Appropriate error messages are displayed when validation fails.  
- The `correct\_value` field is updated and its properties are set correctly based on the value of `correct`.  
- The focus moves to the correct fields based on the conditions specified.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- The provided XML content does not contain any direct CRUD operations that can be executed independently of Oracle Forms constructs. Therefore, no DB queries are included.

# Validate Questionnaire Responses for Pregnancy and Alcoholism

Type: AZBJ\_QUESTIONNAIRE

Title: Validate Questionnaire Responses for Pregnancy and Alcoholism  
  
Acceptance Criteria:  
1. If the question ID is 6 and the sub-question is 5, and the correct value is null, the system should display an error message: "Husband Insurance cannot be null."  
2. If the question ID is 6 and the sub-question is 0, and the correct value is 'Y', the system should set the pregnancy status to 'Y'.  
3. If the question ID is 6 and the sub-question is 0, and the correct value is 'N', the system should set the pregnancy status to 'N'.  
4. If the question ID is 6 and the sub-question is 0, and the correct value is neither 'N' nor 'Y', the system should set the pregnancy status to 'I'.  
5. If the question ID is 6 and the sub-question is 1, and the pregnancy status is 'I', the system should display an error message: "Please check pregnancy status."  
6. If the question ID is 6 and the sub-question is 1, and the pregnancy status is 'N', and the correct value is not null, the system should display an error message: "She is not pregnant... Invalid Entry."  
7. If the question ID is 6 and the sub-question is 1, and the pregnancy status is 'Y', and the correct value is not between 1 and 12, the system should display an error message: "The Pregnancy months can be between 1 and 12."  
8. If the question ID is 6 and the sub-question is 1, and the pregnancy status is 'Y', and the correct value is null, the system should display an error message: "Please enter the month of pregnancy."  
9. If the question ID is 4 and the sub-question is 2, and the correct value is 'Y', the system should set the alcoholism status to 'Y'.  
10. If the question ID is 4 and the sub-question is 2, and the correct value is 'N', the system should set the alcoholism status to 'N'.  
11. If the question ID is 4 and the sub-question is 2, and the correct value is neither 'N' nor 'Y', the system should set the alcoholism status to 'I'.  
12. If the question ID is 4 and the sub-question is 3, 4, or 5, and the alcoholism status is 'I', the system should display an error message: "Please check Alcoholism Question."  
13. If the question ID is 4 and the sub-question is 3, 4, or 5, and the alcoholism status is 'N', and the correct value is not null, the system should display an error message: "The person does not drink alcohol... Invalid answer."  
14. If the question ID is 4 and the sub-question is 3, 4, or 5, and the alcoholism status is 'Y', and the correct value is null, the system should display an error message: "Please give in details of the drinking habits."  
  
Definition of Done:  
- The system should validate the questionnaire responses as per the acceptance criteria.  
- Appropriate error messages should be displayed when validation fails.  
- The pregnancy and alcoholism statuses should be set correctly based on the responses.  
- The system should navigate to the next record or item as required after validation.  
- All validations and navigations should be tested and verified for accuracy.

# Validate and Process Questionnaire Responses

Type: AZBJ\_QUESTIONNAIRE

Title: Validate and Process Questionnaire Responses  
  
Acceptance Criteria:  
1. If the question ID is 6 and the sub-question is 5, and the correct value is null, the system should display an error message: "Husband Insurance cannot be null."  
2. If the question ID is 6 and the sub-question is 0, and the correct value is 'Y', the system should set a variable indicating pregnancy status to 'Y'.  
3. If the question ID is 6 and the sub-question is 0, and the correct value is 'N', the system should set a variable indicating pregnancy status to 'N'.  
4. If the question ID is 6 and the sub-question is 0, and the correct value is neither 'N' nor 'Y', the system should set a variable indicating pregnancy status to 'I'.  
5. If the question ID is 6 and the sub-question is 1, and the pregnancy status variable is 'I', the system should display an error message: "Please check pregnancy status."  
6. If the question ID is 6 and the sub-question is 1, and the pregnancy status variable is 'N', and the correct value is not null, the system should display an error message: "She is not pregnant... Invalid Entry."  
7. If the question ID is 6 and the sub-question is 1, and the pregnancy status variable is 'Y', and the correct value is not between 1 and 12, the system should display an error message: "The Pregnancy months can be between 1 and 12."  
8. If the question ID is 6 and the sub-question is 1, and the pregnancy status variable is 'Y', and the correct value is null, the system should display an error message: "Please enter the month of pregnancy."  
9. If the question ID is 4 and the sub-question is 2, and the correct value is 'Y', the system should set a variable indicating alcohol consumption status to 'Y'.  
10. If the question ID is 4 and the sub-question is 2, and the correct value is 'N', the system should set a variable indicating alcohol consumption status to 'N'.  
11. If the question ID is 4 and the sub-question is 2, and the correct value is neither 'N' nor 'Y', the system should set a variable indicating alcohol consumption status to 'I'.  
12. If the question ID is 4 and the sub-question is 3, 4, or 5, and the alcohol consumption status variable is 'I', the system should display an error message: "Please check Alcoholism Question."  
13. If the question ID is 4 and the sub-question is 3, 4, or 5, and the alcohol consumption status variable is 'N', and the correct value is not null, the system should display an error message: "The person does not drink alcohol... Invalid answer."  
14. If the question ID is 4 and the sub-question is 3, 4, or 5, and the alcohol consumption status variable is 'Y', and the correct value is null, the system should display an error message: "Please give in details of the drinking habits."  
  
Definition of Done:  
- The system correctly validates and processes responses based on the predefined criteria.  
- Error messages are displayed as specified when validation fails.  
- Variables for pregnancy and alcohol consumption status are set correctly based on the responses.  
- The system navigates to the next item or record as required after processing the responses.  
- All acceptance criteria are met and tested successfully.

# Manage Family Details

Type: AZBJ\_FAMILY\_DETAILS1

Title: Manage Family Details  
  
Acceptance Criteria:  
1. The system should allow the user to input and update family member details such as health status, age, and cause of death.  
2. The system should display a list of family members with their respective details.  
3. The system should ensure that the input fields for health status, age, and cause of death are validated for correct data types and constraints.  
4. The system should provide a help button to assist users with any queries related to the family details section.  
  
Definition of Done:  
1. The user can successfully add, update, and view family member details.  
2. All input fields are validated for correct data types and constraints.  
3. The help button is functional and provides relevant assistance to the user.  
4. The family details section is integrated and tested within the overall system.  
  
SQL query for reference (if mentioned in XML then without Oracle Forms terminology):  
- Not applicable as the provided XML content does not include specific database queries or table references.

# Automatic Health Status Update

Type: AZBJ\_FAMILY\_DETAILS1

Title: Automatic Health Status Update  
  
Acceptance Criteria:  
1. When the user enters 'G' in the health status field, the system should automatically update it to 'GOOD'.  
2. The system should perform this update both when the user moves to the next field and when the health status field is validated.  
3. The system should navigate to the next relevant field after updating the health status.  
  
Definition of Done:  
1. The health status field is updated to 'GOOD' when 'G' is entered.  
2. The update occurs during both field validation and when moving to the next field.  
3. The system successfully navigates to the next relevant field after the update.  
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 any direct database CRUD operations that can be executed independently of Oracle Forms constructs.

# Automatic Image Loading for Family Details Section

Type: AZBJ\_FAMILY\_DETAILS1

Title: Automatic Image Loading for Family Details Section  
  
Acceptance Criteria:  
1. The system should determine the current item and block context.  
2. The system should set the visual attributes for the current item based on the block context.  
3. The system should identify the correct image file name based on the current item, block context, and proposal type.  
4. The system should load and display the image in the designated area if the image file is found.  
5. If the image file is not found, the system should handle the exception gracefully and log the error.  
  
Definition of Done:  
- The system correctly identifies the current item and block context.  
- The visual attributes for the current item are set appropriately.  
- The correct image file name is identified and loaded based on the current item, block context, and proposal type.  
- The image is displayed in the designated area without errors.  
- Any errors in loading the image are logged, and the system continues to function without crashing.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- SELECT MAX(strip\_no) INTO v\_strip\_no FROM azbj\_strip\_field\_mapping a WHERE block\_name = v\_current\_block AND field\_name = v\_current\_item AND proposal\_type = PK\_VARS.v\_proposal\_type AND EXISTS (SELECT 1 FROM azbj\_proposal\_stripes b WHERE a.proposal\_type = b.proposal\_type AND INSTR(stripe\_string, '~' || a.strip\_no || '~') > 0);

# Validate and Pre-load Age Field in Family Details

Type: AZBJ\_FAMILY\_DETAILS1

Title: Validate and Pre-load Age Field in Family Details  
  
Acceptance Criteria:  
1. The age field should not accept negative values.  
2. The age field should be pre-loaded with data from an image if available.  
3. The age field should be visible only when necessary.  
4. The age field should be editable and allow updates.  
  
Definition of Done:  
1. The age field is validated to ensure no negative values are entered.  
2. The age field is pre-loaded with data from an image if available.  
3. The age field is only visible when necessary.  
4. The age field is editable and allows updates.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- The query to fetch the maximum strip number for the current block and item:  
 ```sql  
 SELECT MAX(strip\_no)  
 INTO v\_strip\_no  
 FROM azbj\_strip\_field\_mapping a  
 WHERE block\_name = v\_current\_block  
 AND field\_name = v\_current\_item  
 AND proposal\_type = PK\_VARS.v\_proposal\_type  
 AND EXISTS  
 (SELECT 1  
 FROM azbj\_proposal\_stripes b  
 WHERE a.proposal\_type = b.proposal\_type  
 AND INSTR(stripe\_string, '~' || a.strip\_no || '~') > 0);  
 ```

# User Help for Entering Family Details

Type: AZBJ\_FAMILY\_DETAILS1

Title: User Help for Entering Family Details  
  
Acceptance Criteria:  
1. The user should be able to use a shortcut for entering family details.  
2. The format for entering family details should be clearly defined:  
 - Values should be separated by commas (",").  
 - Member strings should be separated by semicolons (";").  
3. The format should include the following details:  
 - V1: Family member (e.g., M for Mother, F for Father, etc.).  
 - V2: Health status (e.g., G for Good, D for Died).  
 - V3: Age (if the person is deceased, this is the age at death; multiple ages for the same relation can be separated by a hyphen, e.g., "24-27").  
 - V4: Life status (e.g., L for LA, P for PH).  
 - V5: Cause of death (e.g., D for Diabetes, C for Cancer, etc.).  
4. A sample string should be provided to illustrate the format: "F,G,56,L;SI,G,42,P;M,D,54,L,ND;B,G,24-26,P".  
5. The user should be able to access this help information easily.  
  
Definition of Done:  
- The help information is accessible to the user.  
- The format for entering family details is clearly defined and understandable.  
- A sample string is provided to illustrate the format.  
- The user can use the defined format to enter family details efficiently.  
  
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 Correct Population and Update of 'Cause of Death' Field

Type: AZBJ\_FAMILY\_DETAILS1

Detailed description: As a user, I want to ensure that the 'Cause of Death' field in the family details section is correctly populated and updated based on the information provided in the form, so that the data integrity is maintained and the information is accurately reflected.  
  
Acceptance criteria:  
1. When the user navigates to the 'Cause of Death' field, the system should load the relevant data from the image associated with the current item.  
2. If the user moves to the next item, the system should:  
 - Check if the 'FM\_IP\_TYPE' field in the family details is not null.  
 - If not null, navigate to the last record and then to the next record.  
 - Update the family details fields with the corresponding values from the current block.  
 - Clear the values in the current block after updating the family details.  
3. The system should apply specific visual attributes to the current item based on the block it belongs to.  
4. The system should determine the appropriate image file to load based on the proposal type and other conditions, and display the image in the designated area.  
  
Definition of Done:  
- The 'Cause of Death' field is correctly populated and updated as per the specified logic.  
- The system loads and displays the correct image based on the current item and proposal type.  
- The visual attributes are applied correctly to the current item.  
- The family details fields are updated and cleared as specified when navigating to the next item.  
- All functionalities are tested and verified to ensure data integrity and accuracy.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- No direct database queries are provided in the XML content that can be executed without modification. The logic involves complex procedures and conditions that are specific to Oracle Forms and cannot be directly translated into standalone SQL queries.

# Capture and Display Age at Death for Family Members

Type: AZBJ\_FAMILY\_DETAILS1

Title: Capture and Display Age at Death for Family Members  
  
Acceptance Criteria:  
1. The system should load the field value from an image when the user navigates to the age at death field.  
2. When the user presses the key to move to the next item, the system should automatically navigate to the cause of death field.  
3. The system should apply specific visual attributes to the field based on the current block context.  
4. The system should determine the correct image file to load based on the proposal type and other conditions, and display the image accordingly.  
5. The system should handle any errors gracefully and log them for debugging purposes.  
  
Definition of Done:  
1. The age at death field is correctly populated from the image.  
2. Navigation to the next field (cause of death) works seamlessly.  
3. Visual attributes are applied correctly based on the block context.  
4. The correct image file is loaded and displayed based on the proposal type and other conditions.  
5. Error handling and logging are implemented and tested.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- The system should query the `azbj\_strip\_field\_mapping` and `azbj\_proposal\_stripes` tables to determine the correct strip number for the image file.  
- Example query:  
 ```sql  
 SELECT MAX(strip\_no)  
 INTO v\_strip\_no  
 FROM azbj\_strip\_field\_mapping a  
 WHERE block\_name = v\_current\_block  
 AND field\_name = v\_current\_item  
 AND proposal\_type = PK\_VARS.v\_proposal\_type  
 AND EXISTS  
 (SELECT 1  
 FROM azbj\_proposal\_stripes b  
 WHERE a.proposal\_type = b.proposal\_type  
 AND INSTR(stripe\_string, '~' || a.strip\_no || '~') > 0);  
 ```

# Load and Display Images Based on Current Item and Block Context

Type: AZBJ\_FAMILY\_DETAILS1

Title: Load and Display Images Based on Current Item and Block Context  
  
Acceptance Criteria:  
1. The system should load the image based on the current item and block context.  
2. The system should set the visual attributes for the current item based on the block it belongs to.  
3. The system should determine the correct image file name and load it accordingly.  
4. The system should handle different blocks and items, setting the appropriate visual attributes and loading the correct image.  
5. The system should handle exceptions and log errors appropriately.  
  
Definition of Done:  
1. The image is loaded correctly based on the current item and block context.  
2. The visual attributes for the current item are set correctly.  
3. The correct image file name is determined and loaded.  
4. The system handles different blocks and items appropriately.  
5. Exceptions are handled, and errors are logged.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- The system should query the `azbj\_strip\_field\_mapping` and `azbj\_proposal\_stripes` tables to determine the correct `strip\_no` for the image to be loaded.  
  
```sql  
SELECT MAX(strip\_no)  
INTO v\_strip\_no  
FROM azbj\_strip\_field\_mapping a  
WHERE block\_name = v\_current\_block  
 AND field\_name = v\_current\_item  
 AND proposal\_type = PK\_VARS.v\_proposal\_type  
 AND EXISTS (  
 SELECT 1  
 FROM azbj\_proposal\_stripes b  
 WHERE a.proposal\_type = b.proposal\_type  
 AND INSTR(stripe\_string, '~' || a.strip\_no || '~') > 0  
 );  
```  
  
- The system should also handle the case where the `strip\_no` is not found and set it to `NULL`.  
  
```sql  
EXCEPTION  
 WHEN OTHERS THEN  
 v\_strip\_no := NULL;  
```  
  
- The system should log errors using the `azbj\_pk1\_acc.azbj\_debug` procedure.  
  
```sql  
EXCEPTION  
 WHEN OTHERS THEN  
 azbj\_pk1\_acc.azbj\_debug(:correct\_value.appln\_no, 'SQLERRM : ' || SQLERRM);  
 NULL;  
```  
  
This user story ensures that the system correctly loads and displays images based on the current context, handles different blocks and items appropriately, and logs errors when necessary.

# Automatic Loading and Display of Family Member Details

Type: AZBJ\_FAMILY\_DETAILS1

Title: Automatic Loading and Display of Family Member Details  
  
Acceptance Criteria:  
1. When the "Family Member" field is selected, the system should trigger a procedure to load the relevant data from an image file.  
2. The system should determine the current block and item, and apply the appropriate visual attributes based on predefined conditions.  
3. The system should identify the correct image file based on the proposal type and other parameters, and display it in the designated area.  
4. If the current block is related to family details, the system should set specific strip numbers and load the corresponding image.  
5. The system should handle exceptions gracefully, logging any errors encountered during the process.  
  
Definition of Done:  
- The "Family Member" field correctly triggers the data loading procedure.  
- Visual attributes are applied as per the conditions defined in the procedure.  
- The correct image file is identified and displayed based on the context.  
- The system handles exceptions without crashing and logs errors appropriately.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- The procedure includes a query to fetch the maximum strip number from the `azbj\_strip\_field\_mapping` table based on the current block, item, and proposal type:  
 ```sql  
 SELECT MAX(strip\_no)  
 INTO v\_strip\_no  
 FROM azbj\_strip\_field\_mapping a  
 WHERE block\_name = v\_current\_block  
 AND field\_name = v\_current\_item  
 AND proposal\_type = PK\_VARS.v\_proposal\_type  
 AND EXISTS  
 (SELECT 1  
 FROM azbj\_proposal\_stripes b  
 WHERE a.proposal\_type = b.proposal\_type  
 AND INSTR(stripe\_string, '~' || a.strip\_no || '~') > 0);  
 ```

# Manage Family Details Section

Type: AZBJ\_FAMILY\_DETAILS1

Title: Manage Family Details Section  
  
Acceptance Criteria:  
1. When the user navigates to the family details section, the system should load the relevant data from the image file based on the current item and block.  
2. The system should decode and clear the family details field when the user moves to the next item.  
3. The system should apply specific visual attributes to the current record and correct text fields based on the block name.  
4. The system should determine the appropriate image file to load based on the proposal type and the current item, and display the image accordingly.  
5. The system should handle exceptions gracefully, logging any errors encountered during the process.  
  
Definition of Done:  
1. The family details section is accessible and functional within the application.  
2. Data is correctly loaded and displayed based on the current item and block.  
3. Visual attributes are applied as specified.  
4. Image files are correctly determined and displayed based on the proposal type and current item.  
5. All exceptions are handled, and errors are logged appropriately.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- The system should query the `azbj\_strip\_field\_mapping` and `azbj\_proposal\_stripes` tables to determine the appropriate strip number for the current item and proposal type.  
- Example query:  
 ```sql  
 SELECT MAX(strip\_no)  
 INTO v\_strip\_no  
 FROM azbj\_strip\_field\_mapping a  
 WHERE block\_name = v\_current\_block  
 AND field\_name = v\_current\_item  
 AND proposal\_type = PK\_VARS.v\_proposal\_type  
 AND EXISTS  
 (SELECT 1  
 FROM azbj\_proposal\_stripes b  
 WHERE a.proposal\_type = b.proposal\_type  
 AND INSTR(stripe\_string, '~' || a.strip\_no || '~') > 0);  
 ```

# Manage Family Details

Type: PH\_FAMILY\_DETAILS1

Title: Manage Family Details  
  
Acceptance Criteria:  
1. The user should be able to view and update family details, including health status, member number, and other related information.  
2. The user should be able to insert new family details.  
3. The user should be able to view a list of family members and their details.  
4. The user should be able to use a help button to get assistance with the form.  
5. The user should be able to select from predefined lists for certain fields, such as family member type and cause of death.  
  
Definition of Done:  
1. The user interface should allow for the viewing, updating, and insertion of family details.  
2. The help button should be functional and provide relevant assistance.  
3. Predefined lists should be available for selection where applicable.  
4. All fields should be validated according to their data types and constraints.  
5. The form should be user-friendly and intuitive.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Not applicable as the provided XML content does not include specific database queries.

# Validate Age Field in Family Details

Type: PH\_FAMILY\_DETAILS1

Title: Validate Age Field in Family Details  
  
Acceptance Criteria:  
1. The age field should only accept numeric values.  
2. The age field should not accept negative values.  
3. If a negative value is entered, an error message should be displayed indicating that the age cannot be less than zero.  
  
Definition of Done:  
1. The age field is implemented and visible in the family details section.  
2. The age field accepts only numeric values.  
3. Validation logic is in place to check for negative values.  
4. An error message is displayed when a negative value is entered.  
5. The feature is tested and verified to work 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 related to CRUD operations for this user story.

# Ensure Correct Population and Update of 'Cause of Death' Field

Type: PH\_FAMILY\_DETAILS1

Title: Ensure Correct Population and Update of 'Cause of Death' Field  
  
Acceptance Criteria:  
1. When navigating to the 'Cause of Death' field, the system should load the relevant data from the image associated with the current item.  
2. Upon pressing the key to move to the next item, the system should:  
 - Navigate to the family details block.  
 - If the family member type is not null, move to the last record and then to the next record.  
 - Update the family details fields with the corresponding values from the current family details.  
 - Clear the current family details fields.  
3. When a new item instance is created, the system should load the relevant data from the image associated with the current item.  
  
Definition of Done:  
- The 'Cause of Death' field is correctly populated and updated based on the corresponding fields.  
- The system loads the relevant data from the image when navigating to the 'Cause of Death' field.  
- The family details fields are updated and cleared as specified when moving to the next item.  
- The functionality is tested and verified to ensure data integrity and accuracy.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- No specific DB queries provided in the XML content for CRUD operations.

# Automatic Health Status Update and Navigation

Type: PH\_FAMILY\_DETAILS1

Title: Automatic Health Status Update and Navigation  
  
Acceptance Criteria:  
1. When the user enters 'G' in the health status field, the system should automatically change it to 'GOOD'.  
2. Upon validation of the health status field, if the value is 'G', it should be updated to 'GOOD'.  
3. When the user presses the key to move to the next item, if the health status is 'G', it should be updated to 'GOOD' and the focus should move to the next field for age.  
  
Definition of Done:  
- The health status field should automatically update from 'G' to 'GOOD' during validation and navigation.  
- The system should navigate to the next field for age after updating the health status.  
- The functionality should be tested and verified to ensure it works as expected without any Oracle Forms-specific dependencies.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- No specific CRUD operations mentioned in the provided XML content.

# Automatic Loading of Family Member Details

Type: PH\_FAMILY\_DETAILS1

Title: Automatic Loading of Family Member Details  
  
Acceptance Criteria:  
1. When the user interacts with the family member field, the system should trigger a procedure to load the relevant data.  
2. The procedure should determine the current block and item, and based on predefined conditions, set the visual attributes for the item.  
3. The procedure should identify the correct block and item names based on the current context.  
4. The system should fetch the appropriate image file name based on the proposal type and other conditions.  
5. The system should handle exceptions gracefully, ensuring that any errors are logged for debugging purposes.  
  
Definition of Done:  
1. The family member field should correctly trigger the data loading procedure when interacted with.  
2. The procedure should accurately determine and set the visual attributes for the item.  
3. The correct image file should be fetched and displayed based on the context and conditions.  
4. All exceptions should be logged, and the system should continue to function without crashing.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- The procedure includes a query to fetch the maximum strip number from the `azbj\_strip\_field\_mapping` table based on certain conditions:  
 ```sql  
 SELECT MAX(strip\_no)  
 INTO v\_strip\_no  
 FROM azbj\_strip\_field\_mapping a  
 WHERE block\_name = v\_current\_block  
 AND field\_name = v\_current\_item  
 AND proposal\_type = PK\_VARS.v\_proposal\_type  
 AND EXISTS  
 (SELECT 1  
 FROM azbj\_proposal\_stripes b  
 WHERE a.proposal\_type = b.proposal\_type  
 AND INSTR(stripe\_string, '~' || a.strip\_no || '~') > 0);  
 ```

# Manage Family Details Section

Type: PH\_FAMILY\_DETAILS1

Title: Manage Family Details Section  
  
Acceptance Criteria:  
1. When the user navigates to the family details section, the system should load the relevant data from the image file based on the current item and block.  
2. The system should decode and clear the family details field when the user moves to the next item.  
3. The system should apply specific visual attributes to the current record and correct text fields based on the block and item properties.  
4. The system should determine the appropriate image file to load based on the proposal type and the number of pages.  
5. The system should handle exceptions gracefully and log any errors encountered during the process.  
  
Definition of Done:  
- The family details section is functional and meets the acceptance criteria.  
- The system correctly loads and displays data based on user interactions.  
- Visual attributes are applied as specified.  
- The appropriate image file is loaded based on the conditions.  
- Error handling and logging are implemented.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- SELECT MAX(strip\_no) INTO v\_strip\_no FROM azbj\_strip\_field\_mapping a WHERE block\_name = v\_current\_block AND field\_name = v\_current\_item AND proposal\_type = PK\_VARS.v\_proposal\_type AND EXISTS (SELECT 1 FROM azbj\_proposal\_stripes b WHERE a.proposal\_type = b.proposal\_type AND INSTR(stripe\_string, '~' || a.strip\_no || '~') > 0);

# Ensure MEMBER\_NO1 Field is Correctly Populated and Validated

Type: PH\_FAMILY\_DETAILS1

Detailed description: As a user, I want to ensure that the MEMBER\_NO1 field in the family details section is correctly populated and validated, so that the data integrity and accuracy are maintained.  
  
Acceptance criteria:  
1. The MEMBER\_NO1 field should load data from an image when the field is accessed.  
2. The MEMBER\_NO1 field should be validated to ensure that the age entered is not less than zero.  
3. The MEMBER\_NO1 field should be displayed in uppercase and should be a numeric field with a maximum length of 3 characters.  
4. The MEMBER\_NO1 field should have specific visual attributes based on the current block and item properties.  
5. The MEMBER\_NO1 field should correctly handle different proposal types and load the appropriate image based on the proposal type and other conditions.  
  
Definition of Done:  
1. The MEMBER\_NO1 field loads data from an image when accessed.  
2. The MEMBER\_NO1 field validation ensures that the age entered is not less than zero.  
3. The MEMBER\_NO1 field is displayed in uppercase and is a numeric field with a maximum length of 3 characters.  
4. The MEMBER\_NO1 field has the correct visual attributes based on the current block and item properties.  
5. The MEMBER\_NO1 field correctly handles different proposal types and loads the appropriate image based on the proposal type and other conditions.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- The following query is used to retrieve the maximum strip number for the current block and item:  
 ```sql  
 SELECT MAX(strip\_no)  
 INTO v\_strip\_no  
 FROM azbj\_strip\_field\_mapping a  
 WHERE block\_name = v\_current\_block  
 AND field\_name = v\_current\_item  
 AND proposal\_type = PK\_VARS.v\_proposal\_type  
 AND EXISTS  
 (SELECT 1  
 FROM azbj\_proposal\_stripes b  
 WHERE a.proposal\_type = b.proposal\_type  
 AND INSTR(stripe\_string, '~' || a.strip\_no || '~') > 0);  
 ```

# Input Age at Death for Family Member

Type: PH\_FAMILY\_DETAILS1

Title: Input Age at Death for Family Member  
  
Acceptance Criteria:  
1. The field should accept only numeric values.  
2. The field should have a maximum length of 3 digits.  
3. The field should be pre-populated with data from an image if available.  
4. Upon entering data, the focus should automatically move to the next field, which is for the cause of death.  
5. The field should be visible only when required.  
6. The field should be editable and allow updates even if initially null.  
  
Definition of Done:  
1. The field for entering the age at which a family member died is implemented and functional.  
2. The field accepts only numeric values and has a maximum length of 3 digits.  
3. The field is pre-populated with data from an image if available.  
4. The focus automatically moves to the next field upon data entry.  
5. The field is visible only when required.  
6. The field is editable and allows updates even if initially null.  
7. All acceptance criteria are met and tested.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- The following query is used to fetch the maximum strip number for a given block and field name, considering the proposal type:  
 ```sql  
 SELECT MAX(strip\_no)  
 INTO v\_strip\_no  
 FROM azbj\_strip\_field\_mapping a  
 WHERE block\_name = v\_current\_block  
 AND field\_name = v\_current\_item  
 AND proposal\_type = PK\_VARS.v\_proposal\_type  
 AND EXISTS  
 (SELECT 1  
 FROM azbj\_proposal\_stripes b  
 WHERE a.proposal\_type = b.proposal\_type  
 AND INSTR(stripe\_string, '~' || a.strip\_no || '~') > 0);  
 ```

# Help Message for Entering Family Details

Type: PH\_FAMILY\_DETAILS1

Title: Help Message for Entering Family Details  
  
Acceptance Criteria:  
1. The user should be able to see a help message that explains the format for entering family details.  
2. The help message should include the following information:  
 - The format for entering family details using a string with values separated by commas and members separated by semicolons.  
 - Examples of the string format.  
 - Short codes for family members (e.g., M for Mother, F for Father).  
 - Health status codes (e.g., G for Good, D for Died).  
 - Age format, including how to indicate age at death and multiple ages for the same relation.  
 - Life status codes (e.g., L for LA, P for PH).  
 - Cause of death codes (e.g., D for Diabetes, C for Cancer).  
  
Definition of Done:  
1. The help message is displayed when the user requests help.  
2. The help message is clear, concise, and includes all necessary information for entering family details.  
3. The help message is accessible and visible 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.

# Automatic Image Loading Based on Current Item and Block Context

Type: PH\_FAMILY\_DETAILS1

Detailed description: As a user, I want the system to automatically load and display the appropriate image based on the current item and block context when interacting with family details, so that I can easily view and verify the relevant information.  
  
Acceptance criteria:  
1. The system should load the image corresponding to the current item and block context when the item is selected or changed.  
2. The system should handle different blocks and items, setting visual attributes appropriately based on the block context.  
3. The system should determine the correct image file name based on the proposal type and number of pages.  
4. The system should handle exceptions gracefully, logging any errors encountered during the image loading process.  
  
Definition of Done:  
1. The image loading functionality is implemented and tested.  
2. The system correctly identifies and loads the appropriate image for the current item and block context.  
3. Visual attributes are set correctly based on the block context.  
4. The system handles exceptions and logs errors as expected.  
5. The functionality is verified through user acceptance testing.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- The following query is used to retrieve the maximum strip number for the current block and item:  
 ```sql  
 SELECT MAX(strip\_no)  
 INTO v\_strip\_no  
 FROM azbj\_strip\_field\_mapping a  
 WHERE block\_name = v\_current\_block  
 AND field\_name = v\_current\_item  
 AND proposal\_type = PK\_VARS.v\_proposal\_type  
 AND EXISTS  
 (SELECT 1  
 FROM azbj\_proposal\_stripes b  
 WHERE a.proposal\_type = b.proposal\_type  
 AND INSTR(stripe\_string, '~' || a.strip\_no || '~') > 0);  
 ```

# View and Manage Error Codes and Descriptions

Type: RESULT

Title: View and Manage Error Codes and Descriptions  
  
Acceptance Criteria:  
1. The section should display a list of error codes and their corresponding descriptions.  
2. The error codes should be displayed in a column labeled "ERR CODE".  
3. The error descriptions should be displayed in a column labeled "ERR DESCRIPTION".  
4. There should be a hidden field for additional error descriptions that is not visible to the user.  
5. The section should include a button labeled "Exit" that allows the user to exit the section.  
6. Clicking on any item that is not a button should navigate to that item.  
  
Definition of Done:  
1. The section is implemented and displays error codes and descriptions as specified.  
2. The hidden field for additional error descriptions is present but not visible to the user.  
3. The "Exit" button is functional and allows the user to exit the section.  
4. Navigation to items within the section works as specified.  
5. 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.

# Double-click Navigation for Result Item

Type: RESULT

Title: Double-click Navigation for Result Item  
  
Acceptance Criteria:  
1. When the result item is not null and the result type is either 'E' or 'W':  
 - If the form question is 'BK\_ALL\_VALIDATIONS', the system should navigate to the 'POPULATE' item in the rider details section.  
 - Otherwise, the system should navigate to the item specified by the form question.  
2. The system should handle the navigation logic without any errors or interruptions.  
  
Definition of Done:  
- The double-click functionality on the result item should be implemented and tested.  
- The navigation to the appropriate form question or rider details should work as specified in the acceptance criteria.  
- The feature should be free of any Oracle Forms-specific terminology and should be independent of the underlying technology.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- No specific database queries are mentioned in the provided XML content.

# Navigation based on Result field content and type

Type: RESULT

Title: Navigation based on Result field content and type  
  
Acceptance Criteria:  
1. When the "Result" field is double-clicked:  
 - If the "Result" field is not empty and the result type is either 'E' or 'W':  
 - If the form question is 'BK\_ALL\_VALIDATIONS', navigate to the 'Populate' item in the 'Rider Details' section.  
 - Otherwise, navigate to the item specified by the form question.  
2. When the "Enter" key is pressed while the "Result" field is focused:  
 - If the "Result" field is not empty and the result type is 'E':  
 - If the form question is 'BK\_ALL\_VALIDATIONS', navigate to the 'Populate' item in the 'Rider Details' section.  
 - Otherwise, navigate to the item specified by the form question.  
  
Definition of Done:  
- The navigation logic is implemented and tested.  
- The "Result" field correctly triggers navigation based on the specified conditions.  
- The functionality is verified to work in both double-click and Enter key scenarios.  
- 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 direct database queries.

# Exit Button Functionality

Type: RESULT

Title: Exit Button Functionality  
  
Acceptance Criteria:  
1. When the exit button is pressed, the system should navigate to the first record in the result set.  
2. If the result field is not null and the result type is 'E', the system should display the PDF and image windows.  
3. If the result field is null or the result type is not 'E', the system should still display the PDF and image windows.  
  
Definition of Done:  
- The exit button functionality is implemented and tested.  
- The system correctly navigates to the first record in the result set upon pressing the exit button.  
- The PDF and image windows are displayed based on the specified conditions.  
- 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.

# Manage Family Member Details

Type: PH\_FAMILY\_DETAILS

Title: Manage Family Member Details  
  
Acceptance Criteria:  
1. The user should be able to add new family member details, including the type of member, name, age, health status, age at death, and cause of death.  
2. The user should be able to update existing family member details.  
3. The user should be able to delete a family member's details using a delete button.  
4. The form should display a list of family members with their respective details.  
5. The form should include a dropdown list for selecting the type of family member.  
6. The form should include text fields for entering the age, health status, age at death, and cause of death.  
7. The form should have a close button to exit the family details section.  
  
Definition of Done:  
1. The form allows the user to add, update, and delete family member details.  
2. The form displays all family member details in a structured manner.  
3. The dropdown list for family member type is functional and displays the correct options.  
4. Text fields for age, health status, age at death, and cause of death are functional and accept valid inputs.  
5. The delete button removes the selected family member's details from the form.  
6. The close button exits the family details section without errors.  
7. All functionalities are tested and verified to work as expected.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- No specific DB queries provided in the XML content.

# Delete Family Member Record

Type: PH\_FAMILY\_DETAILS

Title: Delete Family Member Record  
  
Acceptance Criteria:  
- When the "Delete" button is pressed, the selected family member's record should be removed from the family details section.  
- The system should confirm the deletion action before removing the record to prevent accidental deletions.  
  
Definition of Done:  
- The "Delete" button is visible and enabled in the family details section.  
- Pressing the "Delete" button triggers the deletion of the selected family member's record.  
- A confirmation prompt appears before the record is deleted.  
- The record is successfully removed from the family details section upon confirmation.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include specific database queries.

# Input and Validate Health Status

Type: PH\_FAMILY\_DETAILS

Title: Input and Validate Health Status  
  
Acceptance Criteria:  
1. When the user inputs 'G' in the health status field, the system should automatically update the field to 'GOOD'.  
2. The system should navigate to the next field (age died) after the health status is validated.  
3. The health status field should be pre-loaded with data from an image if available.  
  
Definition of Done:  
1. The health status field accepts user input and updates to 'GOOD' if 'G' is entered.  
2. The system successfully navigates to the next field after validation.  
3. The health status field is pre-loaded with data from an image if applicable.  
4. All functionalities are tested and verified to work as expected.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- No direct database queries are provided in the XML content for CRUD operations.

# Input and Validate Age in Family Details Section

Type: PH\_FAMILY\_DETAILS

Detailed description: As a user, I want to input and validate the age of a family member in the family details section of the proposal form, ensuring that the age entered is a positive number and that the field is pre-populated with relevant data from an image if available.  
  
Acceptance criteria:  
1. The age field should allow the user to input a numeric value up to 3 digits.  
2. The age field should be pre-populated with data from an image if available.  
3. The age field should not accept negative values.  
4. The age field should be editable and allow updates.  
5. The age field should be displayed with a white background and black text.  
6. The age field should be positioned correctly within the family details section.  
  
Definition of Done:  
1. The age field is implemented and allows numeric input up to 3 digits.  
2. The age field is pre-populated with data from an image if available.  
3. Validation is in place to ensure the age entered is not negative.  
4. The age field is editable and updates are allowed.  
5. The age field is displayed with the specified visual properties.  
6. The age field is correctly positioned within the family details section.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- No specific DB queries provided in the XML content for CRUD operations.

# Implement 'Close' Button Functionality in Family Details Section

Type: PH\_FAMILY\_DETAILS

Title: Implement 'Close' Button Functionality in Family Details Section  
  
Acceptance Criteria:  
1. The 'Close' button should be labeled as 'Close' and should be positioned at the specified coordinates within the family details section.  
2. The button should be disabled and hidden by default.  
3. When the button is pressed, it should:  
 - Navigate to the item 'CH\_BOOKING\_FREQUENCY'.  
 - Perform a null operation.  
 - Navigate to the block 'BBPROCESS'.  
 - Hide the window named 'RISK'.  
  
Definition of Done:  
- The 'Close' button is correctly labeled and positioned.  
- The button is disabled and hidden by default.  
- The button performs the specified actions when pressed.  
- All actions are 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 specific database CRUD operations.

# Input and Update Cause of Death for Family Members

Type: PH\_FAMILY\_DETAILS

Title: Input and Update Cause of Death for Family Members  
  
Acceptance Criteria:  
1. When the user navigates to the "Cause of Death" field, the system should load data from an image file and display it.  
2. Upon attempting to move to the next item, the system should prompt the user with a question: "Do you wish to add more members?".  
 - If the user selects "Yes", the system should navigate to the next tab.  
 - If the user selects "No", the system should move to the next record.  
3. The system should handle different blocks and items dynamically, ensuring the correct image file is loaded based on the current context.  
4. The system should apply specific visual attributes to the current item based on predefined conditions.  
  
Definition of Done:  
- The "Cause of Death" field is functional and allows for data input and updates.  
- The prompt for adding more members appears and functions as expected.  
- Data from the image file is correctly loaded and displayed based on the current item.  
- Visual attributes are applied correctly to the current item.  
- All functionalities are tested and verified to work as intended.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- No direct database queries are provided in the XML content that can be executed without modification.

# Dynamic Loading and Display of Family Member Details

Type: PH\_FAMILY\_DETAILS

Detailed description: As a user, I want the system to dynamically load and display the appropriate family member details based on the current context and user actions, ensuring that the correct visual attributes and data are presented.  
  
Acceptance criteria:  
1. The system should dynamically determine the current context (block and item) and set the appropriate visual attributes for the item.  
2. The system should load the correct image file based on the determined context and proposal type.  
3. The system should handle different blocks and items, such as family details, previous policies, rider details, and fund details, and set the appropriate visual attributes and data accordingly.  
4. The system should handle exceptions gracefully and log any errors encountered during the process.  
  
Definition of Done:  
1. The system correctly identifies the current context and sets the appropriate visual attributes for the item.  
2. The correct image file is loaded and displayed based on the determined context and proposal type.  
3. The system handles different blocks and items appropriately, setting the correct visual attributes and data.  
4. All exceptions are handled gracefully, and errors are logged for further investigation.  
5. The functionality 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 determine the maximum strip number for a given block, field, and proposal type:  
 ```sql  
 SELECT MAX(strip\_no)  
 INTO v\_strip\_no  
 FROM azbj\_strip\_field\_mapping a  
 WHERE block\_name = v\_current\_block  
 AND field\_name = v\_current\_item  
 AND proposal\_type = PK\_VARS.v\_proposal\_type  
 AND EXISTS  
 (SELECT 1  
 FROM azbj\_proposal\_stripes b  
 WHERE a.proposal\_type = b.proposal\_type  
 AND INSTR(stripe\_string, '~' || a.strip\_no || '~') > 0);  
 ```

# Member Number Field Validation and Population

Type: PH\_FAMILY\_DETAILS

Title: Member Number Field Validation and Population  
  
Acceptance Criteria:  
1. The "Member Number" field should be left-justified, have a maximum length of 3 characters, and be displayed in uppercase.  
2. The field should be hidden by default and only become visible when necessary.  
3. The field should be populated with data from an image when the user navigates to it.  
4. The field should validate that the age entered is not less than zero (this validation is currently commented out but should be considered for future implementation).  
5. The field should be associated with the correct visual attributes based on the current block and item context.  
6. The field should load the appropriate image file based on the proposal type and the number of pages.  
7. The field should handle exceptions gracefully and log any errors encountered during the process.  
  
Definition of Done:  
- The "Member Number" field meets all the specified acceptance criteria.  
- The field is tested to ensure it behaves correctly in different contexts and scenarios.  
- Any errors or exceptions are logged appropriately for debugging purposes.  
- The implementation is reviewed and approved by the relevant stakeholders.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- The following query is used to retrieve the maximum strip number for the current block and item:  
 ```sql  
 SELECT MAX(strip\_no)  
 INTO v\_strip\_no  
 FROM azbj\_strip\_field\_mapping a  
 WHERE block\_name = v\_current\_block  
 AND field\_name = v\_current\_item  
 AND proposal\_type = PK\_VARS.v\_proposal\_type  
 AND EXISTS  
 (SELECT 1  
 FROM azbj\_proposal\_stripes b  
 WHERE a.proposal\_type = b.proposal\_type  
 AND INSTR(stripe\_string, '~' || a.strip\_no || '~') > 0);  
 ```

# Automatic Image Loading for Age Died Field

Type: PH\_FAMILY\_DETAILS

Title: Automatic Image Loading for Age Died Field  
  
Acceptance Criteria:  
1. When the user navigates to the "Age Died" field, the system should determine the current block and item.  
2. The system should set the visual attributes for the current item based on the block context.  
3. The system should identify the correct image file name based on the proposal type and other conditions.  
4. The system should load and display the image in the designated area if the image file is found.  
5. If the image file is not found or an error occurs, the system should handle the exception gracefully and log the error for debugging purposes.  
  
Definition of Done:  
1. The system correctly identifies the current block and item when the user navigates to the "Age Died" field.  
2. The visual attributes for the current item are set appropriately based on the block context.  
3. The correct image file name is identified and loaded based on the proposal type and other conditions.  
4. The image is displayed in the designated area without errors.  
5. Any errors encountered during the process are logged, and the system continues to function without crashing.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- The following query is used to determine the maximum strip number for the current block and item:  
 ```sql  
 SELECT MAX(strip\_no)  
 INTO v\_strip\_no  
 FROM azbj\_strip\_field\_mapping a  
 WHERE block\_name = v\_current\_block  
 AND field\_name = v\_current\_item  
 AND proposal\_type = PK\_VARS.v\_proposal\_type  
 AND EXISTS  
 (SELECT 1  
 FROM azbj\_proposal\_stripes b  
 WHERE a.proposal\_type = b.proposal\_type  
 AND INSTR(stripe\_string, '~' || a.strip\_no || '~') > 0);  
 ```

# Select Insurance Policy Type and Load Field from Image

Type: PH\_FAMILY\_DETAILS

Title: Select Insurance Policy Type and Load Field from Image  
  
Acceptance Criteria:  
1. The list of insurance policy types should include the following options: "IP", "PH", "8", "10", and "SL".  
2. When a new item instance is created, the system should load the field from the image based on the current item.  
3. When the list selection changes, the system should load the field from the image based on the current item.  
4. The system should handle different blocks and items to set visual attributes and load the appropriate image based on the selected policy type and other conditions.  
5. The system should determine the correct image file name based on the proposal type and the selected policy type, and display the image accordingly.  
  
Definition of Done:  
1. The user can see and select from a list of predefined insurance policy types.  
2. The system correctly loads and displays the field from the image when a new item instance is created or when the list selection changes.  
3. The system sets the appropriate visual attributes for the current item based on the block and item conditions.  
4. The correct image file is loaded and displayed based on the selected policy type and other conditions.  
5. All acceptance criteria are met, and the functionality is tested and verified.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- The following query is used to determine the maximum strip number for the current block and item:  
 ```sql  
 SELECT MAX(strip\_no)  
 INTO v\_strip\_no  
 FROM azbj\_strip\_field\_mapping a  
 WHERE block\_name = v\_current\_block  
 AND field\_name = v\_current\_item  
 AND proposal\_type = PK\_VARS.v\_proposal\_type  
 AND EXISTS  
 (SELECT 1  
 FROM azbj\_proposal\_stripes b  
 WHERE a.proposal\_type = b.proposal\_type  
 AND INSTR(stripe\_string, '~' || a.strip\_no || '~') > 0);  
 ```

# Handle Verification and Deletion in Checker Module

Type: AZBJ\_UL\_CHECKER

Detailed description: As a user, I want to ensure that the system correctly handles the verification and deletion of records related to the "Checker" module, so that data integrity is maintained when managing related records.  
  
Acceptance criteria:  
1. When a record in the "Checker" module is being populated, if the verification number is not null, the system should query and populate related details from the "Partners" and "Address" modules.  
2. When attempting to delete a record in the "Checker" module, the system should check for existing related records in the "Partners" and "Address" modules. If related records exist, the system should prevent the deletion and display an appropriate message.  
  
Definition of Done:  
- The system correctly populates related details from the "Partners" and "Address" modules when the verification number is present.  
- The system prevents the deletion of a "Checker" record if related records exist in the "Partners" or "Address" modules and displays a message indicating the reason for the prevention.  
- All functionalities are tested and verified to ensure data integrity and proper error handling.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Query to check for related records in the "Partners" module:  
 ```sql  
 SELECT 1 FROM azbj\_ul\_chk\_partners a WHERE a.verification\_no = :verification\_no;  
 ```  
- Query to check for related records in the "Address" module:  
 ```sql  
 SELECT 1 FROM azbj\_ul\_chk\_address a WHERE a.verification\_no = :verification\_no;  
 ```

# Auto-populate Mailing Address based on Residence Address Checkbox

Type: AZBJ\_UL\_CHECKER

Detailed description: As a user, I want to ensure that the mailing address can be automatically populated with the same details as the residence address when a checkbox is selected, so that I can save time and avoid manual entry errors.  
  
Acceptance criteria:  
1. When the checkbox labeled "Same as Residence Address" is checked:  
 - The system should prompt the user with a confirmation message: "This action will overwrite Mailing Address also, please select YES to overwrite else select NO to keep existing!"  
 - If the user selects "YES":  
 - The mailing address fields should be populated with the corresponding residence address details.  
 - The mailing address fields should be disabled to prevent further editing.  
 - The focus should move to the "Mobile Phone" field.  
 - If the user selects "NO":  
 - The mailing address fields should remain unchanged.  
2. When the checkbox labeled "Same as Residence Address" is unchecked:  
 - The mailing address fields should be cleared.  
 - The mailing address fields should be enabled for editing.  
 - The focus should move to the "PIN" field.  
  
Definition of Done:  
- The checkbox functionality is implemented and tested.  
- The confirmation message is displayed correctly.  
- The mailing address fields are populated and disabled/enabled as per the user's selection.  
- The focus moves to the appropriate fields based on the checkbox state.  
- All changes are saved and persist correctly in the database.  
  
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.

# Update Policy Holder Details Based on Insured Person Checkbox

Type: AZBJ\_UL\_CHECKER

Title: Update Policy Holder Details Based on Insured Person Checkbox  
  
Acceptance Criteria:  
1. If the insured person's age is less than 18 and the checkbox is checked, display an error message stating that the life assured cannot be equal to the policy holder for minors.  
2. If the proposal type is "Joint Life" and the checkbox is checked, display an error message stating that the life assured cannot be equal to the policy holder for joint life proposals.  
3. If the checkbox is checked and the spouse is not present, update the policy holder's details to match the insured person's details, including personal information, address, and contact details.  
4. If the checkbox is unchecked, clear the policy holder's details and enable the fields for manual entry.  
5. Ensure that the appropriate fields and tabs are enabled or disabled based on the checkbox status and other conditions such as product code and partner type.  
  
Definition of Done:  
- The system correctly updates the policy holder's details to match the insured person's details when the checkbox is checked.  
- Error messages are displayed as per the acceptance criteria when conditions are not met.  
- Fields and tabs are enabled or disabled appropriately based on the checkbox status and other conditions.  
- The functionality is tested and verified to ensure it works as expected 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 direct database queries.

# Auto-update Mailing Address based on Residence Address

Type: AZBJ\_UL\_CHECKER

Title: Auto-update Mailing Address based on Residence Address  
  
Acceptance Criteria:  
1. When the "Same as Residence Address" checkbox is checked:  
 - The system should prompt the user with a confirmation message: "This action will overwrite Mailing Address also, please select YES to overwrite else select NO to keep existing."  
 - If the user selects "YES":  
 - The system should copy the residence address fields to the corresponding mailing address fields.  
 - The copied fields should be disabled to prevent further editing.  
 - If the phone number starts with '09' and is 11 digits long, the system should set the mobile number field to the phone number without the leading '0'.  
 - If the user selects "NO":  
 - No changes should be made to the mailing address fields.  
2. When the "Same as Residence Address" checkbox is unchecked:  
 - The system should clear the mailing address fields.  
 - The cleared fields should be enabled for editing.  
  
Definition of Done:  
- The functionality should be implemented and tested.  
- The user should be able to see the confirmation message when checking the checkbox.  
- The mailing address fields should be updated or cleared based on the user's selection.  
- The fields should be enabled or disabled as specified.  
- The changes should be saved and reflected in the database.  
  
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.

# Auto-populate Policy Holder's Address based on 'Same as LA' Checkbox

Type: AZBJ\_UL\_CHECKER

Title: Auto-populate Policy Holder's Address based on 'Same as LA' Checkbox  
  
Acceptance Criteria:  
1. When the 'Same as LA' checkbox is checked:  
 - The policy holder's country name should be set to the insured's country name.  
 - The policy holder's door number should be set to the insured's door number.  
 - The policy holder's building name should be set to the insured's building name.  
 - The policy holder's plot number should be set to the insured's plot number.  
 - The policy holder's pin code should be set to the insured's pin code.  
 - The policy holder's address line 4 should be set to the insured's address line 4.  
 - The policy holder's address line 5 should be set to the insured's address line 5.  
 - The policy holder's area should be set to the insured's area.  
 - The policy holder's area free text should be set to the insured's area free text.  
 - The policy holder's address line 3 should be set to the insured's address line 3.  
 - The policy holder's place free text should be set to the insured's place free text.  
 - The policy holder's residence proof should be set to the insured's residence proof.  
 - The policy holder's mobile door number should be set to the insured's mobile door number.  
 - The policy holder's mobile building name should be set to the insured's mobile building name.  
 - The policy holder's mobile plot number should be set to the insured's mobile plot number.  
 - The policy holder's mobile area should be set to the insured's mobile area.  
 - The policy holder's mobile area free text should be set to the insured's mobile area free text.  
 - The policy holder's mobile address line 3 should be set to the insured's mobile address line 3.  
 - The policy holder's mobile place free text should be set to the insured's mobile place free text.  
 - The policy holder's mobile address line 4 should be set to the insured's mobile address line 4.  
 - The policy holder's mobile address line 5 should be set to the insured's mobile address line 5.  
 - The policy holder's mobile pin code should be set to the insured's mobile pin code.  
 - The policy holder's mobile country name should be set to the insured's mobile country name.  
 - The policy holder's mobile phone should be set to the insured's mobile phone.  
 - The policy holder's email should be set to the insured's email.  
 - The policy holder's mobile residence proof should be set to the insured's mobile residence proof.  
 - If the 'CHK\_SAME' checkbox is checked, the 'CHK\_SAME\_PHRES' checkbox should also be checked.  
  
2. When the 'Same as LA' checkbox is unchecked:  
 - The policy holder's country name should be cleared.  
 - The policy holder's door number should be cleared.  
 - The policy holder's building name should be cleared.  
 - The policy holder's plot number should be cleared.  
 - The policy holder's pin code should be cleared.  
 - The policy holder's address line 4 should be cleared.  
 - The policy holder's address line 5 should be cleared.  
 - The policy holder's area should be cleared.  
 - The policy holder's area free text should be cleared.  
 - The policy holder's address line 3 should be cleared.  
 - The policy holder's place free text should be cleared.  
 - The policy holder's residence proof should be cleared.  
 - The policy holder's mobile door number should be cleared.  
 - The policy holder's mobile building name should be cleared.  
 - The policy holder's mobile plot number should be cleared.  
 - The policy holder's mobile area should be cleared.  
 - The policy holder's mobile area free text should be cleared.  
 - The policy holder's mobile address line 3 should be cleared.  
 - The policy holder's mobile place free text should be cleared.  
 - The policy holder's mobile address line 4 should be cleared.  
 - The policy holder's mobile address line 5 should be cleared.  
 - The policy holder's mobile pin code should be cleared.  
 - The policy holder's mobile country name should be cleared.  
 - The policy holder's mobile phone should be cleared.  
 - The policy holder's email should be cleared.  
 - The policy holder's mobile residence proof should be cleared.  
 - The 'CHK\_SAME\_PHRES' checkbox should be unchecked.  
  
Definition of Done:  
- The functionality should be implemented and tested.  
- The system should correctly populate or clear the policy holder's address details based on the state of the 'Same as LA' checkbox.  
- All acceptance criteria should be met.  
- The changes should be reviewed and approved by the stakeholders.

# User Interaction with Proposal Form Controls

Type: PROPOSALFORM

Title: User Interaction with Proposal Form Controls  
  
Acceptance Criteria:  
1. The form should have buttons for navigating (left, right, up, down) and these buttons should be clearly labeled and functional.  
2. The form should include buttons for zooming in and out, and resetting the zoom level.  
3. There should be a button to skip the application process.  
4. The form should have buttons to view individual images and all images.  
5. The form should include a button to validate the proposal data.  
6. The form should have a save button that is initially disabled and becomes enabled when there are changes to save.  
7. The form should include a button to exit the application.  
8. The form should have a checkbox to populate DE1 records.  
9. The form should handle errors gracefully, suppressing specific error messages and setting the system message level appropriately.  
  
Definition of Done:  
1. All buttons and controls are implemented and functional as per the acceptance criteria.  
2. The form layout is user-friendly and visually consistent.  
3. Error handling is implemented as specified, ensuring a smooth user experience.  
4. The form is tested for usability and functionality, ensuring all buttons perform their intended actions.  
5. The form is free of Oracle Forms-specific terminology and can be understood independently of the underlying technology.  
  
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.

# Image Navigation with Left Button

Type: PROPOSALFORM

Title: Image Navigation with Left Button  
  
Acceptance Criteria:  
1. When the user presses the "Left" button, the horizontal position parameter (`para\_x`) should decrease by 20 units.  
2. The image should scroll to the new position based on the updated `para\_x` value.  
  
Definition of Done:  
- The "Left" button is functional and decreases the horizontal position parameter by 20 units when pressed.  
- The image scrolls correctly to the new position after the button is pressed.  
- 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.

# Zoom Functionality for Image in Proposal Form

Type: PROPOSALFORM

Title: Zoom Functionality for Image in Proposal Form  
  
Acceptance Criteria:  
1. When the "Zoom" button is pressed, the system should call a function to zoom into the image identified by 'image51' using the zoom percentage and a correct value factor.  
2. The system should then navigate to the current item in focus.  
  
Definition of Done:  
- The "Zoom" button is visible and functional on the proposal form.  
- Pressing the "Zoom" button successfully zooms into the specified image.  
- The system navigates back to the current item after zooming.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include any database CRUD operations.

# Implement Left Button Functionality

Type: PROPOSALFORM

Title: Implement Left Button Functionality  
  
Acceptance Criteria:  
1. When the "Left" button is pressed, the system should call a function to set the adjacent stripe number.  
2. The function should take the proposal type and the previous stripe number as inputs and determine the new stripe number accordingly.  
3. The button should be visually styled with a gray background, centered text, and bold font.  
  
Definition of Done:  
1. The "Left" button is present on the proposal form.  
2. Pressing the "Left" button successfully triggers the function to set the adjacent stripe number.  
3. The function correctly updates the stripe number based on the provided inputs.  
4. The button's visual appearance matches the specified design.  
  
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.

# Display Previous Stripe Image on Button Press

Type: PROPOSALFORM

Title: Display Previous Stripe Image on Button Press  
  
Acceptance Criteria:  
1. When the "Current Stripe" button is pressed, the system should display the previous stripe image associated with the given application number.  
2. The cursor style should be set to 'DEFAULT' after the image is displayed.  
  
Definition of Done:  
1. The "Current Stripe" button is functional and displays the previous stripe image correctly.  
2. The cursor style is reset to 'DEFAULT' after the image is displayed.  
3. The feature is tested and verified to work as expected.  
4. 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 any direct database CRUD operations.

# Trigger Adjacent Strip Number on Bottom Button Press

Type: PROPOSALFORM

Title: Trigger Adjacent Strip Number on Bottom Button Press  
  
Acceptance Criteria:  
1. When the "Bottom" button is pressed, the system should call a function to set the adjacent strip number.  
2. The function should take the proposal type and the previous stripe number as inputs and set the adjacent strip number accordingly.  
3. The function should handle the 'B' type strip specifically.  
  
Definition of Done:  
1. The "Bottom" button is visible and clickable on the proposal form.  
2. Pressing the "Bottom" button successfully triggers the function to set the adjacent strip number.  
3. The function correctly processes the proposal type and previous stripe number to set the adjacent strip number.  
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 Scroll Up Button Functionality

Type: PROPOSALFORM

Title: Image Scroll Up Button Functionality  
  
Acceptance Criteria:  
1. When the "Up" button is pressed, the vertical position of the image should decrease by 20 units.  
2. The image should scroll upwards by 20 units each time the button is pressed.  
  
Definition of Done:  
- The "Up" button is functional and correctly scrolls the image upwards by 20 units.  
- The image scrolls smoothly without any glitches or delays.  
- The button is labeled "Up" and is easily accessible 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 CRUD operations.

# Scroll Image Vertically with Button Press

Type: PROPOSALFORM

Title: Scroll Image Vertically with Button Press  
  
Acceptance Criteria:  
1. When the button labeled "Down" is pressed, the vertical position of the image should increase by 20 units.  
2. The image should scroll down by 20 units each time the button is pressed.  
  
Definition of Done:  
1. The button labeled "Down" is present and functional.  
2. Pressing the button results in the image scrolling down by 20 units.  
3. The image scrolls smoothly and consistently with each button press.  
4. The feature is tested and verified to work as expected.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable for this user story.

# Implement 'Top' Button Functionality

Type: PROPOSALFORM

Title: Implement 'Top' Button Functionality  
  
Acceptance Criteria:  
1. When the 'Top' button is pressed, the system should call the function `Get\_N\_Set\_Adjacent\_strip\_no` with the following parameters:  
 - `v\_proposal\_type` from the `PK\_VARS` package.  
 - `v\_previous\_stripe\_no` from the `azbj\_new\_qc\_val` package.  
 - A constant value 'T'.  
2. The function should correctly set the adjacent strip number based on the provided parameters.  
  
Definition of Done:  
- The 'Top' button is visible and functional on the proposal form.  
- Pressing the 'Top' button triggers the `Get\_N\_Set\_Adjacent\_strip\_no` function with the correct parameters.  
- The adjacent strip number is set correctly based on the proposal type and previous stripe number.  
  
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.

# Image Horizontal Scroll on Button Press

Type: PROPOSALFORM

Title: Image Horizontal Scroll on Button Press  
  
Acceptance Criteria:  
1. When the button labeled "Right" is pressed, the horizontal position parameter (`para\_x`) should increase by 20 units.  
2. The image should scroll horizontally based on the updated `para\_x` value.  
3. The vertical position parameter (`para\_y`) should remain unchanged during this operation.  
  
Definition of Done:  
- The button is functional and correctly updates the horizontal position parameter.  
- The image scrolls horizontally as expected when the button is pressed.  
- The feature is tested and verified to work without any errors or unexpected behavior.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include any database CRUD operations.

# Retrieve and Set Adjacent Strip Number on Right Button Press

Type: PROPOSALFORM

Title: Retrieve and Set Adjacent Strip Number on Right Button Press  
  
Acceptance Criteria:  
1. When the "Right" button is pressed, the system should call a function to get and set the adjacent strip number.  
2. The function should take three parameters: proposal type, previous stripe number, and a direction indicator ('R' for right).  
3. The function should update the adjacent strip number based on the provided parameters.  
  
Definition of Done:  
1. The "Right" button is visible and functional on the proposal form.  
2. Pressing the "Right" button successfully triggers the function to get and set the adjacent strip number.  
3. The adjacent strip number is correctly updated based on the proposal type and previous stripe number.  
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 CRUD operations on database tables.

# Skip Application with Reason

Type: PROPOSALFORM

Detailed description: As a user, I want to be able to skip an application by providing a reason, so that the system can handle the application accordingly and perform necessary cleanup actions.  
  
Acceptance criteria:  
1. If the reason for skipping the application is not provided, the system should display an error message: "Please Select Reason For Skipping Application".  
2. The system should set a global approval flag to 'N' and call a specific form to handle error applications.  
3. The system should delete any associated images if they exist.  
4. If the approval flag is set to 'Y', the system should remove associated jobs and exit the form.  
5. If the user ID starts with 'UUT', the system should remove associated jobs and exit the form.  
  
Definition of Done:  
- The user can successfully skip an application by providing a reason.  
- The system displays an appropriate error message if the reason is not provided.  
- The system performs necessary cleanup actions, including deleting images and removing jobs.  
- The form exits correctly after the actions are performed.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- The system should query the `AZBJ\_STRIP\_QUESTIONS\_MAPPING` table to get the strip number based on the proposal type, question ID, and sub-question ID.  
- The system should query the `azbj\_bbu\_questions` and `azbj\_fcf\_questionnaire` tables to populate questions based on the partner number and application number.

# Reset Zoom and Navigate to Application Number Field

Type: PROPOSALFORM

Title: Reset Zoom and Navigate to Application Number Field  
  
Acceptance Criteria:  
1. When the reset button is pressed, the zoom level of the image should be reset to its default state.  
2. The system should automatically navigate to the application number field after the reset button is pressed.  
  
Definition of Done:  
- The reset button should be functional and visible on the form.  
- Pressing the reset button should clear any zoom adjustments made to the image.  
- The focus should move to the application number field after the reset action is performed.  
- 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.

# Exit Button Functionality

Type: PROPOSALFORM

Detailed description: As a user, I want to be able to exit the proposal form and ensure that all related data is properly cleaned up and any necessary updates are made to the database, so that the system remains consistent and no unnecessary data is left behind.  
  
Acceptance criteria:  
1. When the exit button is pressed, the system should delete records from the `azbj\_cp\_merge` table where the `REQUEST\_ID` matches the `REQUEST\_ID` in the `azbj\_cp\_merge\_request` table for the given application number.  
2. The system should delete records from the `azbj\_cp\_merge\_request` table where the `APPLN\_NO` matches the given application number.  
3. The system should update the `azbj\_phub\_tracker` table to set the `locking\_flag` to 'N', update the `proposal\_modif\_date` to the current date, and set the `proposal\_modif\_user` to the current user where the `application\_no` matches the given application number.  
4. If the `call\_form\_name` parameter is not 'NEW\_BBU', the system should delete images from the specified directory for the given application number.  
5. The system should set the application property for the date format to the global date format.  
6. The form should exit without validation.  
  
Definition of Done:  
- The exit button functionality is implemented and tested.  
- All specified database operations are performed correctly.  
- The system handles exceptions gracefully and provides appropriate error messages.  
- The form exits as expected without any validation errors.  
  
DB queries for Table reference CRUD operations only(With Usage):  
```sql  
-- Delete from azbj\_cp\_merge where REQUEST\_ID matches the given application number  
DELETE FROM azbj\_cp\_merge   
WHERE REQUEST\_ID IN (  
 SELECT request\_id   
 FROM azbj\_cp\_merge\_request   
 WHERE APPLN\_NO = TO\_NUMBER(:correct\_value.appln\_no)  
);  
  
-- Delete from azbj\_cp\_merge\_request where APPLN\_NO matches the given application number  
DELETE FROM azbj\_cp\_merge\_request   
WHERE APPLN\_NO = TO\_NUMBER(:correct\_value.appln\_no);  
  
-- Update azbj\_phub\_tracker to set locking\_flag to 'N' and update modification details  
UPDATE azbj\_phub\_tracker  
SET locking\_flag = 'N',  
 proposal\_modif\_date = SYSDATE,  
 proposal\_modif\_user = USER  
WHERE application\_no = NVL(TO\_CHAR(:correct\_value.appln\_no), :correct\_value.appln\_no);  
```

# View Image Functionality

Type: PROPOSALFORM

Detailed description: As a user, I want to be able to view images associated with a proposal, including those in PDF format, so that I can easily access and review the visual content related to the proposal.  
  
Acceptance criteria:  
1. When the "View Image" button is pressed, the system should determine the correct file path for the image based on the form name.  
2. If the form name is 'NEW\_BBU', the image path should be adjusted from a server path to a local path.  
3. The system should transfer the image file from the server to the client machine.  
4. If the image file is a PDF, it should be opened using the default PDF viewer.  
5. If the image file is not a PDF, it should be opened using the default image viewer.  
6. The system should handle both local file paths and URLs for image files.  
7. The system should hide the view canvas after the image is displayed.  
  
Definition of Done:  
- The "View Image" button functionality is implemented and tested.  
- The system correctly handles different file paths and formats.  
- The image or PDF is displayed using the appropriate viewer.  
- The view canvas is hidden after the image is displayed.  
- 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 CRUD operations.

# View All Images Functionality

Type: PROPOSALFORM

Detailed description: As a user, I want to be able to view all images associated with a specific application number in a temporary directory on my local machine, so that I can easily access and review them.  
  
Acceptance criteria:  
1. When the "View All Images" button is pressed, a temporary directory should be created on the local machine using the application number.  
2. The system should navigate to the image details section and iterate through all records.  
3. For each record, if an image path is available, the image should be copied from the server to the local temporary directory.  
4. If the image is in PDF format, it should be opened using the default PDF viewer.  
5. If the image is in any other format, it should be opened using the default image viewer.  
6. The process should handle any exceptions gracefully without crashing.  
  
Definition of Done:  
- The "View All Images" button successfully creates a temporary directory on the local machine.  
- All images associated with the application number are copied to the temporary directory.  
- Images in PDF format are opened with the default PDF viewer.  
- Images in other formats are opened with the default image viewer.  
- The process completes without any unhandled exceptions.  
  
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.

# Automatic Population of Records in Proposal Form

Type: PROPOSALFORM

Title: Automatic Population of Records in Proposal Form  
  
Acceptance Criteria:  
- When the "Populate" button is pressed, the system should check the value of the `POPULATE\_DE1\_RECORDS` field.  
- If the value of `POPULATE\_DE1\_RECORDS` is 'Y', the system should call the function to populate records with the parameter 'D1'.  
- If the value of `POPULATE\_DE1\_RECORDS` is not 'Y', the system should call the function to populate records with the parameter 'D2'.  
  
Definition of Done:  
- The "Populate" button is visible and functional on the proposal form.  
- The system correctly checks the value of the `POPULATE\_DE1\_RECORDS` field when the button is pressed.  
- The appropriate function is called with the correct parameter based on the value of `POPULATE\_DE1\_RECORDS`.  
- The records are populated as expected based on the function call.

# Automatically Populate DE2 Records Based on Checkbox State in Proposal Form

Type: PROPOSALFORM

Title: Automatically Populate DE2 Records Based on Checkbox State in Proposal Form  
  
Description:   
As a user, I want the system to automatically populate DE2 records based on the state of a checkbox in the proposal form, so that I can ensure the correct data is populated without manual intervention.  
  
Acceptance Criteria:  
1. When the checkbox labeled "DE1/ Desktop" is checked, the system should automatically call a function to populate DE2 records with the parameter 'D1'.  
2. When the checkbox labeled "DE1/ Desktop" is unchecked, the system should automatically call a function to populate DE2 records with the parameter 'D2'.  
  
Definition of Done:  
1. The checkbox should be visible and functional within the proposal form.  
2. The system should correctly call the function to populate DE2 records with the appropriate parameter based on the checkbox state.  
3. The functionality should be tested and verified to ensure that DE2 records are populated correctly when the checkbox state changes.  
  
Block Name: PROPOSALFORM

# Save Proposal Form Data with Validations

Type: PROPOSALFORM

Detailed description: As a user, I want to save the proposal form data after performing necessary validations and checks to ensure data integrity and compliance with business rules.  
  
Acceptance criteria:  
1. When the save button is pressed, the system should check if the product group is 'N'. If true, it should call the main validation procedure.  
2. If any error occurs during validation, the save button should be disabled, and an error message should be displayed indicating that changes were made after validation and re-validation is required.  
3. If the insured checkbox is unchecked and the spouse is not present, the system should navigate to the insured checkbox and execute the related trigger.  
4. If the product group is 'Y', the system should disable non-group items and validate the save for group products.  
5. If the product group is not 'Y', the system should navigate to the quality control questions block and iterate through the records to check specific conditions related to question ID 80.  
6. If the mandatory rule for income tax declaration is 'Y' and no related records are found, an error message should be displayed prompting the user to check the declaration under Income Tax Rules 2015.  
7. Finally, the system should call the main save procedure to save the data.  
  
Definition of Done:  
- The save functionality should be implemented as per the acceptance criteria.  
- All validations and checks should be performed correctly.  
- Appropriate error messages should be displayed when validation fails.  
- The data should be saved successfully if all validations pass.  
  
DB queries for Table reference CRUD operations only (With Usage):  
```sql  
SELECT COUNT(contract\_id)  
 INTO v\_cnt\_itrule  
 FROM AZBJ\_ITRULE\_ANSWER\_DTLS  
 WHERE application\_no = :correct\_value.appln\_no;  
```

# Validate Proposal Form Data

Type: PROPOSALFORM

User Story: Validate Proposal Form Data  
  
Detailed Description:  
As a user, I want to validate the proposal form data to ensure that all necessary information is correctly entered and meets the required conditions before submission.  
  
Acceptance Criteria:  
1. When the "Validate" button is pressed, the system should:  
 - Log the validation attempt with relevant details.  
 - Check if the application number exists in the system.  
 - Verify if the solution name is selected and matches the product code.  
 - Retrieve and validate credit card details if the renewal payment method is 'CCSI'.  
 - Retrieve and validate bank account details if the renewal payment method is 'EMAND'.  
 - Enable or disable PAN issuance date fields based on the age proof type.  
 - Ensure that the insured's age proof matches the policyholder's age proof if applicable.  
 - Enable or disable the date of birth field based on the e-KYC status.  
 - Validate fund details if the product is unit-linked.  
 - Check if the solution name is correctly selected for the given product code.  
 - Validate rider details and set the entry age and partner type based on the rider cover code and other conditions.  
 - Enable or disable nominee details based on the product code and insured status.  
 - Ensure that the payout date is correctly calculated based on the policy's backdated status and payout frequency.  
 - Log any errors or warnings encountered during the validation process.  
  
Definition of Done:  
- The validation process should be triggered by pressing the "Validate" button.  
- All specified checks and validations should be performed.  
- Relevant logs should be created for each validation attempt.  
- Errors and warnings should be displayed to the user as appropriate.  
- The form should be ready for submission if all validations pass.  
  
DB Queries for Table Reference CRUD Operations:  
- Retrieve application details from `azbj\_solution\_appln\_bases` using the application number.  
- Retrieve solution configuration from `azbj\_solution\_config\_mst` using the solution ID.  
- Retrieve credit card details from `azbj\_ccsi\_registration` using the application number.  
- Retrieve bank account details from `AZBJ\_NACH\_REGISTRATION` and `AZBJ\_NACH\_REGISTRATION\_DET` using the application number and registration ID.  
- Retrieve fund details from `carequote\_bi` using the request ID or transaction ID.  
  
Please review the user story and provide feedback.

# Manage Family Details in Proposal Form

Type: AZBJ\_FAMILY\_DETAILS

Title: Manage Family Details in Proposal Form  
  
Acceptance Criteria:  
1. The user should be able to add new family members with details such as type, name, age, health status, age at death, and cause of death.  
2. The user should be able to update existing family member details.  
3. The user should be able to delete a family member from the list.  
4. The form should display a list of family members with their details.  
5. The form should have a button to delete a selected family member.  
6. The form should have a dropdown list to select the type of family member.  
7. The form should have text fields for entering the age, health status, age at death, and cause of death of the family member.  
8. The form should have a close button to exit the family details section.  
  
Definition of Done:  
1. The user can successfully add, update, and delete family member details.  
2. The form displays all family members with their respective details.  
3. The delete button removes the selected family member from the list.  
4. The dropdown list for family member type is functional and displays the correct options.  
5. The text fields for age, health status, age at death, and cause of death are functional and accept valid inputs.  
6. The close button exits the family details section without errors.  
7. 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 database queries.

# Delete Family Member

Type: AZBJ\_FAMILY\_DETAILS

Title: Delete Family Member  
  
Acceptance Criteria:  
- When the "Delete" button is pressed, the system should delete the selected family member's record from the family details section.  
  
Definition of Done:  
- The "Delete" button is visible and enabled on the family details section.  
- Pressing the "Delete" button successfully removes the selected family member's record.  
- The user receives a confirmation that the record has been deleted.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- No specific DB queries provided in the XML content.

# Implement 'Close' button in family details section

Type: AZBJ\_FAMILY\_DETAILS

Title: Implement 'Close' button in family details section  
  
Acceptance Criteria:  
- The button should be labeled 'Close'.  
- The button should be located at the specified position within the family details section.  
- The button should be disabled and not visible by default.  
- When the button is pressed, it should trigger the logic to close the current window.  
  
Definition of Done:  
- The 'Close' button is implemented in the family details section.  
- The button is disabled and not visible by default.  
- Pressing the button successfully closes the current window.  
- The button's position and appearance match the specified requirements.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include any database CRUD operations.

# Input and Update Cause of Death for Family Members

Type: AZBJ\_FAMILY\_DETAILS

Title: Input and Update Cause of Death for Family Members  
  
Acceptance Criteria:  
1. When the user navigates to the "Cause of Death" field, the system should load the relevant data from an image file based on the current item.  
2. When the user attempts to move to the next item, an alert should prompt the user with the question "Do you wish to add more members?".  
 - If the user selects "Yes", the system should navigate to the "Insured" checkbox.  
 - If the user selects "No", the system should move to the next record.  
3. The system should handle the loading of images based on the current block and item, and apply visual attributes to the current record.  
4. The system should determine the appropriate image file to load based on the proposal type and other conditions, and display the image in the designated area.  
  
Definition of Done:  
- The "Cause of Death" field can be successfully navigated to and from, with the appropriate data loaded from images.  
- The alert prompt functions correctly, allowing the user to either add more members or move to the next record.  
- Images are loaded and displayed correctly based on the current block and item, with the correct visual attributes applied.  
- The system handles all specified conditions and exceptions without errors.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- No direct database queries are provided in the XML content that can be executed without modification.

# Health Status Input and Validation

Type: AZBJ\_FAMILY\_DETAILS

Detailed description: As a user, I want to input and validate the health status of a family member in the system, so that the health status is automatically updated to "GOOD" if the input is 'G'.  
  
Acceptance criteria:  
1. When the user inputs 'G' in the health status field, the system should automatically update the field to 'GOOD'.  
2. The health status field should be validated upon moving to the next field and upon validation of the current field.  
3. The system should navigate to the next relevant field after the health status is validated.  
  
Definition of Done:  
1. The health status field accepts input and updates correctly based on the input value.  
2. The validation logic is triggered both when the field is validated and when the user moves to the next field.  
3. The system navigates to the next field after the health status is validated.  
4. The feature is tested and confirmed to work as expected without any Oracle Forms-specific dependencies.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- No direct database queries are mentioned in the provided XML content that are independent of Oracle Forms-specific constructs.

# Select Insurance Policy Type from Predefined List

Type: AZBJ\_FAMILY\_DETAILS

Title: Select Insurance Policy Type from Predefined List  
  
Acceptance Criteria:  
1. The list of insurance policy types should include the following options:  
 - IP  
 - 8  
 - 10  
 - SL  
2. When a user selects an option from the list, the system should load the corresponding field data from an image.  
3. The system should validate that the selected option is not null.  
4. The system should handle the display and formatting of the selected option based on predefined visual attributes.  
  
Definition of Done:  
1. The user can see and select from a list of predefined insurance policy types.  
2. Upon selection, the system loads the relevant field data from an image.  
3. The system validates the selected option to ensure it is not null.  
4. The selected option is displayed with the correct visual attributes.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- The system should query the `azbj\_strip\_field\_mapping` and `azbj\_proposal\_stripes` tables to determine the appropriate strip number based on the selected insurance policy type and proposal type.  
- Example query:  
 ```sql  
 SELECT MAX(strip\_no)  
 INTO v\_strip\_no  
 FROM azbj\_strip\_field\_mapping a  
 WHERE block\_name = v\_current\_block  
 AND field\_name = v\_current\_item  
 AND proposal\_type = PK\_VARS.v\_proposal\_type  
 AND EXISTS  
 (SELECT 1  
 FROM azbj\_proposal\_stripes b  
 WHERE a.proposal\_type = b.proposal\_type  
 AND INSTR(stripe\_string, '~' || a.strip\_no || '~') > 0);  
 ```

# Automatic Image Loading for Family Member Field

Type: AZBJ\_FAMILY\_DETAILS

Title: Automatic Image Loading for Family Member Field  
  
Acceptance Criteria:  
1. When the "Family Member" field is selected, the system should determine the current block and item.  
2. The system should set the visual attributes for the current item based on the block context.  
3. The system should identify the correct image file name based on the proposal type and the current item.  
4. The system should load and display the image in the designated area if the image file exists.  
5. If the image file does not exist, the system should handle the exception gracefully without crashing.  
  
Definition of Done:  
1. The system correctly identifies the current block and item when the "Family Member" field is selected.  
2. The visual attributes for the current item are set appropriately based on the block context.  
3. The correct image file name is determined and loaded based on the proposal type and current item.  
4. The image is displayed in the designated area without any errors.  
5. The system handles exceptions gracefully, ensuring no crashes occur if the image file is missing.  
  
DB queries for Table reference CRUD operations only (With Usage):  
```sql  
SELECT MAX(strip\_no)  
INTO v\_strip\_no  
FROM azbj\_strip\_field\_mapping a  
WHERE block\_name = v\_current\_block  
AND field\_name = v\_current\_item  
AND proposal\_type = PK\_VARS.v\_proposal\_type  
AND EXISTS  
(SELECT 1  
FROM azbj\_proposal\_stripes b  
WHERE a.proposal\_type = b.proposal\_type  
AND INSTR(stripe\_string, '~' || a.strip\_no || '~') > 0);  
```

# Automatic Image Loading for Family Details Section

Type: AZBJ\_FAMILY\_DETAILS

Detailed description: As a user, I want the system to automatically load and display the relevant image based on the current item in the family details section, so that I can easily view the associated image without manual intervention.  
  
Acceptance criteria:  
1. The system should determine the current block and item.  
2. If the current block is 'PROPOSALFORM' or 'CORRECT\_VALUE', the system should set the visual attribute to 'VA\_CURRENT\_RECORD'.  
3. If the current block is one of the specified blocks (e.g., 'AZBJ\_FUND\_DETAILS1', 'AZBJ\_SSO\_FUND\_DETAILS1', etc.), the system should set the visual attribute to 'VA\_CORRECT\_TEXT'.  
4. The system should determine the appropriate block and item names based on the current block.  
5. The system should fetch the maximum strip number from the 'azbj\_strip\_field\_mapping' table where the block name, field name, and proposal type match, and the strip number exists in the 'azbj\_proposal\_stripes' table.  
6. If the current item is in the 'PH\_FAMILY\_DETAILS%' block, the strip number should be set to 34.  
7. If the strip number is not null, the system should determine the file name based on the strip number and proposal type.  
8. The system should read and display the image file in the 'PROPOSALFORM.image51' item if the form version is '10.%'.  
9. If the form version is not '10.%', the system should use a different method to read and display the image file.  
  
Definition of Done:  
- The image associated with the current item in the family details section is automatically loaded and displayed.  
- The visual attributes are correctly set based on the current block and item.  
- The correct strip number is fetched and used to determine the image file name.  
- The image file is read and displayed using the appropriate method based on the form version.  
- The functionality is tested and verified to work as expected.  
  
DB queries for Table reference CRUD operations only(With Usage):  
```sql  
SELECT MAX(strip\_no)  
INTO v\_strip\_no  
FROM azbj\_strip\_field\_mapping a  
WHERE block\_name = v\_current\_block  
 AND field\_name = v\_current\_item  
 AND proposal\_type = PK\_VARS.v\_proposal\_type  
 AND EXISTS  
 (SELECT 1  
 FROM azbj\_proposal\_stripes b  
 WHERE a.proposal\_type = b.proposal\_type  
 AND INSTR(stripe\_string, '~' || a.strip\_no || '~') > 0);  
```

# Input and Validate Age in Family Details

Type: AZBJ\_FAMILY\_DETAILS

Title: Input and Validate Age in Family Details  
  
Acceptance Criteria:  
1. The age field should accept only numeric values.  
2. The age field should have a maximum length of 3 digits.  
3. The age field should be validated to ensure it is not less than zero.  
4. The age field should be pre-populated with data from an image if available.  
5. The age field should be editable and allow updates.  
  
Definition of Done:  
1. The age field is implemented and visible in the family details section.  
2. The age field accepts only numeric input and has a maximum length of 3 digits.  
3. Validation logic is in place to ensure the age is not less than zero.  
4. The age field is pre-populated with data from an image if available.  
5. The age field is editable and allows updates.  
6. All acceptance criteria are met and tested.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- The following query is used to fetch the maximum strip number for the current block and item:  
 ```sql  
 SELECT MAX(strip\_no)  
 INTO v\_strip\_no  
 FROM azbj\_strip\_field\_mapping a  
 WHERE block\_name = v\_current\_block  
 AND field\_name = v\_current\_item  
 AND proposal\_type = PK\_VARS.v\_proposal\_type  
 AND EXISTS  
 (SELECT 1  
 FROM azbj\_proposal\_stripes b  
 WHERE a.proposal\_type = b.proposal\_type  
 AND INSTR(stripe\_string, '~' || a.strip\_no || '~') > 0);  
 ```

# Automatic Image Loading for Member Number Field

Type: AZBJ\_FAMILY\_DETAILS

Detailed description: As a user, I want the system to automatically load and display the appropriate image based on the current item and block context when I navigate to the "Member Number" field in the family details section, so that I can view the relevant image without manual intervention.  
  
Acceptance criteria:  
1. When the user navigates to the "Member Number" field, the system should determine the current block and item.  
2. The system should set the visual attributes for the current item based on the block context.  
3. The system should identify the correct image file name based on the proposal type and other conditions.  
4. The system should load and display the image in the designated area if the image file is found.  
5. If the image file is not found or an error occurs, the system should handle the exception gracefully and log the error.  
  
Definition of Done:  
1. The image is automatically loaded and displayed when the user navigates to the "Member Number" field.  
2. The visual attributes for the current item are correctly set based on the block context.  
3. The correct image file name is identified and used to load the image.  
4. The system handles exceptions gracefully and logs any errors that occur during the image loading process.  
5. The functionality is tested and verified to work as expected in different scenarios.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- The following query is used to determine the maximum strip number for the current block and item:  
 ```sql  
 SELECT MAX(strip\_no)  
 INTO v\_strip\_no  
 FROM azbj\_strip\_field\_mapping a  
 WHERE block\_name = v\_current\_block  
 AND field\_name = v\_current\_item  
 AND proposal\_type = PK\_VARS.v\_proposal\_type  
 AND EXISTS  
 (SELECT 1  
 FROM azbj\_proposal\_stripes b  
 WHERE a.proposal\_type = b.proposal\_type  
 AND INSTR(stripe\_string, '~' || a.strip\_no || '~') > 0);  
 ```

# View and Manage Upload Request Details

Type: BLK\_UPLOAD\_REQ

Title: View and Manage Upload Request Details  
  
Acceptance Criteria:  
1. The system should display the following fields for each upload request:  
 - Application Number  
 - Request Code  
 - Request Description  
 - Upload Date  
 - Upload User  
 - Received Status  
 - Contact Number  
 - Passport Number  
 - PAN Number  
 - Aadhaar Number  
 - Health-related question: "Are you recently hospitalized/suffering from severe illness?"  
 - Policy-related question: "Any previous policy rejected, declined or postponed?"  
  
2. The fields should be read-only and not allow insert or update operations.  
  
3. The "Received Status" should be a list item with predefined values.  
  
4. The health and policy-related questions should be represented as checkboxes with "Yes" and "No" options.  
  
Definition of Done:  
- The user interface displays all the specified fields for each upload request.  
- All fields are read-only and cannot be modified by the user.  
- The "Received Status" field is a dropdown list with predefined values.  
- The health and policy-related questions are displayed as checkboxes with appropriate labels and options.  
- The layout and design of the fields are user-friendly and consistent with the overall application design.  
  
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.

# Implement Received Status Dropdown

Type: BLK\_UPLOAD\_REQ

Title: Implement Received Status Dropdown  
  
Acceptance Criteria:  
1. The received status should be presented as a dropdown list with the following options:  
 - Option 1: "5"  
 - Option 2: "8"  
 - Option 3: "No" (represented internally as "SL")  
2. The dropdown list should be clearly labeled as "Received" and should be visible on the form.  
3. The selected status should be saved and reflected in the database upon submission.  
  
Definition of Done:  
- The dropdown list for received status is implemented and visible on the form.  
- The dropdown list contains the specified options.  
- The selected status is saved correctly in the database.  
- The form is tested to ensure that the status updates are reflected accurately in the system.

# Manage Table Field Mapping

Type: TABLE\_FIELD\_MAPPING

Detailed description: As a user, I want to manage the mapping of table fields to their corresponding form fields, so that I can ensure data consistency and accuracy across the application.  
  
Acceptance criteria:  
1. The system should allow the user to view and edit the following fields:  
 - Column Name  
 - Field Name  
 - Block Name  
 - Lead Sub Code  
 - Sub Name  
 - Referral ID  
 - Referral Name  
 - Employee ID  
 - IC Code  
2. The system should provide buttons for "Nominee Details" and "Family Details" which are initially hidden.  
3. The system should include a checkbox for "Manual BBU" with options to set it to "Manual" or "Auto".  
4. The system should include a text item for "BBU Reason" which is not editable by the user.  
5. The system should provide a list of values (LOV) for "Referral ID" and "Referral Name" from the "Syndicate" record group.  
6. The system should provide a LOV for "BBU Reason" from the "BBU\_REASON" record group.  
  
Definition of Done:  
- The user can successfully view and edit the specified fields.  
- The "Nominee Details" and "Family Details" buttons are hidden by default.  
- The "Manual BBU" checkbox functions correctly, allowing the user to toggle between "Manual" and "Auto".  
- The "BBU Reason" field is displayed but not editable.  
- The LOVs for "Referral ID", "Referral Name", and "BBU Reason" are correctly populated from their respective record groups.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- For "Syndicate" LOV:  
 ```sql  
 SELECT branch\_code, branch\_name  
 FROM azbj\_syn\_bank\_branches  
 WHERE br\_type = (CASE WHEN :CORRECT\_VALUE.FSC\_CODE LIKE '522%' THEN 'B'  
 ELSE SUBSTR(:CORRECT\_VALUE.FSC\_CODE, 1, 3) END);  
 ```  
  
- For "BBU\_REASON" LOV:  
 ```sql  
 SELECT SYS\_DESC  
 FROM azbj\_system\_constants  
 WHERE sys\_type = 'QC' AND sys\_code = 'QC\_MANUAL\_BBU';  
 ```

# Manage Manual BBU Status

Type: TABLE\_FIELD\_MAPPING

Title: Manage Manual BBU Status  
  
Acceptance Criteria:  
1. When the user selects the "Manual BBU" checkbox, the system should prompt the user to select a reason from a predefined list.  
2. If the "Manual BBU" checkbox is unchecked, the system should clear any previously selected reason.  
  
Definition of Done:  
- The "Manual BBU" checkbox should have two states: checked (Manual) and unchecked (Auto).  
- When the checkbox is checked, a list of reasons should be displayed for the user to select from.  
- When the checkbox is unchecked, any selected reason should be cleared.  
- The list of reasons should be fetched from the database table `azbj\_system\_constants` where `sys\_type` is 'QC' and `sys\_code` is 'QC\_MANUAL\_BBU'.  
  
DB queries for Table reference CRUD operations only(With Usage):  
```sql  
SELECT SYS\_DESC   
FROM azbj\_system\_constants   
WHERE sys\_type='QC'   
AND sys\_code='QC\_MANUAL\_BBU';  
```  
  
This query is used to fetch the list of reasons for manual BBU status.

# Validate Referral Name Input Based on FSC Code and Lead Sub-Code

Type: TABLE\_FIELD\_MAPPING

Title: Validate Referral Name Input Based on FSC Code and Lead Sub-Code  
  
Acceptance Criteria:  
1. If the FSC code is not null and the lead sub-code field is enabled, the system should check if the lead sub-code is null.  
2. If the lead sub-code is null, the system should display an error message prompting the user to enter the lead sub-code.  
  
Definition of Done:  
- The system correctly validates the referral name input based on the specified conditions.  
- An error message is displayed if the lead sub-code is required but not provided.  
- The validation logic is tested and confirmed to work as expected.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- The LOV (List of Values) for the referral name should fetch data from the `azbj\_syn\_bank\_branches` table based on the FSC code. The query should be:  
 ```sql  
 SELECT branch\_code, branch\_name  
 FROM azbj\_syn\_bank\_branches  
 WHERE br\_type = (CASE   
 WHEN :CORRECT\_VALUE.FSC\_CODE LIKE '522%' THEN 'B'  
 ELSE SUBSTR(:CORRECT\_VALUE.FSC\_CODE, 1, 3)  
 END);  
 ```

# Navigation and Image Handling in Family Details Section

Type: TABLE\_FIELD\_MAPPING

Detailed description: As a user, I want to navigate through the family details section of the proposal form efficiently, so that I can view and update the necessary information based on the proposal type and number of pages.  
  
Acceptance criteria:  
1. When the user is in the 'CORRECT' block, the system should navigate to the previous item based on the mapping in the `azbj\_new\_bbu\_field\_map` table.  
2. If the user is not in the 'CORRECT' block, the system should navigate to the item prefixed with 'L\_'.  
3. When the button is pressed, the system should:  
 - Check the proposal type and number of pages.  
 - Set parameters `para\_x` and `para\_y` to 0.  
 - Construct the image file name based on the proposal type and number of pages.  
 - If the form version is 10.x, read the image file and display it in the specified image item.  
 - If the form version is not 10.x, hide the image item if it is visible, adjust the image file path, and read the image using a different method.  
 - Set the loaded page to 1.  
  
Definition of Done:  
1. The navigation logic for the family details section is implemented and tested.  
2. The image loading functionality based on proposal type and number of pages is implemented and tested.  
3. The system correctly handles different form versions and adjusts the image display accordingly.  
4. All acceptance criteria are met, and the functionality is verified through testing.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- The query to fetch the previous item in the 'CORRECT' block:  
 ```sql  
 SELECT key\_prev\_item  
 INTO v\_key\_prev  
 FROM azbj\_new\_bbu\_field\_map  
 WHERE LIST\_NAME = :system.current\_item  
 AND LIST\_BLOCK\_NAME = :system.current\_block;  
 ```

# Referal Id Field Validation

Type: TABLE\_FIELD\_MAPPING

Title: Referal Id Field Validation  
  
Acceptance Criteria:  
1. If the group type is 'T', 'I', or 'W', the "Referal Id" field must not be empty. If it is empty, an error message should be displayed prompting the user to enter the "Referal Id" and name.  
2. If the "Referal Id" field is empty and the group type is 'SY', a warning message should be displayed, and a list of values (LOV) should be shown for the user to select from.  
3. If the "Referal Id" field is empty and the group type contains 'Y', a LOV should be created and populated with a specific query. The "Referal Id" field should then be validated based on the list flag value.  
  
Definition of Done:  
- The "Referal Id" field validation logic is implemented as per the acceptance criteria.  
- Appropriate error and warning messages are displayed based on the conditions.  
- The LOV is correctly created and populated with the required data.  
- The "Referal Id" field is validated based on the list flag value.  
- The functionality is tested and verified to ensure it works as expected.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Query to populate LOV for 'SYNDICATE':  
 ```sql  
 Select branch\_code, branch\_name  
 from azbj\_syn\_bank\_branches  
 where br\_type = (case when :CORRECT\_VALUE.FSC\_CODE like '522%' then 'B'  
 else substr(:CORRECT\_VALUE.FSC\_CODE, 1, 3) end)  
 ```  
  
- Query to populate LOV for 'AZBJ\_XY\_RG':  
 ```sql  
 Select branch\_code, branch\_name  
 from azbj\_syn\_bank\_branches  
 where br\_type like '3A6%'  
 ```

# Validation and Population of 'Lead by' Field

Type: TABLE\_FIELD\_MAPPING

Detailed description: As a user, I want to ensure that the "Lead by" field in the proposal form is validated and populated correctly based on specific conditions, so that the data integrity and accuracy are maintained.  
  
Acceptance criteria:  
1. When the "Lead by" field is double-clicked, if the group code (`get\_bnkasr\_grp`) is one of 'T', 'I', 'O', 'DM', 'LG', 'BP' or contains 'X', a list of values should be displayed.  
2. When the "Lead by" field is validated:  
 - If the field is not null and is enabled, and the group code is one of 'T', 'I', 'S', 'W', 'SY', 'DM', 'BP', 'LG' or contains 'X':  
 - If the "Lead by" field is null, an error message "Please Enter Lead By" should be displayed.  
 - The system should fetch the `CUST\_PART\_UNIQUE\_CODE` and `int\_id` from the `dmt\_agents` table where the `reference\_code` matches the `fsc\_code`.  
 - The system should fetch all details from the `CP\_PARTNERS` table where the `PARTNER\_REF` matches the fetched `CUST\_PART\_UNIQUE\_CODE`.  
 - The `ic\_name` field should be populated with the concatenated values of `BEFORE\_TITLE`, `FIRST\_NAME`, `MIDDLE\_NAME`, and `SURNAME` from the `CP\_PARTNERS` table.  
 - If the `fsc\_code` is null or empty, an error message "Please Enter Agent Code" should be displayed.  
  
Definition of Done:  
- The "Lead by" field should trigger the appropriate list of values on double-click based on the group code.  
- The validation logic for the "Lead by" field should be implemented as described, ensuring error messages are displayed and data is fetched and populated correctly.  
- The solution should be tested to ensure it meets the acceptance criteria and works as expected.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Fetching `CUST\_PART\_UNIQUE\_CODE` and `int\_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 = :correct\_value.fsc\_code;  
 ```  
  
- Fetching all details from `CP\_PARTNERS`:  
 ```sql  
 SELECT   
 INTO :AZBJ\_MC\_PARTNER   
 FROM CP\_PARTNERS   
 WHERE CP\_PARTNERS.PARTNER\_REF = :AZBJ\_MC\_PART\_ID;  
 ```

# View Nominee Details on Button Press

Type: TABLE\_FIELD\_MAPPING

Title: View Nominee Details on Button Press  
  
Acceptance Criteria:  
1. When the "Nominee Details" button is pressed:  
 - If the proposal type is 'N' and the number of pages is 19:  
 - Set the parameters `para\_x` and `para\_y` to 0.  
 - Construct the image file name using the image path and file name with a suffix '\_12.TIF'.  
 - If the form version starts with '10.', read the image file and display it in the designated image area.  
 - If the form version does not start with '10.', ensure the image area is not visible, adjust the image file path, and read the image using a different method.  
 - Set the loaded page to 12.  
 - If the proposal type is 'N' and the number of pages is 50:  
 - Set the parameters `para\_x` and `para\_y` to 0.  
 - Construct the image file name using the image path and file name with a suffix '\_25.TIF'.  
 - If the form version starts with '10.', read the image file and display it in the designated image area.  
 - If the form version does not start with '10.', ensure the image area is not visible, adjust the image file path, and read the image using a different method.  
 - Set the loaded page to 1.  
 - If the proposal type is 'O' and the number of pages is 14:  
 - Set the parameters `para\_x` and `para\_y` to 0.  
 - Construct the image file name using the image path and file name with a suffix '\_6.TIF'.  
 - If the form version starts with '10.', read the image file and display it in the designated image area.  
 - If the form version does not start with '10.', ensure the image area is not visible, adjust the image file path, and read the image using a different method.  
 - Set the loaded page to 6.  
 - If the proposal type is 'O' and the number of pages is 27:  
 - Set the parameters `para\_x` and `para\_y` to 0.  
 - Construct the image file name using the image path and file name with a suffix '\_11.TIF'.  
 - If the form version starts with '10.', read the image file and display it in the designated image area.  
 - If the form version does not start with '10.', ensure the image area is not visible, adjust the image file path, and read the image using a different method.  
 - Set the loaded page to 6.  
  
Definition of Done:  
- The "Nominee Details" button should trigger the appropriate actions based on the proposal type and number of pages.  
- The image file should be correctly constructed and displayed or hidden as per the conditions.  
- The loaded page should be set correctly based on the conditions.  
- The functionality should be tested and verified to ensure it works as expected.

# Employee ID Validation

Type: TABLE\_FIELD\_MAPPING

Title: Employee ID Validation  
  
Acceptance Criteria:  
1. The Employee ID must be between 4 and 6 digits long.  
2. The Employee ID must not contain any special characters.  
3. The Employee ID must not contain any alphabetic characters.  
4. If the Employee ID is greater than 6 digits and the FSC code starts with '59C', '59S', '5X9', or '59L', an error message should be displayed stating "Employee Id should not be greater than 6 digits."  
5. If the Employee ID is less than 4 digits and the FSC code starts with '59C', '59D', '59S', '5X9', '52S', or '59L', an error message should be displayed stating "Employee Id should not be less than 4 digits."  
6. If the Employee ID contains special characters, an error message should be displayed stating "Special characters are not allowed."  
7. If the Employee ID contains alphabetic characters, an error message should be displayed stating "Alphabetic characters are not allowed."  
  
Definition of Done:  
- The validation rules for the Employee ID field are implemented and tested.  
- Error messages are displayed correctly based on the validation rules.  
- The system prevents the user from entering invalid Employee IDs.  
- The functionality is tested and verified to ensure it meets the acceptance criteria.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- The following query checks if the Employee ID LOV is allowed based on the MASTER\_POLICY\_NO:  
 ```sql  
 SELECT COUNT (1)  
 INTO v\_emplov\_cnt  
 FROM azbj\_system\_constants  
 WHERE sys\_type = 'GROUP'  
 AND sys\_code = 'ALLOW\_EMPLOYEE\_ID\_LOV'  
 AND sys\_desc LIKE '%|' || nvl(:CORRECT\_VALUE.MASTER\_POLICY\_NO,v\_master\_policy\_no) || '|%';  
 ```  
  
- The following query checks the count of agents based on the FSC code:  
 ```sql  
 SELECT COUNT ()  
 INTO v\_agent\_count  
 FROM azbj\_system\_constants  
 WHERE sys\_type = 'GRP'  
 AND sys\_code = 'GRP\_AGENT'  
 AND char\_value = :correct\_value.fsc\_code;  
 ```

# Display LOV for SUB\_NAME based on specific conditions

Type: TABLE\_FIELD\_MAPPING

Detailed description: As a user, I want the system to display a list of values (LOV) for the "SUB\_NAME" field based on specific conditions, so that I can select the appropriate value efficiently.  
  
Acceptance criteria:  
1. When the user double-clicks on the "SUB\_NAME" field or presses a key to list values:  
 - If the group value (retrieved by `get\_bnkasr\_grp`) is one of 'T', 'I', 'O', 'DM', 'LG', 'BP' or contains 'X', and the `FSC\_CODE` does not start with '59D', then the system should display a list of values.  
 - If the `FSC\_CODE` starts with '59D', then the system should show a specific LOV named 'GRP\_XX2\_LOV'.  
  
Definition of Done:  
- The system correctly displays the LOV based on the specified conditions.  
- The LOV is populated with values from the `azbj\_syn\_bank\_branches` table where the branch type matches 'N59DL%'.  
- The functionality is tested and verified to work as expected in both scenarios.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- The LOV 'GRP\_XX2\_LOV' is populated using the following query:  
 ```sql  
 SELECT BRANCH\_NAME   
 FROM azbj\_syn\_bank\_branches  
 WHERE br\_type LIKE 'N59DL%'  
 ```

# Manage Proposal, Agent, and Insured Person Details

Type: CONTROL

Title: Manage Proposal, Agent, and Insured Person Details  
  
Acceptance Criteria:  
1. The system should allow the user to view and update personal details such as name, occupation, and contact information.  
2. The system should provide options to manage policy details, including policy numbers, dates, and statuses.  
3. The system should include functionalities to handle verification processes for documents like PAN cards and Aadhar cards.  
4. The system should support the management of additional details such as financial documents, consent information, and relationship details.  
5. The system should provide buttons for actions like saving, canceling, and validating information.  
6. The system should display various statuses and flags related to the proposal, agent, and insured person, such as eligibility flags, risk scores, and verification statuses.  
  
Definition of Done:  
1. The user interface should be able to display and update personal details, policy details, and verification statuses.  
2. The system should include buttons for saving, canceling, and validating information.  
3. The system should display relevant statuses and flags.  
4. The system should handle the initialization and updating of various fields as per the requirements.  
5. The system should be tested 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 that can be directly executed in the database.

# Manage visibility and enablement of commission field based on 'IC's Own Policy' checkbox

Type: CONTROL

Detailed description: As a user, I want to manage the visibility and enablement of the commission field based on the selection of the "IC's Own Policy" checkbox, so that the commission field is only visible and enabled when the checkbox is checked.  
  
Acceptance criteria:  
1. When the "IC's Own Policy" checkbox is checked:  
 - The commission field should become visible.  
 - The commission field should become enabled.  
2. When the "IC's Own Policy" checkbox is unchecked:  
 - The commission field should become hidden.  
  
Definition of Done:  
- The commission field's visibility and enablement should dynamically change based on the state of the "IC's Own Policy" checkbox.  
- The changes should be tested and verified to ensure they work as expected in all 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.

# Manage Assignment Flag for Policies

Type: CONTROL

Title: Manage Assignment Flag for Policies  
  
Acceptance Criteria:  
1. When the assignment flag is marked at the scrutiny level and the user attempts to remove it, an alert should prompt the user to confirm the action.  
 - If the user confirms, the flag should be removed.  
 - If the user cancels, the flag should remain set.  
2. If the policyholder has opted for an Electronic Insurance Account (eIA) and the assignment flag is checked, the system should automatically uncheck the flag and display a message indicating that eIA is not allowed for assignment cases.  
  
Definition of Done:  
- The assignment flag can be set or removed based on user actions and predefined conditions.  
- Alerts and messages are displayed correctly to guide the user through the process.  
- The system ensures that eIA and assignment flags are mutually exclusive.  
- All changes are saved and reflected accurately in the system.  
  
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.

# View IC Commission Details

Type: CONTROL

Title: View IC Commission Details  
  
Acceptance Criteria:  
1. When the button labeled "IC Commission Det" is pressed, the system should check if a parameter list named 'Param1' already exists.  
2. If 'Param1' exists, it should be destroyed and a new parameter list named 'Param1' should be created.  
3. The parameter list should include a parameter named 'PARAM\_AGENT\_CODE' with the value of the agent code from the current context.  
4. The system should then open a new form named 'AZBJ\_IC\_COMMISSION\_DTLS' with the created parameter list.  
  
Definition of Done:  
- The button should be visible and correctly labeled as "IC Commission Det".  
- Pressing the button should trigger the described sequence of actions.  
- The new form 'AZBJ\_IC\_COMMISSION\_DTLS' should open with the correct parameter passed.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include any direct database queries.

# View and Verify Aadhaar Details for Multiple Customer Profiles

Type: CONTROL

Title: View and Verify Aadhaar Details for Multiple Customer Profiles  
  
Acceptance Criteria:  
1. When the user presses the "Aadhaar details with multiple CP" button, the system should:  
 - Check if the Aadhaar details for the current customer profile (IP and PH) are available and valid.  
 - Retrieve and verify the Aadhaar details from the AML block if they exist.  
 - If the Aadhaar details are valid, pass the relevant parameters (IP and PH Aadhaar numbers, module name, etc.) to the Aadhaar details form.  
 - If the Aadhaar details are not selected or invalid, display a warning message to the user.  
  
Definition of Done:  
- The button should be disabled by default and only enabled when the necessary conditions are met.  
- The system should correctly handle the retrieval and verification of Aadhaar details.  
- The system should pass the correct parameters to the Aadhaar details form.  
- Appropriate messages should be displayed to the user if Aadhaar details are not selected or invalid.  
- 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 logic involves Oracle Forms-specific constructs and cannot be executed directly in the database without modification.

# RCU Reasons Button Functionality

Type: CONTROL

Title: RCU Reasons Button Functionality  
  
Acceptance Criteria:  
1. When the "RCU Reasons" button is pressed, the system should check if a parameter list named 'Param1' already exists.  
2. If 'Param1' exists, it should be destroyed and a new parameter list named 'Param1' should be created.  
3. The system should add the following parameters to the parameter list:  
 - 'RCU\_COMMENTS' with the value from the field `CONTROL.RCU\_COMMENTS`.  
 - 'MODULE\_NAME' with the value 'DEQC'.  
 - 'APPLN\_NO' with the value from the field `correct\_value.appln\_no`.  
4. The system should then call a form named 'AZBJ\_RCU\_COMMENTS' with the created parameter list.  
  
Definition of Done:  
- The "RCU Reasons" button is visible and clickable.  
- Pressing the button performs the described actions and opens the 'AZBJ\_RCU\_COMMENTS' form with the correct parameters.  
- The form 'AZBJ\_RCU\_COMMENTS' displays the comments and other relevant information based on the provided parameters.  
  
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 Enrichment Details for Insured Person

Type: CONTROL

Title: Manage Enrichment Details for Insured Person  
  
Acceptance Criteria:  
1. When the "Enrichment Details" button is pressed, the system should concatenate the insured person's title, first name, middle name, and surname into a single string.  
2. The system should check if a parameter list named 'Param1' exists. If it does, it should be destroyed and recreated.  
3. The system should add the following parameters to the newly created parameter list:  
 - `APPLICATION\_NO` with the value of the application number.  
 - `CALL\_FORM\_NAME` with the value 'DEQC'.  
 - `PROPOSAL\_NO` with a null value.  
 - `LA\_NAME` with the concatenated insured person's name.  
4. The system should call a form named 'AZBJ\_DATA\_ENRICH\_FORM' with the created parameter list.  
5. If any error occurs during this process, an error message should be displayed with the error details.  
  
Definition of Done:  
- The "Enrichment Details" button should trigger the described logic and call the appropriate form with the correct parameters.  
- The system should handle errors gracefully and display appropriate error messages.  
- 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 logic does not include direct database CRUD operations.

# Manage Nominee Details Based on MWP Act Checkbox

Type: CONTROL

Title: Manage Nominee Details Based on MWP Act Checkbox  
  
Acceptance Criteria:  
1. When the MWP Act checkbox is checked:  
 - The nominee name should be set to "Policy is issued under MWP Act".  
 - The nominee name, birthplace, date of birth, and relationship fields should be disabled and not allow insert or update operations.  
2. When the MWP Act checkbox is unchecked:  
 - The nominee name should be cleared.  
 - The nominee name, birthplace, date of birth, and relationship fields should be enabled and allow insert and update operations.  
  
Definition of Done:  
- The system correctly updates the nominee details based on the MWP Act checkbox status.  
- The nominee fields are enabled or disabled as per the acceptance criteria.  
- The changes are tested and verified to ensure they meet 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 CRUD operations.

# View Previous Policy PAN Details

Type: CONTROL

Title: View Previous Policy PAN Details  
  
Acceptance Criteria:  
1. When the "Previous Policy PAN Details" button is pressed, the system should:  
 - Check if a parameter list named 'Param1' already exists. If it does, it should be destroyed.  
 - Create a new parameter list named 'Param1'.  
 - Add the following parameters to the parameter list:  
 - `PAR\_PH\_PART\_ID` with the value of `ph\_part\_id`.  
 - `PAR\_PAN\_CARD\_NO` with the value of `PH\_PAN\_CARD\_NO`.  
 - `PAR\_PAN\_PH\_NAME` with the value of `PH\_NAME`.  
 - `PAR\_PAN\_PH\_DOB` with the value of `PH\_DOB`.  
 - `PAR\_MODULE` with the value 'DEQC'.  
 - Call the form 'AZBJ\_OLD\_POLICY\_DTLS' with the created parameter list.  
 - Set the variable `v\_pan\_pol\_validate` to 'Y'.  
  
Definition of Done:  
- The button should be visible and labeled "Previous Policy PAN Details".  
- The button should be disabled by default.  
- When the button is pressed, the system should perform the actions as described in the acceptance criteria.  
- The form 'AZBJ\_OLD\_POLICY\_DTLS' should be called with the correct parameters.  
- The variable `v\_pan\_pol\_validate` should be set to 'Y' after the form is called.  
  
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 EIA Details for Application

Type: CONTROL

Title: Manage EIA Details for Application  
  
Acceptance Criteria:  
1. When the "EIA Details" button is pressed, the system should:  
 - Retrieve the scrutiny number associated with the application number.  
 - Set a flag indicating the EIA details button was clicked.  
 - Create a parameter list and add the following parameters:  
 - Application number  
 - Module flag set to 'DEQC'  
 - Property type  
 - Scrutiny number  
 - Policy reference  
 - Partner ID  
 - Agent code  
 - Call the form 'AZBJ\_INSURANCE\_REPOSITORY' with the created parameter list.  
 - Check if there are existing EIA details for the application with a top indicator of 'Y' and an EIA account type of 'New\_Applicant'.  
 - Enable or disable the 'EI\_ACCOUNT\_OPENING\_DE' field based on the existence of the EIA details.  
  
2. When the user navigates to the next item, the system should:  
 - Move the focus to the 'AML\_NEXT\_TAB' field.  
 - Set a variable indicating the next item is 'AML\_NEXT\_TAB'.  
  
Definition of Done:  
- The "EIA Details" button functionality is implemented and tested.  
- The system correctly retrieves and processes the scrutiny number.  
- The parameter list is created and passed to the 'AZBJ\_INSURANCE\_REPOSITORY' form.  
- The 'EI\_ACCOUNT\_OPENING\_DE' field is enabled or disabled based on the existence of EIA details.  
- The navigation to the next item works as expected.  
- All functionalities are tested and verified to be working correctly.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Retrieve scrutiny number:  
 ```sql  
 SELECT scrutiny\_no  
 INTO v\_scrutiny\_no  
 FROM azbj\_phub\_scrutiny\_prop  
 WHERE application\_no = :correct\_value.appln\_no;  
 ```  
  
- Check existence of EIA details:  
 ```sql  
 SELECT COUNT (1)  
 INTO v\_exists  
 FROM azbj\_eia\_details  
 WHERE application\_no = :correct\_value.appln\_no  
 AND top\_indicator = 'Y'  
 AND eia\_account\_type = 'New\_Applicant';  
 ```

# Implement MHR Field with Dropdown List

Type: CONTROL

Title: Implement MHR Field with Dropdown List  
  
Acceptance Criteria:  
1. The "MHR" field should display a dropdown list with the following options:  
 - 5  
 - 8  
 - 10  
 - SL (No)  
2. The "MHR" field should be editable, allowing users to select a value from the list.  
3. The "MHR" field should be visible on the "AML\_KYC" tab page.  
4. The "MHR" field should have a white background and black text.  
5. The "MHR" field should be positioned at coordinates (710, 220) on the screen.  
6. The "MHR" field should have a prompt labeled "MHR" with specific font settings (Tahoma, Bold, size 800).  
  
Definition of Done:  
- The "MHR" field is implemented and displays the correct list of options.  
- Users can select and update the value in the "MHR" field.  
- The "MHR" field is correctly positioned and styled as per the requirements.  
- The "MHR" field is visible on the "AML\_KYC" tab page.  
- 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 database queries.

# Document DE Button Functionality

Type: CONTROL

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 the following parameters to the parameter list:  
 - 'PARAM\_APPLN\_NO' with the value from the field `correct\_value.appln\_no`.  
 - 'PARAM\_PROPOSAL\_NO' with a null value.  
 - 'PARAM\_MODULE\_NAME' with the value 'DEQC'.  
 - 'PARAM\_PH\_DOB' with the value from the field `CORRECT\_VALUE.PH\_DOB`.  
2. The system should navigate to the 'AML' block and iterate through all records to find the value of 'PAN Card' if the 'proof\_type' is 'PC'.  
3. The system should add the 'PAN Card' value to the parameter list as 'PARAM\_PAN'.  
4. The system should increment a counter `pk\_vars.v\_document\_de\_cnt`.  
5. The system should call another form named 'AZBJ\_FINANCIAL\_DOCUMENT' with the parameter list.  
  
Definition of Done:  
- The button "Document DE" is functional and initiates the described process.  
- The parameters are correctly collected and passed to the next form.  
- The counter `pk\_vars.v\_document\_de\_cnt` is incremented as expected.  
- The form 'AZBJ\_FINANCIAL\_DOCUMENT' is called with the correct parameters.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided logic does not include direct CRUD operations on database tables.

# Manage Passport Details for Insured Person and Policyholder

Type: CONTROL

Title: Manage Passport Details for Insured Person and Policyholder  
  
Acceptance Criteria:  
1. When the "Passport Details" button is pressed, the system should check if the age proof for the insured person (IP) or policyholder (PH) is a passport.  
2. If the age proof is a passport, the system should retrieve and store the passport number, date of birth, age, and gender for both the insured person (IP) and policyholder (PH).  
3. The system should navigate to the 'AML' section and iterate through records to find any additional passport details if available.  
4. The system should determine if the insured person (IP) and policyholder (PH) are the same based on a specific condition.  
5. The system should add the retrieved details as parameters and call a form to display the passport details.  
6. If no passport details are found for either the insured person (IP) or policyholder (PH), the system should display a warning message indicating that passport details are not selected.  
  
Definition of Done:  
- The "Passport Details" button functionality is implemented and tested.  
- The system correctly retrieves and stores passport details for both the insured person (IP) and policyholder (PH).  
- The system navigates to the 'AML' section and processes records as required.  
- The system correctly determines if the insured person (IP) and policyholder (PH) are the same.  
- The system successfully calls the form to display passport details with the correct parameters.  
- A warning message is displayed if no passport details are found for either the insured person (IP) or policyholder (PH).  
- All acceptance criteria are met and verified through testing.

# View Bank Account Details

Type: CONTROL

Title: View Bank Account Details  
  
Acceptance Criteria:  
1. When the "Same Bank Dtls" button is pressed, the system should check if both the bank account number and IFSC code fields are not empty.  
2. If either the bank account number or IFSC code is empty, the system should display a warning message: "Please enter the Account No and IFSC Code.!" and prevent further action.  
3. If both fields are filled, the system should pass these values as parameters to another form named 'AZBJ\_SAME\_BANK\_DETAILS' and open this form for the user to view the bank details.  
4. The system should handle any exceptions that occur during this process gracefully without crashing.  
  
Definition of Done:  
- The "Same Bank Dtls" button is functional and performs the described actions.  
- The system correctly validates the presence of the bank account number and IFSC code before proceeding.  
- The warning message is displayed appropriately when required fields are missing.  
- The 'AZBJ\_SAME\_BANK\_DETAILS' form is opened with the correct parameters when both fields are filled.  
- All exceptions are handled, and the system remains stable during the process.

# Manage Driving License Details for IP and PH

Type: CONTROL

Title: Manage Driving License Details for IP and PH  
  
Acceptance Criteria:  
1. When the "Driving License Det" button is pressed, the system should:  
 - Check if the age proof for both IP and PH is a driving license.  
 - Retrieve and store the driving license number and date of birth for both IP and PH.  
 - Navigate to the AML block and iterate through the records to find any driving license details.  
 - Determine if the IP and PH details are the same based on a specific condition.  
 - Add the retrieved details as parameters for further processing.  
 - Call a form to display the driving license details if any driving license number is found.  
 - Display a warning message if no driving license details are selected for IP/PH.  
  
2. When the user navigates to the next item, the system should:  
 - Move the focus to the next item in the "CORRECT\_VALUE" block.  
  
Definition of Done:  
- The functionality to view and manage driving license details for both IP and PH is implemented and tested.  
- The system correctly retrieves, processes, and displays the driving license details.  
- Appropriate messages are displayed when no driving license details are selected.  
- Navigation to the next item works as expected.  
- All acceptance criteria are met and verified through testing.

# Dynamic Management of PAN Card Fields Based on Checkbox Selection

Type: CONTROL

Title: Dynamic Management of PAN Card Fields Based on Checkbox Selection  
  
Acceptance Criteria:  
1. When the checkbox for "PAN Card not Available" is checked:  
 - The field for selecting PAN card options (PH\_NO\_PAN\_LOV) should become visible and enabled.  
 - The field for entering the PAN card number (PH\_PAN\_CARD\_NO) should be disabled and its value cleared.  
 - The fields for entering the first name, middle name, last name, current status, and last updated date related to the PAN card should be cleared.  
 - If the address is from specific states (Arunachal Pradesh, Manipur, Mizoram, Assam, Nagaland, Meghalaya), the customer status field (ST\_CUSTOMER) should become visible and enabled.  
  
2. When the checkbox for "PAN Card not Available" is unchecked:  
 - The field for selecting PAN card options (PH\_NO\_PAN\_LOV) should become hidden and its value cleared.  
 - The field for entering the PAN card number (PH\_PAN\_CARD\_NO) should be enabled.  
 - The customer status field (ST\_CUSTOMER) should become hidden if it was previously visible.  
  
Definition of Done:  
- The form dynamically adjusts the visibility and enablement of fields based on the checkbox selection.  
- All specified fields are cleared or reset as per the conditions.  
- The form logic is thoroughly tested to ensure it works as expected for all specified conditions.  
- The implementation is independent of any specific technology or Oracle Forms terminology.

# Partner Type Selection

Type: CONTROL

Title: Partner Type Selection  
  
Acceptance Criteria:  
1. The partner type field should display a list of predefined values: "5", "8", "10", and "SL".  
2. When a partner type is selected from the list, the system should trigger an event to handle the change in partner type.  
  
Definition of Done:  
1. The partner type field is visible and accessible on the user interface.  
2. The list of predefined values is correctly populated in the partner type field.  
3. Selecting a value from the list triggers the appropriate event to handle the change in partner type.  
4. The user interface updates to reflect the selected partner type.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include specific database queries.

# Voter ID Verification for Insured Person and Policyholder

Type: CONTROL

Title: Voter ID Verification for Insured Person and Policyholder  
  
Acceptance Criteria:  
1. When the "Voter ID" button is pressed, the system should check if the age proof for either the insured person or the policyholder is marked as 'VI' (Voter ID).  
2. If the age proof is 'VI', the system should set a flag indicating that the Voter ID is verified.  
3. If the age proof is not 'VI', the system should navigate to the 'AML' block and iterate through the records to check if any proof type is 'VI'. If found, the system should set the Voter ID verification flag.  
4. The system should gather and format the following details:  
 - Full name of the insured person and policyholder  
 - Date of birth of the insured person and policyholder  
 - Address of the insured person and policyholder  
 - State of the insured person  
 - Whether the insured person and policyholder are the same  
5. The system should add these details as parameters and call a verification form if the Voter ID is verified.  
6. If the Voter ID is not verified, the system should display a message indicating that the Voter’s ID Card Details are not selected for IP/PH.  
  
Definition of Done:  
- The Voter ID verification process is triggered upon pressing the "Voter ID" button.  
- The system correctly identifies and sets the Voter ID verification flag based on the provided age proof.  
- The system accurately gathers and formats the required details.  
- The verification form is called with the correct parameters if the Voter ID is verified.  
- An appropriate message is displayed if the Voter ID is not verified.  
- The functionality is tested and confirmed to work as expected without any Oracle Forms-specific terminology.  
  
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.

# PAN Card Verification and Question Population

Type: CONTROL

Title: PAN Card Verification and Question Population  
  
Acceptance Criteria:  
1. When the PAN card verification status is changed, the system should handle the change without any errors.  
2. If the PAN card is not verified, a warning message should be displayed indicating that the PAN card is received but not verified.  
3. Upon pressing the key to move to the next item, the system should navigate to the "Populate Questions" section.  
4. The system should populate the questions based on the provided partner details and application number.  
5. The system should delete existing records if the member number matches the partner number.  
6. The system should create new records with the populated questions.  
7. Specific questions should be deleted based on predefined conditions (e.g., question ID and sub-question values).  
  
Definition of Done:  
- The PAN card verification status change is handled without errors.  
- A warning message is displayed if the PAN card is not verified.  
- Navigation to the "Populate Questions" section works correctly.  
- Questions are populated based on the provided partner details and application number.  
- Existing records are deleted if the member number matches the partner number.  
- New records are created with the populated questions.  
- Specific questions are deleted based on predefined conditions.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- The provided SQL query in the XML content is used to fetch and populate questions based on the partner details and application number. The query joins two tables (`azbj\_bbu\_questions` and `azbj\_fcf\_questionnaire`) and applies conditions to filter the results. The results are then used to populate the questionnaire records.

# BI Date Validation

Type: CONTROL

Title: BI Date Validation  
  
Acceptance Criteria:  
1. When the BI Date is entered, it should be validated to ensure it is not greater than the Proposal Sign Date.  
2. If the BI Date is greater than the Proposal Sign Date, a warning message should be displayed: "BI SIGNATURE DATE entered is greater than PROPOSAL SIGN DATE".  
3. The system should allow navigation to the next item after entering the BI Date.  
  
Definition of Done:  
1. The BI Date field is implemented and visible on the user interface.  
2. The validation logic for comparing BI Date with Proposal Sign Date is implemented and functional.  
3. A warning message is displayed if the BI Date is greater than the Proposal Sign Date.  
4. The user can navigate to the next item after entering the BI Date.  
5. All acceptance criteria are met and tested successfully.

# BI Number Manual Entry and Validation

Type: CONTROL

Title: BI Number Manual Entry and Validation  
  
Acceptance Criteria:  
1. The system should check if the BI number field is enabled and not null.  
2. If the BI number field is null, the system should display a warning message: "BI No. is mandatory. Please check."  
3. The system should populate various fields related to the application using the `populate\_carequote` procedure.  
4. If there is an error during the population process, the system should display the error message.  
5. The system should handle the BI date field based on the BI number's presence and enable the BI date field if it is disabled.  
6. The system should navigate to the BI received date field after processing the BI number.  
  
Definition of Done:  
- The BI number validation and manual entry process is implemented.  
- The warning message for a missing BI number is displayed correctly.  
- The `populate\_carequote` procedure is called with the appropriate parameters.  
- Error messages are displayed if any issues occur during the population process.  
- The BI date field is handled correctly based on the BI number's presence.  
- The system navigates to the BI received date field after processing the BI number.  
  
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.

# PAN Card Number Validation and Personal Details Retrieval

Type: CONTROL

Detailed description: As a user, I want to validate the PAN card number entered in the system to ensure it is not associated with any existing policies and retrieve associated personal details if it is valid.  
  
Acceptance criteria:  
1. When a PAN card number is entered, the system should check if the PAN card number exists in the database and is not associated with specific tax IDs ('AG/NRI/60A', 'AG/NRI/61A').  
2. If the PAN card number is found in the database and meets the criteria, the system should:  
 - Set a validation flag to 'N'.  
 - Enable the field for previous policy details.  
 - Display a warning message to the user to check the PAN number found in previous policies.  
3. If the PAN card number is not found or does not meet the criteria, the system should:  
 - Set the validation flag to 'Y'.  
 - Disable the field for previous policy details.  
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 no data is found for the PAN card number, the system should handle the exception gracefully without any errors.  
6. The user should be prompted to select the partner type after the validation process.  
  
Definition of Done:  
- The PAN card number validation logic is implemented and tested.  
- The system correctly enables or disables the previous policy details field based on the validation result.  
- The system retrieves and displays the associated personal details if the PAN card number is valid.  
- Appropriate messages are displayed to the user based on the validation result.  
- The user is prompted to select the partner type after the validation process.  
- All acceptance criteria are met, and the functionality is tested and verified.  
  
DB queries for Table reference CRUD operations only(With Usage):  
```sql  
-- Query to check if the PAN card number exists and meets the criteria  
SELECT COUNT(1)  
INTO v\_pan\_cnt  
FROM ocp\_policy\_bases a, ocp\_interested\_parties b, cp\_partners c  
WHERE a.contract\_id = b.contract\_id  
 AND b.partner\_id = c.part\_id  
 AND c.tax\_id = :control.ph\_pan\_card\_no  
 AND C.TAX\_ID NOT IN ('AG/NRI/60A', 'AG/NRI/61A')  
 AND a.top\_indicator = 'Y'  
 AND a.action\_code <> 'D'  
 AND b.top\_indicator = 'Y'  
 AND b.action\_code <> 'D'  
 AND ROWNUM = 1;  
  
-- Query to retrieve personal details associated with the PAN card number  
SELECT middle\_name, first\_name, surname, DATE\_OF\_BIRTH  
INTO :control.ph\_pan\_mid\_name,  
 :control.ph\_pan\_fir\_name,  
 :control.ph\_pan\_last\_name,  
 :control.ph\_pan\_dob  
FROM cp\_partners  
WHERE part\_id = CASE   
 WHEN :CONTROL.IP\_PH = 'IP' THEN :correct\_value.ip\_part\_id  
 WHEN :CONTROL.IP\_PH = 'PH' THEN :correct\_value.ph\_part\_id  
 ELSE :correct\_value.ph\_part\_id  
 END;  
```

# Save Duplicate Contact Number Details

Type: CONTROL

Title: Save Duplicate Contact Number Details  
  
Acceptance Criteria:  
1. When the "Save" button is pressed, the system should execute the function to save duplicate contact number details.  
2. If an error occurs during the saving of duplicate contact number details, an appropriate error message should be displayed.  
3. The system should then check if the variable `v\_grp\_product` is set to 'Y'.  
 - If `v\_grp\_product` is 'Y', the system should execute the function to save group details.  
 - If `v\_grp\_product` is not 'Y', the system should execute the function to save non-group details.  
4. If an error occurs during the saving of group or non-group details, an appropriate error message should be displayed.  
  
Definition of Done:  
- The "Save" button functionality is implemented and tested.  
- The system correctly saves duplicate contact number details and handles any errors.  
- The system correctly checks the `v\_grp\_product` variable and saves group or non-group details accordingly, handling any errors.  
- All acceptance criteria are met and verified through testing.

# Customer Status Dropdown List

Type: CONTROL

Title: Customer Status Dropdown List  
  
Acceptance Criteria:  
1. The customer status field should be a dropdown list with the following options:  
 - 5  
 - 8  
 - SL  
2. The dropdown list should be located on the "AML\_KYC" tab of the form.  
3. The field should be initially hidden and only become visible under specific conditions defined by the business logic.  
4. The dropdown list should be styled with a white background and black text.  
5. The dropdown list should be positioned at coordinates (373, 0) on the form.  
  
Definition of Done:  
- The customer status dropdown list is implemented and displays the correct options.  
- The dropdown list is correctly positioned and styled as specified.  
- The dropdown list is hidden by default and becomes visible based on the defined conditions.  
- 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 mentioned in the provided XML content.

# Handle Selection and Validation of PH\_NO\_PAN\_LOV List Item

Type: CONTROL

Detailed description: As a user, I want the system to handle the selection of a specific list item (PH\_NO\_PAN\_LOV) in a way that ensures the correct handling of form types and address validations, so that the appropriate actions are taken based on the user's input and address details.  
  
Acceptance criteria:  
1. When the user selects 'FORM\_60' from the list, the system should:  
 - Set a flag indicating that 'FORM\_60' is selected.  
 - Disable the input field for the PAN card number if it is currently enabled.  
  
2. When the user selects any other option from the list, the system should:  
 - Reset the flag indicating that 'FORM\_60' is not selected.  
  
3. The system should check if the address lines contain specific values ('LEH' and 'JAMMU AND KASHMIR') and set a flag accordingly.  
  
4. The system should validate the selected form type based on the user's nationality and address details:  
 - If 'FORM\_60' is selected and the user is an Indian national or the address does not meet specific criteria, the system should display an error message indicating that 'FORM\_60' is allowed only for NRI proposals.  
 - If 'FORM\_61' is selected and the user's occupation is not 'Agriculture', the system should display an error message indicating that 'FORM\_61' is allowed only for the occupation 'Agriculture'.  
  
Definition of Done:  
- The system correctly handles the selection of 'FORM\_60' and 'FORM\_61' based on the user's input and address details.  
- Appropriate error messages are displayed when the form type selection does not meet the specified criteria.  
- The PAN card number input field is disabled when 'FORM\_60' is selected.  
- The system sets and resets flags as per the specified conditions.  
- All validations and actions are performed 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 CRUD operations.

# Implement Fresh Proposal Number Field

Type: CONTROL

Title: Implement Fresh Proposal Number Field  
  
Acceptance Criteria:  
1. When the user enters a fresh proposal number, the system should automatically move the cursor to the next item, which is the fresh proposal date.  
2. The fresh proposal number field should accept numeric input only and should be displayed in uppercase.  
3. The field should have a maximum length of 15 characters.  
4. The field should be visually distinct with a gray background and black text.  
5. The prompt for the field should read "Fresh Proposal No./DGH No." and should be bold and plain in style.  
  
Definition of Done:  
1. The fresh proposal number field is implemented and accepts numeric input only.  
2. The cursor moves to the fresh proposal date field upon entering a value in the fresh proposal number field.  
3. The field adheres to the specified visual and formatting requirements.  
4. The prompt is correctly displayed and styled as specified.  
  
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.

# User can select the type of account from a predefined list

Type: CONTROL

Title: User can select the type of account from a predefined list  
  
Acceptance Criteria:  
1. The user should be able to see a list of account types with the following options:  
 - 5  
 - 8  
 - 10  
 - SL (No)  
2. When the user selects an account type and presses the "Next" key, the focus should move to the "Proposal Sign Date" field.  
3. When the user presses the "Previous" key, the focus should move to the "IFSC Code" field.  
  
Definition of Done:  
- The list of account types is displayed correctly.  
- Navigation between fields using the "Next" and "Previous" keys works as specified.  
- 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.

# Cancel Operation

Type: CONTROL

Title: Cancel Operation  
  
Acceptance Criteria:  
1. When the "Cancel" button is pressed, an alert should be displayed asking the user, "Do you want to Cancel?" with options "Yes" and "No".  
2. If the user selects "Yes", the current form should be cleared, and the view should be hidden.  
3. The system should then navigate back to the main form and set focus on the "Validate" field.  
4. If the user selects "No", the system should set focus on the "Save" button of the current form.  
  
Definition of Done:  
- The "Cancel" button should trigger an alert with the specified message and options.  
- Selecting "Yes" should clear the current form, hide the view, and navigate back to the main form with focus on the "Validate" field.  
- Selecting "No" should set focus on the "Save" button of the current form.  
- The functionality should be tested and verified to ensure it works as expected.

# Date Input for Fresh Proposal or DGH Date

Type: CONTROL

Title: Date Input for Fresh Proposal or DGH Date  
  
Acceptance Criteria:  
1. The date input field should accept dates in the format DD/MM/YYYY.  
2. Upon entering the date, the system should automatically navigate to the next input field for the PAN card number.  
3. The date input field should be displayed with a gray background and black text.  
4. The prompt for the date input field should be displayed above the field and should read "Fresh Proposal Sign Date/DGH Date".  
5. The prompt should be bold and in plain font style.  
  
Definition of Done:  
1. The date input field is implemented and accepts dates in the specified format.  
2. The system successfully navigates to the PAN card number input field after the date is entered.  
3. The date input field and its prompt are displayed with the specified styles and positions.  
4. All acceptance criteria are met and tested for accuracy.  
  
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.

# Validate BI Received Date against BI Date

Type: CONTROL

Detailed description: As a user, I want to ensure that the "BI Received Date" field is validated against the "BI Date" field so that the received date is not earlier than the BI date, ensuring data integrity and accuracy.  
  
Acceptance criteria:  
1. If the "BI Received Date" is not null and is earlier than the "BI Date":  
 - If the "instab" field is 'INSTAB' and the "PROPTYPE" field is 'I', and the "fsc\_code" is not '2000003060' and the "product\_code" is not 321, a warning message should be displayed: "BI Received date should not be less than BI Date."  
 - Otherwise, an error message should be displayed: "BI Received date should not be less than BI Date."  
2. The focus should remain on the "BI Received Date" field if the validation fails.  
3. If the validation passes, the focus should move to the "BANK\_NEXT\_TAB" field.  
  
Definition of Done:  
- The validation logic for the "BI Received Date" field is implemented and tested.  
- Appropriate warning or error messages are displayed based on the conditions specified.  
- The focus behavior of the fields is correctly implemented based on the validation results.  
- The functionality is tested and verified to ensure data integrity and accuracy.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided logic does not include any direct database CRUD operations.

# Consent Information Input

Type: CONTROL

Detailed description: As a user, I want to provide consent information for data sharing in a structured format, so that my preferences are clearly recorded and can be easily referenced.  
  
Acceptance criteria:  
1. The consent information should be presented as a list with predefined options.  
2. The list should include the following options:  
 - Option 1: Value "5"  
 - Option 2: Value "8"  
 - Option 3: Value "10"  
 - Option 4: Value "SL" (No)  
3. Upon selecting an option and pressing the next key, the focus should automatically move to the next input field for thumb impression.  
  
Definition of Done:  
1. The consent information list is displayed correctly with all predefined options.  
2. The user can select an option from the list.  
3. After selection, pressing the next key moves the focus to the thumb impression input field.  
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 any database-specific operations.

# Save Shortcut Button Functionality

Type: CONTROL

Title: Save Shortcut Button Functionality  
  
Acceptance Criteria:  
- The save shortcut button should be enabled only if the main save functionality is available.  
- When the save shortcut button is pressed, it should trigger the main save action.  
- The system should handle any exceptions gracefully without causing the application to crash.  
  
Definition of Done:  
- The save shortcut button is implemented and visible on the user interface.  
- The button is only enabled when the main save functionality is available.  
- Pressing the save shortcut button successfully triggers the main save action.  
- Any exceptions during the save process are handled gracefully, ensuring the application remains stable.  
  
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 Proposal Form

Type: CONTROL

Title: Validate Proposal Form  
  
Acceptance Criteria:  
- When the "Validate" button is pressed, the system should navigate to the "Proposal Form" and trigger the validation process.  
- The validation process should be executed without causing any errors, even if unexpected issues occur.  
  
Definition of Done:  
- The "Validate" button is present on the application details canvas.  
- Pressing the "Validate" button successfully initiates the validation process for the proposal form.  
- The system handles any exceptions gracefully without crashing or displaying error messages 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.

# View All Images Shortcut Button

Type: CONTROL

Title: View All Images Shortcut Button  
  
Acceptance Criteria:  
- When the user clicks the "All Images Shortcut" button, the system should navigate to the "View All Images" section of the proposal form.  
- The system should execute any necessary triggers associated with the "View All Images" section.  
- If an error occurs during the navigation or trigger execution, the system should handle the error gracefully without crashing.  
  
Definition of Done:  
- The "All Images Shortcut" button is implemented and visible on the user interface.  
- Clicking the button successfully navigates the user to the "View All Images" section.  
- All associated triggers are executed without errors.  
- Error handling is in place to manage any unexpected issues during the 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.

# Navigation based on Age Proof Type

Type: CONTROL

Title: Navigation based on Age Proof Type  
  
Acceptance Criteria:  
1. If the selected age proof type is 'PP' (Passport), the system should navigate to the Passport Details section.  
2. If the selected age proof type is 'DL' (Driver's License), the system should navigate to the Driver's License Details section.  
3. If the selected age proof type is 'VI' (Voter's ID), the system should navigate to the Voter's ID Details section.  
4. If none of the above conditions are met, the system should navigate to the Suspected Case section.  
  
Definition of Done:  
- The system correctly navigates to the appropriate section based on the selected age proof type.  
- The navigation logic is thoroughly tested and verified.  
- The user interface reflects the correct section based on the user's selection.  
- 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 any direct database CRUD operations.

# Confirm Relationship Status for Duplicate Number

Type: CONTROL

Title: Confirm Relationship Status for Duplicate Number  
  
Acceptance Criteria:  
1. If the user selects 'CORRECT\_NUMBER' from the list:  
 - Display a message prompting the user to approve the case from a supervisor.  
 - Enable the 'Approve' button if it is currently disabled.  
 - Make the 'Supervisor Check' checkbox visible and enabled if it is currently hidden.  
 - Disable the 'Save' button if it is currently enabled.  
2. If the user selects any other option:  
 - Disable the 'Approve' button if it is currently enabled.  
 - Hide the 'Supervisor Check' checkbox if it is currently visible.  
 - Enable the 'Save' button if it is currently disabled.  
  
Definition of Done:  
- The user can select an option from the list to confirm the relationship status.  
- The system performs the appropriate actions based on the selected option.  
- The user receives a message if 'CORRECT\_NUMBER' is selected.  
- The 'Approve' button, 'Supervisor Check' checkbox, and 'Save' button are enabled or disabled as per the conditions 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.

# Signature Decision Options

Type: CONTROL

Title: Signature Decision Options  
  
Acceptance Criteria:  
1. The decision options should be presented as a list with the following values:  
 - "8" for approved  
 - "LA" for pending  
 - "No" for rejected  
2. The list should be displayed in a user-friendly interface with clear labels and appropriate spacing.  
3. The decision should be validated to ensure it is selected from the provided list.  
  
Definition of Done:  
1. The user interface displays a list of decision options for the signature.  
2. The user can select one of the options from the list.  
3. The selected decision is validated and saved correctly.  
4. The interface is visually consistent with the rest of the application and meets the specified design requirements.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include specific database queries.

# Partner Type Selection and PAN Validation Navigation

Type: CONTROL

Title: Partner Type Selection and PAN Validation Navigation  
  
Acceptance Criteria:  
1. When the user selects a partner type from the list, the system should automatically navigate to the PAN validation field.  
2. The system should then execute the trigger associated with the button press event.  
  
Definition of Done:  
1. The user can see a dropdown list with the following options: IP, PH, 5, 8, 10, and SL.  
2. Upon selecting an option, the system navigates to the PAN validation field and executes the necessary trigger.  
3. 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 any database-specific queries.

# Validate and Process Annual Amount to be Invested

Type: CONTROL

Title: Validate and Process Annual Amount to be Invested  
  
Acceptance Criteria:  
1. The system should validate that the entered amount is a valid number. If not, an error message should be displayed: "Please enter a valid number in Amount Invested".  
2. If the solution name is set to 1, the system should calculate the nominee's age based on the nominee's date of birth and the current date.  
3. If the calculated nominee's age is 12 or older, an error message should be displayed: "For the solution selected, the nominee has to be a minor with age less than 12."  
4. The system should populate the solution product details.  
5. The system should navigate to the "Rider Details" section and set the "Total Rider Invest" field based on the sum assured values.  
6. The "Total Rider Invest" field should be set to read-only for the current record if the rider cover code starts with 'L'.  
7. The system should navigate to the "Total Rider Invest" field after processing.  
  
Definition of Done:  
- The user can input the annual amount to be invested.  
- The system performs all necessary validations and calculations as per the acceptance criteria.  
- Appropriate error messages are displayed for invalid inputs.  
- The system navigates to the "Rider Details" section and updates the "Total Rider Invest" field as required.  
- The "Total Rider Invest" field is set to read-only for the current record if applicable.  
- The user is navigated to the "Total Rider Invest" field after processing.  
  
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 Insert EIA Details for New Applicant

Type: CONTROL

Title: Verify and Insert EIA Details for New Applicant  
  
Acceptance Criteria:  
1. The system should check if an EIA exists for the given application number and if the account type is 'New\_Applicant'.  
2. If the EIA exists, the system should:  
 - Verify the existence of the applicant's details in the DEQC table.  
 - Retrieve and store various internal values (state, nation, bank) based on the applicant's address and bank details.  
 - Insert the applicant's details into the DEQC table if they do not already exist.  
 - Open a webpage with the applicant's details.  
3. If the EIA does not exist, the system should display an error message indicating that the application number is not registered for an e-Insurance Account.  
  
Definition of Done:  
- The system correctly verifies the existence of an EIA for the given application number.  
- The system accurately retrieves and stores internal values for state, nation, and bank based on the applicant's details.  
- The system successfully inserts the applicant's details into the DEQC table if they do not already exist.  
- The system opens the appropriate webpage with the applicant's details upon successful insertion.  
- An error message is displayed if the application number is not registered for an e-Insurance Account.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Check if EIA exists:  
 ```sql  
 SELECT COUNT(1)  
 INTO v\_exists  
 FROM azbj\_eia\_details  
 WHERE application\_no = :correct\_value.appln\_no  
 AND top\_indicator = 'Y'  
 AND eia\_account\_type = 'New\_Applicant';  
 ```  
  
- Check if applicant details exist in DEQC table:  
 ```sql  
 SELECT COUNT(1)  
 INTO v\_exists  
 FROM azbj\_eia\_deqc\_de\_details  
 WHERE application\_no = :correct\_value.appln\_no  
 AND top\_indicator = 'Y';  
 ```  
  
- Retrieve internal values for state, nation, and bank:  
 ```sql  
 SELECT internal\_value  
 INTO v\_state  
 FROM inf\_dnm\_poplists  
 WHERE poplist\_code = 'EIA\_STATES'  
 AND UPPER(screen\_value) = UPPER(:correct\_value.ph\_address\_line5);  
  
 SELECT internal\_value  
 INTO v\_m\_state  
 FROM inf\_dnm\_poplists  
 WHERE poplist\_code = 'EIA\_STATES'  
 AND UPPER(screen\_value) = UPPER(:correct\_value.ph\_m\_address\_line5);  
  
 SELECT internal\_value  
 INTO v\_nation  
 FROM inf\_dnm\_poplists  
 WHERE poplist\_code = 'EIA\_COUNTRY'  
 AND UPPER(screen\_value) = UPPER(:correct\_value.ph\_country\_name);  
  
 SELECT internal\_value  
 INTO v\_m\_nation  
 FROM inf\_dnm\_poplists  
 WHERE poplist\_code = 'EIA\_COUNTRY'  
 AND UPPER(screen\_value) = UPPER(:correct\_value.ph\_m\_country\_name);  
  
 SELECT internal\_value  
 INTO v\_bank  
 FROM inf\_dnm\_poplists  
 WHERE poplist\_code = 'EIA\_BANKS'  
 AND UPPER(screen\_value) = UPPER(:correct\_value.bank\_name);  
 ```  
  
- Insert applicant details into DEQC table:  
 ```sql  
 INSERT INTO azbj\_eia\_deqc\_de\_details (  
 application\_no, policy\_no, module\_flag, 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, top\_indicator, create\_user, agent\_code  
 ) VALUES (  
 :correct\_value.appln\_no, NULL, 'DEQC', SUBSTR(:correct\_value.ph\_name, 1, 25), SUBSTR(:correct\_value.ph\_middle\_name, 1, 25), SUBSTR(:correct\_value.ph\_surname, 1, 25), CASE WHEN :correct\_value.ph\_father IS NULL THEN SUBSTR(:correct\_value.ph\_spouse, 1, 30) ELSE SUBSTR(:correct\_value.ph\_father, 1, 30) END, :correct\_value.ph\_gender, :correct\_value.ph\_dob, SUBSTR(:correct\_value.ph\_age\_proof, 1, 3), NULL, SUBSTR(:correct\_value.ph\_pan, 1, 10), SUBSTR(:correct\_value.ph\_door\_no, 1, 25), SUBSTR(:correct\_value.ph\_building\_name, 1, 25), SUBSTR(:correct\_value.ph\_plot\_no, 1, 25), SUBSTR(:correct\_value.ph\_area, 1, 25), SUBSTR(:correct\_value.ph\_address\_line4, 1, 25), SUBSTR(:correct\_value.ph\_pin, 1, 10), v\_state, v\_nation, SUBSTR(:correct\_value.ph\_residence\_proof, 1, 3), SUBSTR(:correct\_value.ph\_m\_door\_no, 1, 25), SUBSTR(:correct\_value.ph\_m\_building\_name, 1, 25), SUBSTR(:correct\_value.ph\_m\_plot\_no, 1, 25), SUBSTR(:correct\_value.ph\_m\_area, 1, 25), SUBSTR(:correct\_value.ph\_m\_address\_line4, 1, 25), SUBSTR(:correct\_value.ph\_m\_pin, 1, 10), v\_m\_state, v\_m\_nation, SUBSTR(:correct\_value.ph\_m\_residence\_proof, 1, 3), SUBSTR(:correct\_value.ph\_phone, 1, 13), SUBSTR(:correct\_value.ph\_m\_phone, 1, 13), SUBSTR(:correct\_value.ph\_mobile, 1, 11), SUBSTR(:correct\_value.ph\_email, 1, 50), CASE WHEN UPPER(:correct\_value.acc\_type) LIKE '%SAVI%' THEN 'S' ELSE 'C' END, SUBSTR(:correct\_value.bank\_account\_no, 1, 30), v\_bank, SUBSTR(:correct\_value.bank\_branch, 1, 30), SUBSTR(:correct\_value.micr\_code, 1, 15), SUBSTR(:correct\_value.ifsc\_code, 1, 11), NULL, 'Y', USER, :correct\_value.FSC\_CODE  
 );  
 ```  
  
- Open webpage with applicant's details:  
 ```sql  
 web.show\_document(  
 v\_call\_url || 'p\_appln\_no=' || :correct\_value.appln\_no || '&p\_policy\_no=' || NVL(:correct\_value.new\_policy\_ref, '0') || '&p\_module\_flag=DEQC&p\_call\_url\_red=' || v\_call\_url\_red || '&p\_user\_id=' || USER,  
 '\_blank'  
 );  
 ```  
  
- Display error message if EIA does not exist:  
 ```sql  
 azbj\_message('E', :correct\_value.appln\_no || 'is not registered for e-Insurance Account');  
 ```

# User can select relationship status with staff from a dropdown list

Type: CONTROL

Title: User can select relationship status with staff from a dropdown list  
  
Acceptance Criteria:  
1. The user should be able to see a dropdown list labeled "Relation with Staff" on the form.  
2. The dropdown list should contain the following options:  
 - IP  
 - 5  
 - 8  
 - LA  
 - No  
3. The dropdown list should be enabled and allow the user to select an option.  
4. The selected option should be saved and updated in the database when the form is submitted.  
  
Definition of Done:  
1. The dropdown list is visible and correctly labeled as "Relation with Staff".  
2. The dropdown list contains all the specified options.  
3. The user can select an option from the dropdown list.  
4. The selected option is saved and updated in the database upon form submission.  
5. The form is tested to ensure that the dropdown list functions correctly and the data is accurately recorded.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include specific database queries.

# Navigation based on Age Proof Type

Type: CONTROL

Title: Navigation based on Age Proof Type  
  
Acceptance Criteria:  
1. When the user provides 'PP' (Passport) as the age proof, the system should navigate to the Passport Details section.  
2. When the user provides 'DL' (Driver's License) as the age proof, the system should navigate to the Driver's License Details section.  
3. When the user provides 'VI' (Voter's ID) as the age proof, the system should navigate to the Voter's ID Details section.  
4. If none of the above age proofs are provided, the system should navigate to the next tab in the process.  
  
Definition of Done:  
- The system correctly navigates to the appropriate details section based on the age proof provided.  
- The navigation logic is thoroughly tested and verified to ensure it works as expected.  
- The user interface is updated to reflect the navigation 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 direct database CRUD operations.

# Manage Multiple Nominees for an Application

Type: CONTROL

Detailed description: As a user, I want to manage multiple nominees for an application, so that I can ensure all nominee details are accurately captured and validated.  
  
Acceptance criteria:  
1. When the "Multiple Nominee" button is pressed, the system should:  
 - Check if there are existing nominees for the given application number with a top indicator of 'Y'.  
 - If nominees exist, retrieve and display the nominee details (name, birthplace, date of birth, relation, and gender) for the first nominee.  
 - Calculate the age of each nominee and identify if any nominee is a minor (under 18 years old).  
 - If a minor nominee is found, prompt the user to enter appointee details.  
 - Enable or disable nominee-related fields based on the presence of minor nominees.  
  
2. The system should handle exceptions gracefully and display an appropriate message if an error occurs during the process.  
  
Definition of Done:  
- The "Multiple Nominee" button functionality is implemented and tested.  
- The system correctly retrieves and displays nominee details.  
- The system accurately identifies minor nominees and prompts for appointee details.  
- All nominee-related fields are enabled or disabled as required.  
- Exception handling is in place, and appropriate error messages are displayed.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Retrieve nominee details:  
 ```sql  
 SELECT nominee\_name, nominee\_birthplace, nominee\_dob, nominee\_relation, nominee\_gender  
 FROM azbj\_multiple\_nominee\_dtls  
 WHERE application\_no = :correct\_value.appln\_no  
 AND top\_indicator = 'Y'  
 AND nominee\_number = 1;  
 ```  
  
- Count the number of nominees:  
 ```sql  
 SELECT COUNT(1)  
 INTO v\_nom\_count  
 FROM azbj\_multiple\_nominee\_dtls  
 WHERE application\_no = :correct\_value.appln\_no  
 AND top\_indicator = 'Y';  
 ```  
  
- Check for multiple nominees:  
 ```sql  
 SELECT CASE WHEN COUNT() > 1 THEN 'Y' ELSE 'N' END  
 INTO v\_multiple\_nominee\_flag  
 FROM azbj\_multiple\_nominee\_dtls  
 WHERE application\_no = :correct\_value.appln\_no  
 AND top\_indicator = 'Y';  
 ```

# Validate Product Code and Adjust Field Visibility Based on Solution Name

Type: CONTROL

Detailed description: As a user, I want the system to validate the selected product code and adjust the visibility and enabled state of certain fields based on the solution name provided, so that I can ensure only allowed solution products are used and the form fields are appropriately displayed and enabled.  
  
Acceptance criteria:  
1. When the solution name is changed, the system should:  
 - Check if the product code exists in the `azbj\_solution\_config\_mst` table with a product type of 'M'.  
 - If the product code does not exist, display an error message: "This Product is not allowed Solution Product."  
 - If the solution name is null, the system should:  
 - Hide the "COVER\_SOLUTION" tab.  
 - Hide the "INVESTED\_AMOUNT" field.  
 - Enable the "ASSIGN\_FLAG" field.  
 - If the solution name is not null, the system should:  
 - Show and enable the "COVER\_SOLUTION" tab.  
 - Show and enable the "INVESTED\_AMOUNT" field.  
 - Disable the "ASSIGN\_FLAG" field.  
  
Definition of Done:  
- The system correctly validates the product code against the `azbj\_solution\_config\_mst` table.  
- Appropriate error messages are displayed when the product code is not allowed.  
- The visibility and enabled state of the "COVER\_SOLUTION" tab, "INVESTED\_AMOUNT" field, and "ASSIGN\_FLAG" field are correctly adjusted based on the solution name.  
- All changes are tested and verified to ensure they meet the acceptance criteria.  
  
DB queries for Table reference CRUD operations only(With Usage):  
```sql  
-- Query to check if the product code exists in the azbj\_solution\_config\_mst table  
SELECT COUNT(product\_id)  
INTO v\_sol\_cnt  
FROM azbj\_solution\_config\_mst  
WHERE product\_id = :correct\_value.product\_code AND product\_type = 'M';  
```

# Initiate E-KYC (AADHAAR) Verification

Type: CONTROL

Title: Initiate E-KYC (AADHAAR) Verification  
  
Acceptance Criteria:  
1. When the user presses the "E-KYC(AADHAAR) DET" button, the system should:  
 - Check if a parameter list named 'Param1' already exists and destroy it if it does.  
 - Create a new parameter list named 'Param1'.  
 - Add the AADHAAR number and module name ('DEQC') to the parameter list.  
 - Increment the AADHAAR count by 1.  
 - Call the form 'AZBJ\_ADHAAR\_KYC' with the created parameter list.  
  
2. When the user navigates to the next item using the keyboard, the system should:  
 - Move the focus to the item 'CORRECT\_VALUE.AGENT\_NEXT\_TAB'.  
  
Definition of Done:  
- The button "E-KYC(AADHAAR) DET" is visible and enabled for user interaction.  
- The E-KYC process is initiated correctly upon button press, with the appropriate parameters passed and the form 'AZBJ\_ADHAAR\_KYC' called.  
- The focus moves to the next item 'CORRECT\_VALUE.AGENT\_NEXT\_TAB' when the user navigates using the keyboard.  
- 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.

# Exit Button Functionality with Signature Decision Verification

Type: CONTROL

Title: Exit Button Functionality with Signature Decision Verification  
  
Acceptance Criteria:  
1. When the exit button is pressed, the system should check if a signature decision has been made.  
2. If a signature decision is present:  
 - For non-desktop modules, the system should retrieve the contract ID associated with the application number.  
 - If a contract ID is found, the system should delete any existing signature verification records for that contract ID.  
 - The system should then insert a new record into the signature verification table with the contract ID, signature decision, and other relevant details.  
3. If no signature decision is present, the system should display an error message prompting the user to select a signature decision.  
4. The system should navigate to the signature population section after processing the signature decision.  
  
Definition of Done:  
- The exit button functionality is implemented and tested.  
- The system correctly handles the presence or absence of a signature decision.  
- Appropriate database operations (select, delete, insert) are performed based on the signature decision.  
- Error messages are displayed when necessary.  
- Navigation to the signature population section is functional.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Select contract ID:  
 ```sql  
 SELECT cont\_id INTO v\_contract\_id   
 FROM azbj\_batch\_items   
 WHERE application\_no = :correct\_value.appln\_no   
 AND rownum < 2;  
 ```  
- Delete existing signature verification:  
 ```sql  
 DELETE FROM azbj\_sign\_address\_verification  
 WHERE contract\_id = v\_contract\_id  
 AND column\_name = 'SIGNATURE';  
 ```  
- Insert new signature verification:  
 ```sql  
 INSERT INTO azbj\_sign\_address\_verification (  
 contract\_id,  
 column\_name,  
 verified\_flag,  
 verified\_user,  
 verified\_date,  
 column\_desc  
 ) VALUES (  
 v\_contract\_id,  
 'SIGNATURE',  
 pk\_vars.v\_match\_ic\_sig,  
 USER,  
 SYSDATE,  
 :control.sign\_decision  
 );  
 ```

# User selection for IP Aadhar field

Type: CONTROL

Title: User selection for IP Aadhar field  
  
Acceptance Criteria:  
- The radio group should have options "IP" and "No".  
- The selected option should be stored and retrievable for further processing.  
  
Definition of Done:  
- The radio group is displayed on the form.  
- The user can select either "IP" or "No".  
- The selected value is saved correctly and can be retrieved when needed.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include specific CRUD operations.

# PAN Validation and Aadhaar Linking

Type: CONTROL

Detailed description: As a user, I want to validate the PAN (Permanent Account Number) entered in the system to ensure it meets the required format and is valid according to the business rules.  
  
Acceptance criteria:  
1. The PAN field should be validated to ensure it is exactly 10 characters long.  
2. The PAN should match the regular expression pattern: `^[A-Z]{3}[C,P,H,F,A,T,B,L,J,G][A-Z][0-9]{4}[A-Z]`.  
3. If the PAN is invalid, an error message should be displayed, and the PAN field should be cleared.  
4. If the PAN is valid, the system should check if the PAN matches the proposal type and update the status accordingly.  
5. The system should handle different proposal types and validate the PAN against specific criteria for each type.  
6. If the PAN is valid, the system should update the relevant fields with the PAN details and mark the PAN as valid.  
7. If the PAN is invalid, the system should display an appropriate error message and clear the PAN details.  
8. The system should log the PAN validation process and any errors encountered during validation.  
9. The system should handle the linking of PAN with Aadhaar and display the status of the linking process.  
  
Definition of Done:  
- The PAN validation functionality is implemented and tested.  
- The system correctly validates the PAN according to the specified criteria.  
- Appropriate error messages are displayed for invalid PAN entries.  
- The PAN details are updated correctly for valid PAN entries.  
- The PAN validation process is logged, and any errors are recorded.  
- The PAN-Aadhaar linking process is implemented and displays the correct status.  
- The functionality is reviewed and approved by the stakeholders.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- The following SQL queries are used for validating and updating PAN details:  
 ```sql  
 SELECT 'Y'  
 INTO v\_ver\_pan  
 FROM DUAL  
 WHERE REGEXP\_LIKE(:control.ph\_pan\_card\_no, '^[A-Z]{3}[C,P,H,F,A,T,B,L,J,G][A-Z][0-9]{4}[A-Z]');  
  
 SELECT CASE WHEN COUNT(1) > 0 THEN 'Y' ELSE 'N' END  
 INTO v\_valid\_pancard  
 FROM AZBJ\_PROPOSAL\_TYPE  
 WHERE INTERNAL\_VALUE = :CORRECT\_VALUE.PROPTYPE  
 AND 1 = CASE  
 WHEN :CORRECT\_VALUE.PROPTYPE IN ('I', 'N', 'F', 'P', 'JL', 'Q', 'R', 'NPS', 'BDP', 'EIA', 'NGST', 'MWP')  
 AND UPPER(v\_digit) IN ('P') THEN 1  
 WHEN :CORRECT\_VALUE.PROPTYPE IN ('K', 'E', 'GS')  
 AND UPPER(v\_digit) IN ('C', 'F', 'G', 'L', 'A', 'B', 'T') THEN 1  
 WHEN :CORRECT\_VALUE.PROPTYPE IN ('H')  
 AND UPPER(v\_digit) IN ('H') THEN 1  
 WHEN :CORRECT\_VALUE.PROPTYPE IN ('PS')  
 AND UPPER(v\_digit) IN ('F') THEN 1  
 WHEN :CORRECT\_VALUE.PROPTYPE IN ('TF')  
 AND UPPER(v\_digit) IN ('T', 'A') THEN 1  
 ELSE 0  
 END;  
  
 SELECT scrutiny\_no  
 INTO v\_scrutiny\_no  
 FROM azbj\_phub\_scrutiny\_prop  
 WHERE application\_no = :correct\_value.appln\_no;  
  
 SELECT pan\_validation\_flag, partner\_id  
 INTO v\_pan\_validation\_flag, v\_partner\_id  
 FROM azbj\_pan\_validation\_dump  
 WHERE application\_no = :correct\_value.appln\_no;  
  
 SELECT middle\_name, first\_name, surname  
 INTO :control.ph\_pan\_mid\_name, :control.ph\_pan\_fir\_name, :control.ph\_pan\_last\_name  
 FROM cp\_partners  
 WHERE part\_id = v\_partner\_id;  
  
 INSERT INTO customer.azbj\_phub\_scrutiny\_proof\_hist (CATEGORY, FIRST\_NAME, MIDDLE\_NAME, LAST\_NAME, MEMBERSHIP\_NO, STATUS, ADDRESS, SCRUTINY\_NO, VERSION\_NO, APPLICATION\_NO, CREATE\_DATE, DOC\_EXP\_DATE, DOC\_TYPE, NSDL\_BALANCE\_ID, NSDL\_STATUS, HIST\_DATE)  
 SELECT CATEGORY, FIRST\_NAME, MIDDLE\_NAME, LAST\_NAME, MEMBERSHIP\_NO, STATUS, ADDRESS, SCRUTINY\_NO, VERSION\_NO, APPLICATION NO, CREATE\_DATE, DOC\_EXP\_DATE, DOC\_TYPE, NSDL\_BALANCE\_ID, NSDL\_STATUS, SYSDATE  
 FROM azbj\_phub\_scrutiny\_proof  
 WHERE APPLICATION\_NO = :correct\_value.appln\_no;  
  
 UPDATE azbj\_phub\_scrutiny\_proof  
 SET CATEGORY = 'PANCARD', first\_name = :control.ph\_pan\_fir\_name, middle\_name = :control.ph\_pan\_mid\_name, last\_name = :control.ph\_pan\_last\_name, membership\_no = :control.ph\_pan\_card\_no, status = :control.ph\_pan\_curr\_status, scrutiny\_no = v\_version\_no, nsdl\_status = v\_nsdl\_status  
 WHERE APPLICATION\_NO = :correct\_value.appln\_no AND partner\_type = :CONTROL.IP\_PH;  
  
 INSERT INTO azbj\_phub\_scrutiny\_proof (CATEGORY, first\_name, middle\_name, last\_name, membership\_no, status, scrutiny\_no, version\_no, application\_no, nsdl\_status, partner\_type)  
 VALUES ('PANCARD', :control.ph\_pan\_fir\_name, :control.ph\_pan\_mid\_name, :control.ph\_pan\_last\_name, :control.ph\_pan\_card\_no, :control.ph\_pan\_curr\_status, v\_scrutiny\_no, v\_version\_no, :correct\_value.appln\_no, v\_nsdl\_status, :control.ip\_ph);  
 ```

# Implement PH\_AADHAR Radio Group

Type: CONTROL

Title: Implement PH\_AADHAR Radio Group  
  
Acceptance Criteria:  
1. The PH\_AADHAR field should be presented as a radio group with the following options:  
 - Option 1: "PH" with a value of "PH"  
 - Option 2: "5" with a value of "5"  
 - Option 3: "No" with a value of "SL"  
2. The radio group should be displayed on the "PH\_INFO" tab page within the "IMAGES" canvas.  
3. The "IMAGES" canvas should be part of the "WINDOW2" window, which has the title "DE Quality Check" and specific dimensions and properties.  
4. The "IMAGE" canvas should be part of the "WINDOW1" window, which has the title "DE Stripes" and specific dimensions and properties.  
  
Definition of Done:  
- The PH\_AADHAR field is correctly displayed as a radio group with the specified options.  
- The radio group is located on the "PH\_INFO" tab page within the "IMAGES" canvas.  
- The "IMAGES" canvas is correctly associated with the "WINDOW2" window.  
- The "IMAGE" canvas is correctly associated with the "WINDOW1" window.  
- All specified properties and dimensions for the canvases and windows are correctly implemented.

# Supervisor Approval for Duplicate Contacts

Type: CONTROL

Title: Supervisor Approval for Duplicate Contacts  
  
Acceptance Criteria:  
1. When the "Supervisor Approval" button is pressed, the system should check if at least one policy is selected for approval.  
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 specific CRUD operations.

# Display and Populate CRPF/Defence Questions

Type: CONTROL

Title: Display and Populate CRPF/Defence Questions  
  
Acceptance Criteria:  
1. When the button labeled "CRPF/Defence Questions" is pressed, a new window should be displayed.  
2. The new window should show a specific view related to CRPF/Defence questions.  
3. The system should populate the questions based on the user's gender, part ID, and full name (including middle name and surname).  
4. The process should handle any exceptions gracefully without crashing.  
  
Definition of Done:  
- The button labeled "CRPF/Defence Questions" is functional and triggers the display of the new window.  
- The new window correctly shows the view related to CRPF/Defence questions.  
- The questions are populated accurately based on the user's gender, part ID, and full name.  
- The system handles exceptions without causing any errors or crashes.  
  
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.

# Display and Populate CRPF/Defence Questions

Type: CONTROL

Title: Display and Populate CRPF/Defence Questions  
  
Acceptance Criteria:  
1. When the button labeled "CRPF/Defence Questions" is pressed, a new window should be displayed.  
2. The new window should show the relevant questions related to CRPF/Defence.  
3. The system should populate the questions based on the user's gender, part ID, and full name (including middle name and surname).  
4. The system should handle any exceptions gracefully without crashing.  
  
Definition of Done:  
- The button labeled "CRPF/Defence Questions" is visible and functional.  
- Pressing the button opens a new window displaying the relevant questions.  
- The questions are populated correctly based on the user's details.  
- The system handles exceptions without crashing or displaying errors to the user.  
  
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.

# Delete Specific Rider Details from Proposal

Type: CONTROL

Title: Delete Specific Rider Details from Proposal  
  
Acceptance Criteria:  
1. When the delete button is pressed, the system should navigate to the "AZBJ\_RIDER\_DETAILS2" section and identify the total number of records.  
2. The system should iterate through each record in the "AZBJ\_RIDER\_DETAILS2" section.  
3. For each record, if the rider cover code starts with 'R' and the delete check flag is 'Y', the system should:  
 - Navigate to the corresponding record in the "AZBJ\_RIDER\_DETAILS" section.  
 - Delete the record if the rider cover code does not start with 'L' and matches the rider cover code in "AZBJ\_RIDER\_DETAILS2".  
 - Navigate back to the "AZBJ\_RIDER\_DETAILS2" section and delete the record.  
4. The system should continue this process until all relevant records are checked and deleted.  
  
Definition of Done:  
- The delete functionality should be implemented and tested to ensure it correctly deletes the specified rider details based on the given conditions.  
- The system should handle navigation between sections and records seamlessly.  
- All relevant records should be deleted, and the remaining records should be intact.  
- The functionality should be 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 contains Oracle Forms-specific constructs that cannot be directly executed in the database without modification.

# Implement Home Loan Type Dropdown

Type: CONTROL

Detailed description: As a user, I want to select the type of home loan from a predefined list so that I can accurately specify the loan type in the application.  
  
Acceptance criteria:  
1. The home loan type field should be a dropdown list.  
2. The dropdown list should contain the following options:  
 - Option 1: "5"  
 - Option 2: "8"  
 - Option 3: "SL" (No)  
3. The field should be positioned at coordinates (93, 139) on the form.  
4. The field should have a width of 95 and a height of 15.  
5. The field should be right-justified and have a background color of gray.  
6. The field should allow both insertion and updates.  
7. The field should display a prompt labeled "Home Loan Type" in bold, plain Tahoma font with a size of 8.  
  
Definition of Done:  
- The home loan type dropdown list is implemented and displays the correct options.  
- The field is correctly positioned and styled as specified.  
- The field allows users to insert and update values.  
- The prompt for the field is correctly displayed and formatted.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include specific database queries.

# Aadhar Card Verification

Type: CONTROL

Title: Aadhar Card Verification  
  
Acceptance Criteria:  
1. The system should enable the Aadhar card input field if it is currently disabled.  
2. If the age proof type is not 'AC' or 'ACS', or if the age proof ID is null, the system should display a message indicating that the Aadhar card data is not available.  
3. The system should check if the Aadhar card number exists in the database.  
4. If the Aadhar card number does not exist in the database, the system should display a message indicating that the data is not available.  
5. If the Aadhar card number exists, the system should retrieve and display the associated details.  
6. The system should call a form to verify the Aadhar card details.  
7. Based on the verification result, the system should display whether the details matched, did not match, or if no details were found.  
  
Definition of Done:  
1. The Aadhar card verification functionality is implemented and tested.  
2. The system correctly enables the Aadhar card input field when necessary.  
3. Appropriate messages are displayed when the Aadhar card data is not available or does not exist in the database.  
4. The system retrieves and displays the correct Aadhar card details from the database.  
5. The form for verifying Aadhar card details is called and functions correctly.  
6. The verification result is accurately displayed to the user.  
7. All acceptance criteria are met, and the functionality is reviewed and approved.  
  
DB queries for Table reference CRUD operations only (With Usage):  
```sql  
-- Check if Aadhar card number exists in the database  
SELECT COUNT(1)  
INTO v\_cnt  
FROM balic\_aadhaar\_dtls  
WHERE aadhaar\_aadhaarnumber = :correct\_value.ph\_age\_proof\_id;  
```

# Populate Solution Details on Button Press

Type: CONTROL

Title: Populate Solution Details on Button Press  
  
Acceptance Criteria:  
1. When the button is pressed, the system should check if a parameter list named 'Param1' already exists.  
2. If 'Param1' exists, it should be destroyed and a new parameter list named 'Param1' should be created.  
3. The following parameters should be added to the parameter list:  
 - 'inception\_date' with a null value.  
 - 'POLICY\_REF' with a null value.  
 - 'APPLICATION\_NO' with the value from the field `correct\_value.appln\_no`.  
 - 'solution\_config\_id' with a null value.  
 - 'solution\_name' with the value from the field `control.SOLUTION\_NAME`.  
 - 'amount\_invested' with the value from the field `control.invested\_amount`.  
 - 'contract\_id' with the value from the variable `pk\_vars.v\_contract\_id`.  
 - 'solution\_id' with the value from the field `solution\_product.solution\_id`.  
 - 'calling\_from' with the value 'DEQC'.  
4. The system should then call the form 'AZBJ\_SOLN\_FORM\_CALL' with the created parameter list.  
  
Definition of Done:  
- The button should be functional and perform the described actions when pressed.  
- The parameters should be correctly set and passed to the called form.  
- The form 'AZBJ\_SOLN\_FORM\_CALL' should be opened with the provided parameters.  
- The solution details should be populated 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.

# Dropdown List for BFL TA Segment in BANK\_DETAILS Tab

Type: CONTROL

Title: Dropdown List for BFL TA Segment in BANK\_DETAILS Tab  
  
Acceptance Criteria:  
1. The segment selection should be available as a dropdown list with the following options:  
 - 5  
 - 8  
 - SL  
2. The dropdown list should be located at the specified position within the "BANK\_DETAILS" tab.  
3. The dropdown list should be initially hidden and only become visible based on specific conditions or user actions.  
4. The dropdown list should have a maximum length of 50 characters and a width of 95 units.  
5. The dropdown list should have a gray background and black text.  
6. The dropdown list should use the "MS Sans Serif" font with a size of 8 points and a medium weight.  
7. The prompt for the dropdown list should read "BFL TA Segment" and should be bold and plain in style.  
  
Definition of Done:  
- The dropdown list for selecting the BFL TA Segment is implemented and integrated within the "BANK\_DETAILS" tab.  
- The dropdown list meets all specified acceptance criteria.  
- The dropdown list is tested and verified to ensure it functions correctly and is displayed as intended.  
- 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.

# Aadhar Card Verification

Type: CONTROL

Title: Aadhar Card Verification  
  
Acceptance Criteria:  
1. When the Aadhar card verification button is pressed, the system should enable the Aadhar input field if it is currently disabled.  
2. If the age proof provided is not an Aadhar card or if the Aadhar card number is not provided, the system should display a message indicating that the data is not available.  
3. If the Aadhar card number is provided, the system should check the database to see if the Aadhar number exists.  
4. If the Aadhar number does not exist in the database, the system should display a message indicating that the data is not available.  
5. If the Aadhar number exists, the system should collect the necessary details (such as Aadhar number, name, address, district, state, pin code, gender, and date of birth) and pass them to another form for further processing.  
6. The system should then call another form to perform the Aadhar KYC verification.  
7. Based on the result of the KYC verification, the system should update the Aadhar input field with the appropriate status message: "Details matched", "Details do not match", or "None".  
  
Definition of Done:  
- The Aadhar card verification button should trigger the appropriate actions as described in the acceptance criteria.  
- The system should correctly handle cases where the Aadhar number is not provided or does not exist in the database.  
- The system should correctly pass the necessary details to the KYC verification form.  
- The system should correctly update the Aadhar input field based on the result of the KYC verification.  
- All functionalities should be tested and verified to ensure they work as expected.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Query to check if the Aadhar number exists in the database:  
 ```sql  
 SELECT COUNT(1)  
 INTO v\_cnt  
 FROM balic\_aadhaar\_dtls  
 WHERE aadhaar\_aadhaarnumber = :correct\_value.ip\_age\_proof\_id;  
 ```  
  
- Note: The above query is provided for reference and should be adapted to the specific database and application context.

# Close Window and Navigate to Profession Description

Type: CONTROL

Title: Close Window and Navigate to Profession Description  
  
User Story:  
As a user, I want to be able to close the current window and navigate to the profession description input field when I press the "Close" button, so that I can efficiently manage my workflow without unnecessary steps.  
  
Acceptance Criteria:  
1. When the "Close" button is pressed, the current window should be hidden.  
2. The view associated with the current window should also be hidden.  
3. The focus should then move to the profession description input field.  
  
Definition of Done:  
- The "Close" button successfully hides the current window.  
- The associated view is hidden upon pressing the "Close" button.  
- The focus is correctly set to the profession description input field after the window is closed.  
  
Dependent Block: CONTROL

# Hide Details View and Navigate to Income Proof on Exit Button Press

Type: CONTROL

Title: Hide Details View and Navigate to Income Proof on Exit Button Press  
  
Acceptance Criteria:  
1. When the "Exit" button is pressed, the details view should be hidden.  
2. The system should navigate to the income proof section after hiding the details view.  
  
Definition of Done:  
1. The "Exit" button hides the details view upon being pressed.  
2. The system successfully navigates to the income proof section after the details view 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 the provided XML content does not include any specific database CRUD operations.

# Manage SPW Percentage Field State Based on SPW Flag Checkbox

Type: CONTROL

Title: Manage SPW Percentage Field State 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 ensure the correct behavior of the "SPW Percentage" field based on the "SPW Flag" checkbox state.

# View and Interact with Child Covers Section

Type: CONTROL

Title: View and Interact with Child Covers Section  
  
Acceptance Criteria:  
1. When the "Child Covers" button is pressed, the system should display the "Child Covers" view.  
2. The focus should automatically move to the "Child Name" field within the "Child Covers" section.  
  
Definition of Done:  
- The "Child Covers" button is visible and functional.  
- Pressing the "Child Covers" button displays the "Child Covers" view.  
- The focus is set to the "Child Name" field when the view 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 specific database queries.

# User can select relationship to child from a predefined list

Type: CONTROL

Title: User can select relationship to child from a predefined list  
  
Acceptance Criteria:  
1. The list should include the following options: "Child", "IP", "5", "8", "10", "LA", and "No".  
2. The list should be displayed in a dropdown format.  
3. The selected relationship should be saved and displayed correctly in the system.  
  
Definition of Done:  
1. The dropdown list is implemented and includes all specified options.  
2. The user can select an option from the dropdown list.  
3. The selected option is saved and displayed correctly in the system.  
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 specific database 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 the following options:  
 - 5  
 - 8  
 - SL (No)  
2. The field should be positioned correctly on the "COVER\_DETAILS" tab.  
3. The field should be editable, allowing users to select any of the predefined options.  
4. The field should be displayed with a white background and black text.  
5. The field should be aligned to the right with a prompt "SPW Freq" displayed next to it.  
  
Definition of Done:  
1. The "SPW Freq" field is implemented and displays the correct options.  
2. The field is editable and allows users to select from the predefined list.  
3. The field is correctly positioned and styled as specified.  
4. The prompt "SPW Freq" is displayed next to the field and aligned to the right.  
  
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.

# Dropdown for Percentage Increase in SA

Type: CONTROL

Title: Dropdown for Percentage Increase in SA  
  
Acceptance Criteria:  
1. The user should see a dropdown list with the following options:  
 - 5%  
 - 8%  
 - No (represented as "SL")  
2. The dropdown list should be located on the "COVER\_DETAILS" tab.  
3. The dropdown list should be labeled as "% Increasing in SA" and the label should be right-aligned.  
4. The dropdown list should be initially invisible and only become visible under certain conditions (not specified in the provided XML).  
5. The user should be able to insert and update the selected value.  
  
Definition of Done:  
1. The dropdown list is implemented and displays the correct options.  
2. The dropdown list is correctly positioned on the "COVER\_DETAILS" tab.  
3. The label "% Increasing in SA" is right-aligned and visible when the dropdown list is visible.  
4. The dropdown list is initially invisible and becomes visible under the specified conditions.  
5. The user can successfully insert and update the selected value from the dropdown list.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include specific database queries.

# Save Questionnaire Answers

Type: CONTROL

Title: Save Questionnaire Answers  
  
Acceptance Criteria:  
1. When the "Save" button is pressed, the system should navigate to the "Questionnaire" section and count the number of records.  
2. The system should delete existing records in the `azbj\_fcf\_questionnaire` table where the application number, proposal status, question ID, and sub-question match specific criteria.  
3. The system should insert new records into the `azbj\_fcf\_questionnaire` table for each record in the "Questionnaire" section, with the appropriate values for each field.  
4. The system should commit the transaction to save the changes.  
5. A message should be displayed to the user indicating that the answers have been saved successfully.  
6. The "Questionnaire" window and view should be hidden, and the focus should be moved to the "Profession Description" field.  
  
Definition of Done:  
- The "Save" button functionality is implemented and tested.  
- The system correctly deletes and inserts records in the `azbj\_fcf\_questionnaire` table based on the specified criteria.  
- The transaction is committed, and a success message is displayed.  
- The "Questionnaire" window and view are hidden, and the focus is moved to the "Profession Description" field.  
- All acceptance criteria are met, and the feature is verified by QA.  
  
DB queries for Table reference CRUD operations only(With Usage):  
```sql  
-- Delete existing records  
DELETE FROM azbj\_fcf\_questionnaire  
WHERE appln\_no = :correct\_value.appln\_no  
 AND de\_flag = :parameter.proposal\_status  
 AND question\_id = 85  
 AND sub\_question IN (52, 53, 54, 55, 56);  
  
-- Insert new records  
INSERT INTO azbj\_fcf\_questionnaire (  
 contract\_id,  
 partition\_no,  
 question\_id,  
 sub\_question,  
 version,  
 part\_id,  
 answer,  
 description,  
 partner\_no,  
 question\_no,  
 appln\_no,  
 member\_no,  
 de\_flag  
) VALUES (  
 NULL,  
 NULL,  
 :blk\_ind\_quest.question\_id,  
 :blk\_ind\_quest.sub\_question,  
 NULL,  
 NULL,  
 :blk\_ind\_quest.correct\_value,  
 :blk\_ind\_quest.ans\_desc,  
 :blk\_ind\_quest.member\_no,  
 :blk\_ind\_quest.form\_question\_no,  
 :correct\_value.appln\_no,  
 :blk\_ind\_quest.member\_no,  
 :parameter.proposal\_status  
);  
```

# Generate BI Report on Button Click

Type: CONTROL

Title: Generate BI Report on Button Click  
  
Acceptance Criteria:  
1. When the 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 `control.bi` field is not null, the system should concatenate the BI link with the value of `control.bi` 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, the system should display an error message: "Please check the URL".  
5. The system should increment the `pk\_vars.v\_bi\_gen\_cnt` counter by 1.  
6. The system should update `pk\_vars.v\_bi\_url\_update` with the final URL.  
  
Definition of Done:  
- The button should successfully generate and open the BI report in a web browser.  
- Error handling should be in place to manage cases where the URL is not valid.  
- The counters and URL updates should be correctly incremented and stored.  
  
DB queries for Table reference CRUD operations only(With Usage):  
```sql  
SELECT sys\_desc  
INTO v\_bi\_link  
FROM azbj\_system\_constants  
WHERE sys\_TYPE = 'BI\_LINK'  
AND sys\_CODE = 'BI\_GENERATE';  
```

# Save Application Details

Type: CONTROL

Title: Save Application Details  
  
Acceptance Criteria:  
1. When the save button is pressed, the system should check if there are any existing records in the `azbj\_aml\_sip\_dtls` table with the same application number, module flag 'DEQC', and top indicator 'Y'.  
2. If such records exist, the system should update the top indicator of these records to 'N'.  
3. The system should then navigate to the 'SIP\_CONTROL' section and iterate through all records.  
4. For each record, if the `SIP\_PROOF\_TYPE` field is not null, a new record should be inserted into the `azbj\_aml\_sip\_dtls` table with the provided details.  
5. The system should display a message indicating the number of records found and another message confirming that the data has been saved successfully.  
6. The system should commit the transaction after all operations are completed.  
  
Definition of Done:  
- The save button functionality is implemented as described.  
- The system correctly updates existing records and inserts new records based on the conditions.  
- Appropriate messages are displayed to the user.  
- The transaction is committed successfully.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Check for existing records:  
 ```sql  
 SELECT COUNT()  
 INTO v\_count  
 FROM azbj\_aml\_sip\_dtls  
 WHERE application\_no = :correct\_value.appln\_no  
 AND MODULE\_FLAG = 'DEQC'  
 AND top\_indicator = 'Y';  
 ```  
  
- Update existing records:  
 ```sql  
 UPDATE azbj\_aml\_sip\_dtls  
 SET top\_indicator = 'N'  
 WHERE application\_no = :correct\_value.appln\_no  
 AND MODULE\_FLAG = 'DEQC'  
 AND top\_indicator = 'Y';  
 ```  
  
- Insert new records:  
 ```sql  
 INSERT INTO azbj\_aml\_sip\_dtls (application\_no, top\_indicator, action\_code, proof\_type, doc\_value, doc\_expiry\_date, module\_flag, sg\_premium, sg\_tasa)  
 VALUES (:correct\_value.appln\_no, 'Y', 'A', :SIP\_CONTROL.SIP\_PROOF\_TYPE, :SIP\_CONTROL.SIP\_FIELD\_VAL, :SIP\_CONTROL.SIP\_EXPIRY\_DATE, 'DEQC', :SIP\_CONTROL.SIP\_DERIVED\_INCOME, :SIP\_CONTROL.SIP\_DERIVED\_TASA);  
 ```

# PAN and Aadhar Link Verification Process

Type: CONTROL

Detailed description: As a user, I want to ensure that the PAN and Aadhar link verification process is correctly handled based on various conditions such as the user's nationality, age, and the presence of specific proofs, so that the system can accurately determine the mandatory requirements and proceed with the appropriate actions.  
  
Acceptance criteria:  
1. The system should check if the desktop flag is set to 'N'. If so, it should fetch the contract ID from the `azbj\_batch\_items` table using the application number and transaction type 'FRP'. If an error occurs during this process, an error message should be displayed.  
2. The system should determine the value of `v\_ip\_no` based on the `CHK\_INSURED` flag and the age of the user.  
3. The system should navigate to the 'azbj\_qc\_questions' block and iterate through the records to check if the question ID is 80 and if the answer is 'Y'. If so, it should set the mandatory flag to 'Y'.  
4. The system should determine the nationality and IP nationality flags based on the user's nationality.  
5. The system should check if the age proof is 'PP' (passport proof) and set the passport proof flag accordingly. If the passport proof flag is 'N', it should navigate to the 'AML' block and iterate through the records to check if any proof type is 'PP'.  
6. If the mandatory flag is 'Y', the system should create a parameter list with various parameters such as application number, IP number, contract ID, mandatory flag, passport proof flag, nationality, IP nationality, and property type. It should then call the 'AZBJ\_INCOMETAX\_QUEST\_DTLS' form with these parameters.  
7. If the mandatory flag is not 'Y', a warning message should be displayed indicating that the declaration under Income Tax Rules 2015 is not received.  
  
Definition of Done:  
- The system correctly fetches the contract ID when the desktop flag is 'N'.  
- The system accurately determines the value of `v\_ip\_no` based on the `CHK\_INSURED` flag and the user's age.  
- The system navigates to the 'azbj\_qc\_questions' block and sets the mandatory flag based on the question ID and answer.  
- The system correctly determines the nationality and IP nationality flags.  
- The system accurately checks the age proof and sets the passport proof flag, navigating to the 'AML' block if necessary.  
- The system creates a parameter list and calls the 'AZBJ\_INCOMETAX\_QUEST\_DTLS' form with the appropriate parameters when the mandatory flag is 'Y'.  
- The system displays a warning message if the mandatory flag is not 'Y'.  
  
DB queries for Table reference CRUD operations only(With Usage):  
```sql  
-- Fetch contract ID based on application number and transaction type  
SELECT cont\_id  
INTO v\_contract\_id  
FROM azbj\_batch\_items  
WHERE application\_no = :correct\_value.appln\_no  
AND transaction\_type = 'FRP'  
AND rownum = 1;  
```

# PAN Validation

Type: CONTROL

Detailed description: As a user, I want to validate the PAN (Permanent Account Number) entered in the system to ensure it meets the required format and is consistent with the proposal type, so that I can ensure the accuracy and validity of the PAN details.  
  
Acceptance criteria:  
1. The system should check if the PAN number is not null and has exactly 10 characters.  
2. The system should validate the PAN format using a regular expression to ensure it follows the pattern: 5 alphabets, 4 digits, and 1 alphabet.  
3. If the PAN number is invalid, the system should display an error message indicating the PAN number is invalid and clear the PAN input field.  
4. The system should check if the PAN number matches the proposal type and display a warning message if there is a mismatch.  
5. The system should update the PAN status to 'Valid' or 'Invalid' based on the validation results.  
6. The system should log the validation process and results for auditing purposes.  
7. The system should handle exceptions gracefully and log any errors encountered during the validation process.  
  
Definition of Done:  
- The PAN validation functionality is implemented and integrated into the system.  
- The validation logic is thoroughly tested to ensure it meets the acceptance criteria.  
- Error and warning messages are displayed appropriately based on the validation results.  
- The validation process and results are logged for auditing purposes.  
- The functionality is reviewed and approved by stakeholders.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- The system should use the following query to validate the PAN format:  
 ```sql  
 SELECT 'Y'  
 INTO v\_ver\_pan  
 FROM DUAL  
 WHERE REGEXP\_LIKE(UPPER(:control.ph\_pan\_card\_no), '^[A-Z]{3}[C,P,H,F,A,T,B,L,J,G][A-Z][0-9]{4}[A-Z]');  
 ```  
- The system should use 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 = :CORRECT\_VALUE.PROPTYPE  
 AND 1 = CASE  
 WHEN :CORRECT\_VALUE.PROPTYPE IN ('I', 'N', 'F', 'P', 'JL', 'Q', 'R', 'NPS', 'BDP', 'EIA', 'NGST', 'MWP') AND UPPER(v\_digit) IN ('P') THEN 1  
 WHEN :CORRECT\_VALUE.PROPTYPE IN ('K', 'E', 'GS') AND UPPER(v\_digit) IN ('C', 'F', 'G', 'L', 'A', 'B', 'T') THEN 1  
 WHEN :CORRECT\_VALUE.PROPTYPE IN ('H') AND UPPER(v\_digit) IN ('H') THEN 1  
 WHEN :CORRECT\_VALUE.PROPTYPE IN ('PS') AND UPPER(v\_digit) IN ('F') THEN 1  
 WHEN :CORRECT\_VALUE.PROPTYPE IN ('TF') AND UPPER(v\_digit) IN ('T', 'A') THEN 1  
 ELSE 0  
 END;  
 ```

# Manage Proposal Summary Details

Type: SUMMARY

Detailed description: As a user, I want to view and manage the summary details of a proposal, including personal information, product details, and financial information, so that I can ensure all necessary data is accurately captured and displayed.  
  
Acceptance criteria:  
1. The summary section should display the following personal information fields:  
 - FSC/IC Code  
 - Age  
 - Education  
 - Gender  
 - Occupation  
 - Date of Birth (DOB)  
 - PAN  
 - Annual Income  
 - Citizenship  
 - Age Proof  
 - Proof ID  
 - Income Proof  
 - Address Proof  
 - BMI  
 - Number of Early Deaths  
 - Signature Type  
 - Health Questions (if any)  
 - IT Declaration (if other than N)  
 - Premium Paid By  
  
2. The summary section should display the following product details:  
 - Product Name  
 - Package Code  
 - Benefit Term (BT)  
 - Premium Term (PT)  
 - Sum Assured  
 - Premium  
 - Fund Details  
 - NSAP Loading  
 - Riders and Sum Assured (SA)  
 - Renewal Premium  
 - Premium Frequency  
  
3. The fields should be displayed with appropriate prompts and should be read-only where applicable.  
  
4. The following fields should be hidden from view:  
 - Maturity FT  
 - Case Summary RCU  
 - Aizwal Flag  
 - PEP Flag  
 - Suspected CP  
  
5. The "Proceed" button should be available to navigate to the next section.  
  
6. The summary section should include a list of values (LOV) for selecting the product and package codes.  
  
Definition of Done:  
- The summary section is implemented and displays all the required fields with appropriate prompts.  
- Fields are read-only where specified.  
- Hidden fields are not visible to the user.  
- The "Proceed" button is functional and allows navigation to the next section.  
- LOVs for product and package codes are implemented and functional.  
- The implementation is tested and verified for accuracy and completeness.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- The following queries are used to populate LOVs for product and package codes:  
  
```sql  
-- Query for PACKAGE\_LOV  
SELECT   
FROM azbj\_package\_master   
WHERE product\_id = :correct\_value.product\_code;  
  
-- Query for PRODUCT\_LOV  
SELECT   
FROM (  
 SELECT product\_ext\_reference prod\_desc,  
 a.product\_id,  
 UPPER(description) description  
 FROM cfg\_v\_oc\_products\_api a, azbj\_product\_launch\_date b  
 WHERE a.product\_id >= 3  
 AND a.product\_id NOT IN (5, 7)  
 AND a.product\_id = b.product\_id  
 AND (b.closure\_date IS NULL OR MONTHS\_BETWEEN(SYSDATE, closure\_date) < 3)  
 AND a.product\_id NOT IN (  
 SELECT TO\_NUMBER(SUBSTR(char\_value, 2, LENGTH(char\_value) - 2))  
 FROM azbj\_system\_constants  
 WHERE UPPER(sys\_type) = 'CASHIER'  
 AND UPPER(sys\_code) = 'PRODUCT\_BLOCKED'  
 AND date\_value < SYSDATE)  
 AND a.product\_id IN (  
 SELECT PRODUCT\_ID  
 FROM AZBJ\_PRODUCT\_COVERS  
 WHERE (CASE  
 WHEN NVL(IL\_FLAG, 'N') IN ('Y', 'P') AND NVL(GROUP\_FLAG, 'N') <> 'Y' THEN 'UL'  
 WHEN NVL(IL\_FLAG, 'N') NOT IN ('Y', 'P') AND NVL(GROUP\_FLAG, 'N') <> 'Y' THEN 'TR'  
 WHEN NVL(GROUP\_FLAG, 'N') = 'Y' THEN 'GRP'  
 END) = (  
 SELECT (CASE  
 WHEN NVL(IL\_FLAG, 'N') IN ('Y', 'P') AND NVL(GROUP\_FLAG, 'N') <> 'Y' THEN 'UL'  
 WHEN NVL(IL\_FLAG, 'N') NOT IN ('Y', 'P') AND NVL(GROUP\_FLAG, 'N') <> 'Y' THEN 'TR'  
 WHEN NVL(GROUP\_FLAG, 'N') = 'Y' THEN 'GRP'  
 END)  
 FROM AZBJ\_PRODUCT\_COVERS  
 WHERE PRODUCT\_ID = :correct\_value.product\_code))  
ORDER BY a.product\_id;  
```

# Navigation and Tab Control with 'Proceed' Button

Type: SUMMARY

Detailed description: As a user, I want to navigate to the next tab and proceed to the 'CORRECT\_VALUE.PROPTYPE' field when I press the 'Proceed' button, so that I can continue my workflow seamlessly.  
  
Acceptance criteria:  
1. When the 'Proceed' button is pressed, the system should enable or disable the 'AGENT' tab based on specific conditions.  
2. After enabling or disabling the 'AGENT' tab, the system should automatically navigate to the 'CORRECT\_VALUE.PROPTYPE' field.  
  
Definition of Done:  
- The 'Proceed' button is functional and triggers the appropriate actions.  
- The 'AGENT' tab is enabled or disabled as required.  
- The system navigates to the 'CORRECT\_VALUE.PROPTYPE' field after the 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 direct database queries.

# Premium Frequency Selection

Type: SUMMARY

Title: Premium Frequency Selection  
  
Acceptance Criteria:  
1. The frequency selection should be a dropdown list with the following options:  
 - "5" with a value of "5"  
 - "8" with a value of "8"  
 - "No" with a value of "SL"  
2. The dropdown list should be displayed on the "SUMMARY" tab.  
3. The dropdown list should be positioned at the specified location on the screen.  
4. The dropdown list should be read-only, meaning users cannot insert or update the values.  
  
Definition of Done:  
1. The dropdown list for selecting premium frequency is implemented and visible on the "SUMMARY" tab.  
2. The dropdown list contains the options "5", "8", and "No" with corresponding values "5", "8", and "SL".  
3. The dropdown list is positioned correctly on the screen as per the design specifications.  
4. The dropdown list is read-only, and users are unable to modify the list items.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include any specific database queries.

# Gender Field Input and Selection

Type: SUMMARY

Title: Gender Field Input and Selection  
  
Acceptance Criteria:  
1. The gender field should display a prompt labeled "Gender".  
2. The gender field should be a text input field with a maximum length of 20 characters.  
3. The gender field should be case-insensitive and automatically convert input to uppercase.  
4. The gender field should have a default value of "M".  
5. The gender field should not allow insertion or updates directly by the user.  
6. The gender field should be linked to the database column named "sex".  
7. The gender field should display a list of predefined gender options for selection.  
8. The predefined gender options should include:  
 - "IP"  
 - "5"  
 - "8"  
 - "10"  
 - "LA"  
 - "No"  
 - "Other"  
  
Definition of Done:  
1. The gender field is implemented and displays the prompt "Gender".  
2. The gender field accepts input up to 20 characters and converts it to uppercase.  
3. The gender field has a default value of "M".  
4. The gender field does not allow direct insertion or updates by the user.  
5. The gender field is correctly linked to the "sex" column in the database.  
6. The predefined gender options are available for selection in the gender field.  
7. The gender field is tested and verified to meet all acceptance criteria.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- No specific DB queries provided in the XML content.

# View and Select Renewal Payment Options

Type: SUMMARY

Detailed description: As a user, I want to view and select renewal payment options from a predefined list on the summary page, so that I can easily choose the appropriate renewal payment option for a proposal.  
  
Acceptance criteria:  
- The renewal payment field should display a list of predefined options.  
- The list should include the following options:  
 - "5"  
 - "8"  
 - "No"  
- The renewal payment field should be read-only and not allow any modifications or insertions by the user.  
- The renewal payment field should be visually aligned and properly formatted on the summary page.  
  
Definition of Done:  
- The renewal payment field is implemented and displays the predefined list of options.  
- The field is read-only and does not allow user modifications.  
- The field is visually aligned and formatted correctly on the summary page.  
- The functionality 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.

# Manage Previous Policy Details

Type: AZBJ\_PREVIOUS\_POLICY1

Title: Manage Previous Policy Details  
  
Acceptance Criteria:  
1. The system should allow the user to input and update the following details for previous policies:  
 - Policy/Proposal Numbers  
 - Insurance Company  
 - Amount of Insurance  
 - Status  
 - IP Type  
  
2. The input fields for the above details should be available on the "Nominee Health Details" tab.  
  
3. The system should ensure that the "Amount of Insurance" field accepts only numerical values and has a maximum length of 15 characters.  
  
4. The "IP Type" field should be a list item with predefined values such as "IP", "5", "8", "10", "NO", "Single", and "No".  
  
Definition of Done:  
- The user can successfully input and update the previous policy details.  
- The input fields are correctly displayed on the "Nominee Health Details" tab.  
- Validation is in place to ensure the "Amount of Insurance" field accepts only numerical values and adheres to the maximum length.  
- The "IP Type" field displays the correct list of predefined values.  
- All changes are saved and reflected accurately 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 table references.

# Navigate through predefined values in form field

Type: AZBJ\_PREVIOUS\_POLICY1

Title: Navigate through predefined values in form field  
  
Acceptance Criteria:  
1. The form field should display a list of predefined values: 5, 8, 10, N, and SL.  
2. The field should not allow new entries or updates to the existing values.  
3. When the user navigates to the next item, the system should automatically move to the next field based on a predefined sequence.  
  
Definition of Done:  
1. The form field displays the predefined list of values.  
2. The field is read-only, preventing any new entries or updates.  
3. Navigation to the next item works as per the predefined sequence 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.

# Manage Previous Policy Details

Type: AZBJ\_PREVIOUS\_POLICY1

Detailed description: As a user, I want to manage the details of previous policies, including the type of insurance policy (IP\_TYPE), so that I can accurately track and update policy information.  
  
Acceptance criteria:  
1. When navigating to the next item, if the previous policy number is not null, the system should:  
 - Move to the last record and then to the next record.  
 - Copy the values from the current policy details to the new record.  
 - Clear the current policy details fields.  
 - Move the focus to the next item.  
  
2. When navigating to the previous item, the system should load the field from the image based on the current item.  
  
Definition of Done:  
- The user can navigate through policy records and update the policy type and other details.  
- The system correctly handles the transition between records, copying and clearing fields as specified.  
- The system loads the appropriate field from the image when navigating to the previous item.  
- All functionalities are tested and verified to work as expected.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- No specific CRUD operations mentioned in the provided XML content.

# Auto-load Insurance Company Details from Image

Type: AZBJ\_PREVIOUS\_POLICY1

Title: Auto-load Insurance Company Details from Image  
  
Acceptance Criteria:  
1. When the user navigates to the "Insurance Company" field, the system should automatically load the relevant details from an image file.  
2. The system should determine the current block and item, and set the visual attributes accordingly.  
3. The system should identify the correct image file based on the proposal type and other parameters.  
4. If the image file is found, it should be displayed in the designated area.  
5. If the image file is not found, an appropriate error message should be logged.  
  
Definition of Done:  
- The functionality to load insurance company details from an image is implemented and tested.  
- The system correctly identifies and displays the image based on the current context.  
- All acceptance criteria are met, and the feature is free of Oracle Forms-specific terminology.  
- The feature is reviewed and approved by the stakeholders.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- The following query is used to fetch the maximum strip number for the current block and item:  
 ```sql  
 SELECT MAX(strip\_no)  
 INTO v\_strip\_no  
 FROM azbj\_strip\_field\_mapping a  
 WHERE block\_name = v\_current\_block  
 AND field\_name = v\_current\_item  
 AND proposal\_type = PK\_VARS.v\_proposal\_type  
 AND EXISTS  
 (SELECT 1  
 FROM azbj\_proposal\_stripes b  
 WHERE a.proposal\_type = b.proposal\_type  
 AND INSTR(stripe\_string, '~' || a.strip\_no || '~') > 0);  
 ```

# Load Field Value from Image for Status Field

Type: AZBJ\_PREVIOUS\_POLICY1

Title: Load Field Value from Image for Status Field  
  
Acceptance Criteria:  
1. When the user navigates to the "Status" field, the system should trigger a procedure to load the field value from an image.  
2. The procedure should determine the current block and item, and set visual attributes based on the block type.  
3. The procedure should identify the correct image file based on the proposal type and other conditions, and load the image into the appropriate field.  
4. If the image loading fails, the system should handle the error gracefully and log the error details.  
  
Definition of Done:  
- The system correctly loads the field value from the image when the user navigates to the "Status" field.  
- The visual attributes are set correctly based on the block type.  
- The correct image file is identified and loaded without errors.  
- Error handling and logging are implemented for any failures during the image loading process.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- The procedure includes a query to select the maximum strip number from the `azbj\_strip\_field\_mapping` table based on the block name, field name, and proposal type, and checks for the existence of the strip number in the `azbj\_proposal\_stripes` table.  
  
```sql  
SELECT MAX(strip\_no)  
INTO v\_strip\_no  
FROM azbj\_strip\_field\_mapping a  
WHERE block\_name = v\_current\_block  
 AND field\_name = v\_current\_item  
 AND proposal\_type = PK\_VARS.v\_proposal\_type  
 AND EXISTS (  
 SELECT 1  
 FROM azbj\_proposal\_stripes b  
 WHERE a.proposal\_type = b.proposal\_type  
 AND INSTR(stripe\_string, '~' || a.strip\_no || '~') > 0  
 );  
```  
  
This query is used to determine the appropriate strip number for loading the image based on the current context.

# Load and Display Correct Insurance Amount

Type: AZBJ\_PREVIOUS\_POLICY1

Title: Load and Display Correct Insurance Amount  
  
Acceptance Criteria:  
1. The system should load the field value from an image based on the current item.  
2. The system should update the visual attributes of the current item based on the block context.  
3. The system should navigate to the next item when the user presses the key to move to the next item.  
4. The system should determine the correct visual attribute to apply based on the block name.  
5. The system should handle different block contexts and set the appropriate visual attributes.  
6. The system should retrieve the correct strip number and file name for the image based on the proposal type and other conditions.  
7. The system should read and display the image file in the appropriate format and location.  
8. The system should handle exceptions and log errors appropriately.  
  
Definition of Done:  
1. The functionality to load and display the correct insurance amount is implemented and tested.  
2. The visual attributes are correctly updated based on the block context.  
3. The navigation to the next item works as expected.  
4. The system correctly handles different block contexts and sets the appropriate visual attributes.  
5. The correct strip number and file name are retrieved and used to display the image.  
6. The image is read and displayed correctly in the specified format and location.  
7. All exceptions are handled, and errors are logged appropriately.  
8. The feature is reviewed and approved by stakeholders.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- The following query is used to retrieve the maximum strip number for the current block and item:  
 ```sql  
 SELECT MAX(strip\_no)  
 INTO v\_strip\_no  
 FROM azbj\_strip\_field\_mapping a  
 WHERE block\_name = v\_current\_block  
 AND field\_name = v\_current\_item  
 AND proposal\_type = PK\_VARS.v\_proposal\_type  
 AND EXISTS  
 (SELECT 1  
 FROM azbj\_proposal\_stripes b  
 WHERE a.proposal\_type = b.proposal\_type  
 AND INSTR(stripe\_string, '~' || a.strip\_no || '~') > 0);  
 ```

# Ensure Correct Population and Navigation of Policy/Proposal Numbers Field

Type: AZBJ\_PREVIOUS\_POLICY1

Title: Ensure Correct Population and Navigation of Policy/Proposal Numbers Field  
  
Acceptance Criteria:  
1. The "Policy/Proposal Numbers" field should load data from an image when the field is accessed.  
2. When navigating to the next item, the system should clear any visual attributes from the current item and move to the "Correct Previous Insurance Company" field.  
3. The system should dynamically set visual attributes based on the current block and item.  
4. The system should determine the appropriate block and item names based on the current context.  
5. The system should fetch the maximum strip number from the `azbj\_strip\_field\_mapping` table based on the current block, item, and proposal type.  
6. If the current item is related to family details and the proposal type is 'N', the strip number should be set to 34 or 35 based on specific conditions.  
7. The system should read and display the image file corresponding to the strip number if it has changed from the previous value.  
  
Definition of Done:  
1. The "Policy/Proposal Numbers" field loads data from an image correctly.  
2. Navigation to the next item works as expected, moving to the "Correct Previous Insurance Company" field.  
3. Visual attributes are dynamically set based on the current block and item.  
4. The system correctly determines and sets the appropriate block and item names.  
5. The maximum strip number is fetched correctly from the database.  
6. The strip number is set to 34 or 35 for family details based on the proposal type and specific conditions.  
7. The image file is read and displayed correctly if the strip number has changed.  
  
DB queries for Table reference CRUD operations only (With Usage):  
```sql  
-- Fetch the maximum strip number based on the current block, item, and proposal type  
SELECT MAX(strip\_no)  
INTO v\_strip\_no  
FROM azbj\_strip\_field\_mapping a  
WHERE block\_name = v\_current\_block  
 AND field\_name = v\_current\_item  
 AND proposal\_type = PK\_VARS.v\_proposal\_type  
 AND EXISTS (  
 SELECT 1  
 FROM azbj\_proposal\_stripes b  
 WHERE a.proposal\_type = b.proposal\_type  
 AND INSTR(stripe\_string, '~' || a.strip\_no || '~') > 0  
 );  
```

# Manage Previous Insurance Policy Details

Type: AZBJ\_PREVIOUS\_POLICY

Title: Manage Previous Insurance Policy Details  
  
Acceptance Criteria:  
1. The system should allow the user to input and update the following details for previous insurance policies:  
 - Policy/Proposal Numbers  
 - Insurance Company  
 - Amount of Insurance  
 - Status  
 - IP Type  
2. The system should display these details in a structured format, allowing for easy viewing and editing.  
3. The system should provide a delete button to remove any previous policy details if needed.  
4. The system should ensure that the input fields for policy details are validated for appropriate data types and lengths.  
5. The system should handle mouse click events to navigate to the appropriate input field when clicked.  
  
Definition of Done:  
- The user can successfully add, update, and delete previous insurance policy details.  
- The input fields are validated for correct data types and lengths.  
- The user interface displays the policy details in a clear and organized manner.  
- The delete functionality works as expected, removing the selected policy details.  
- Mouse click events are handled correctly, allowing for smooth navigation between input fields.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include specific database queries.

# Delete Previous Policy Record

Type: AZBJ\_PREVIOUS\_POLICY

Title: Delete Previous Policy Record  
  
Acceptance Criteria:  
- When the "Delete" button is pressed, the system should delete the selected previous policy record from the database.  
- The button should be labeled "Delete" and be located at the specified position on the user interface.  
- The button should have a gray background, black text, and be styled with a bold font.  
  
Definition of Done:  
- The "Delete" button is implemented and visible on the user interface.  
- Pressing the "Delete" button successfully removes the selected previous policy record from the database.  
- The button's appearance matches the specified design requirements (gray background, black text, bold font).  
  
DB queries for Table reference CRUD operations only (With Usage):  
- No specific DB queries provided in the XML content.

# Input Amount of Insurance

Type: AZBJ\_PREVIOUS\_POLICY

Title: Input Amount of Insurance  
  
Acceptance Criteria:  
1. The text field should allow a maximum of 15 characters.  
2. The text field should accept only numerical input.  
3. The text field should be editable and allow both insertion and updates.  
4. When the text field is focused, it should load the field from an image based on the current item.  
5. Upon pressing the "Next" key, the focus should move to the next item, specifically the "IP\_TYPE" field in the same block.  
  
Definition of Done:  
1. The text field for the amount of insurance is implemented and visible on the form.  
2. The text field adheres to the specified constraints (maximum length, numerical input).  
3. The text field is editable and allows both insertion and updates.  
4. The logic to load the field from an image when the text field is focused is implemented.  
5. The navigation logic to move to the "IP\_TYPE" field upon pressing the "Next" key is implemented and functional.  
  
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.

# User selects IP Type from predefined list

Type: AZBJ\_PREVIOUS\_POLICY

Title: User selects IP Type from predefined list  
  
Acceptance Criteria:  
1. The system should display a list of predefined IP Types for selection.  
2. The list should include the following values: "IP", "5", "8", "10", "N", and "SL".  
3. When the user navigates to the IP Type field, the system should load the field from an image based on the current item.  
4. Upon pressing the key to move to the next item, the system should navigate to the "Previous Policy Status" field.  
  
Definition of Done:  
1. The IP Type field is displayed with a dropdown list containing the predefined values.  
2. The field can be navigated to and from using keyboard actions.  
3. The system correctly loads the field from an image when the user navigates to it.  
4. The system navigates to the "Previous Policy Status" field when the user presses the key to move to the next item.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- The following query is used to retrieve the maximum strip number for the current block and item:  
 ```sql  
 SELECT MAX(strip\_no)  
 INTO v\_strip\_no  
 FROM azbj\_strip\_field\_mapping a  
 WHERE block\_name = v\_current\_block  
 AND field\_name = v\_current\_item  
 AND proposal\_type = PK\_VARS.v\_proposal\_type  
 AND EXISTS  
 (SELECT 1  
 FROM azbj\_proposal\_stripes b  
 WHERE a.proposal\_type = b.proposal\_type  
 AND INSTR(stripe\_string, '~' || a.strip\_no || '~') > 0);  
 ```

# Input and Update Previous Policy Number

Type: AZBJ\_PREVIOUS\_POLICY

Title: Input and Update Previous Policy Number  
  
Acceptance Criteria:  
1. The input field for the previous policy number should be visible and editable.  
2. When the user navigates to the previous policy number field, the system should load the field from an image if applicable.  
3. Upon pressing the key to move to the next item, the system should clear any visual attributes from the current item and navigate to the next item in the sequence.  
4. The system should handle different blocks and items dynamically, setting visual attributes based on the current block and item.  
5. The system should determine the appropriate image file to load based on the current block, item, and proposal type, and display the image accordingly.  
6. The system should handle exceptions gracefully and log any errors encountered during the process.  
  
Definition of Done:  
- The previous policy number field is implemented and functional.  
- The field can be edited and updated by the user.  
- The system dynamically loads and displays images based on the current context.  
- All acceptance criteria are met, and the system handles exceptions and logs errors appropriately.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- The following query is used to retrieve the maximum strip number for the current block and item:  
 ```sql  
 SELECT MAX(strip\_no)  
 INTO v\_strip\_no  
 FROM azbj\_strip\_field\_mapping a  
 WHERE block\_name = v\_current\_block  
 AND field\_name = v\_current\_item  
 AND proposal\_type = PK\_VARS.v\_proposal\_type  
 AND EXISTS  
 (SELECT 1  
 FROM azbj\_proposal\_stripes b  
 WHERE a.proposal\_type = b.proposal\_type  
 AND INSTR(stripe\_string, '~' || a.strip\_no || '~') > 0);  
 ```

# Input and Update Previous Insurance Company Information

Type: AZBJ\_PREVIOUS\_POLICY

Title: Input and Update Previous Insurance Company Information  
  
Acceptance Criteria:  
1. The field should allow a maximum of 50 characters.  
2. The field should be editable and allow both insert and update operations.  
3. When the field is accessed, it should load relevant data from an image if available.  
4. Upon moving to the next field, the system should clear any visual attributes applied to the current field and navigate to the next item in the sequence.  
5. The system should dynamically determine the visual attributes and the current block/item based on predefined conditions.  
6. The system should fetch the appropriate image file based on the current block/item and display it if available.  
  
Definition of Done:  
1. The user can successfully input and update the previous insurance company information.  
2. The system correctly loads data from images when the field is accessed.  
3. Navigation to the next item works seamlessly, clearing any visual attributes from the current field.  
4. The system dynamically applies visual attributes and determines the current block/item as per the predefined conditions.  
5. The appropriate image file is fetched and displayed correctly based on the current block/item.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- The system should execute a query to fetch the maximum strip number from the `azbj\_strip\_field\_mapping` table based on the current block/item and proposal type.  
- The system should check for the existence of a record in the `azbj\_proposal\_stripes` table to validate the strip number.  
  
```sql  
SELECT MAX(strip\_no)  
INTO v\_strip\_no  
FROM azbj\_strip\_field\_mapping a  
WHERE block\_name = v\_current\_block  
 AND field\_name = v\_current\_item  
 AND proposal\_type = PK\_VARS.v\_proposal\_type  
 AND EXISTS (  
 SELECT 1  
 FROM azbj\_proposal\_stripes b  
 WHERE a.proposal\_type = b.proposal\_type  
 AND INSTR(stripe\_string, '~' || a.strip\_no || '~') > 0  
 );  
```

# Validate Age and Load Image Based on Context

Type: AZBJ\_PREVIOUS\_POLICY

Title: Validate Age and Load Image Based on Context  
  
Acceptance Criteria:  
1. The system should validate that the age entered in the family history section is not less than zero.  
2. The system should load the appropriate image based on the current item and block context.  
3. The system should set visual attributes for the current item based on the block context.  
4. The system should determine the current block and item based on specific conditions.  
5. The system should retrieve the maximum strip number from the `azbj\_strip\_field\_mapping` table based on the block name, field name, and proposal type.  
6. The system should handle exceptions and set the strip number to null if an error occurs.  
7. The system should adjust the strip number based on specific conditions related to family details and proposal type.  
8. The system should read and display the image file based on the strip number and proposal type.  
  
Definition of Done:  
- The age validation logic is implemented and tested.  
- The image loading logic is implemented and tested.  
- Visual attributes are correctly set based on the block context.  
- The current block and item determination logic is implemented and tested.  
- The maximum strip number retrieval logic is implemented and tested.  
- Exception handling is implemented and tested.  
- The strip number adjustment logic is implemented and tested.  
- The image reading and displaying logic is implemented and tested.  
  
DB queries for Table reference CRUD operations only (With Usage):  
```sql  
SELECT MAX(strip\_no)  
INTO v\_strip\_no  
FROM azbj\_strip\_field\_mapping a  
WHERE block\_name = v\_current\_block  
AND field\_name = v\_current\_item  
AND proposal\_type = PK\_VARS.v\_proposal\_type  
AND EXISTS (  
 SELECT 1  
 FROM azbj\_proposal\_stripes b  
 WHERE a.proposal\_type = b.proposal\_type  
 AND INSTR(stripe\_string, '~' || a.strip\_no || '~') > 0  
);  
```

# Update Previous Policy Status and Navigate Through Form

Type: AZBJ\_PREVIOUS\_POLICY

Title: Update Previous Policy Status and Navigate Through Form  
  
Detailed description: As a user, I want to be able to update the status of a previous policy and navigate through the form to add more policies if needed. The system should prompt me with a question asking if I wish to add more policies. Based on my response, the system should either move to the next record and update the relevant fields or perform specific actions based on the type and number of pages of the proposal.  
  
Acceptance criteria:  
1. When the user navigates to the "Previous Policy Status" field, the system should load the field from an image.  
2. Upon pressing the key to move to the next item, the system should prompt the user with a question: "Do you wish to add more Policies?".  
3. If the user selects "Yes":  
 - The system should move to the next record.  
 - The system should navigate to the "Previous Policy Number" field.  
 - The system should update the next item bean to "Previous Policy Number".  
4. If the user selects "No":  
 - The system should check the proposal type and number of pages.  
 - Based on the proposal type and number of pages, the system should set parameters and load the appropriate image file.  
 - The system should update the loaded page count.  
 - The system should navigate to the "Nominee Next Tab" field.  
 - The system should update the next item bean to "Nominee Next Tab".  
  
Definition of Done:  
- The user is able to update the status of a previous policy.  
- The system prompts the user with a question when navigating to the next item.  
- The system performs the correct actions based on the user's response and the proposal type.  
- The system navigates to the appropriate fields and updates the next item bean accordingly.  
- The system loads the correct image file based on the proposal type and number of pages.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- No specific DB queries provided in the XML content for CRUD operations.

# View and Interact with Receipt Details

Type: SUSAC

Title: View and Interact with Receipt Details  
  
Acceptance Criteria:  
1. The user should be able to view the following details for each receipt:  
 - Receipt Number  
 - Amount  
 - Date Received  
 - Cheque Status  
 - Premium Payer  
 - Branch Code  
 - Branch Name  
 - Branch State  
2. The total amount of all receipts should be automatically calculated and displayed.  
3. The user should be able to navigate through the records using a scrollbar.  
4. The user should be able to click on a record to select it and view its details.  
  
Definition of Done:  
1. The user interface displays all the required receipt details.  
2. The total amount is correctly calculated and displayed.  
3. The scrollbar allows the user to navigate through the records.  
4. Clicking on a record highlights it and displays its details.  
5. 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 operations or queries.

# Manage PAN Approval Details

Type: BLK\_PAN\_APPROVAL

Detailed description: As a user, I want to manage PAN approval details, including issuance dates, age proof, and supervisor comments, so that I can ensure accurate and complete information is recorded for each PAN approval request.  
  
Acceptance criteria:  
1. The system should display fields for entering the following details:  
 - IP PAN Issuance Date  
 - PH PAN Issuance Date  
 - Age Proof  
 - Age Proof ID  
 - PAN Issuance Date  
 - UW Reason  
 - Supervisor Comments  
2. The system should provide checkboxes to indicate if the IP PAN Date or PH PAN Date is unavailable.  
3. The system should include buttons for "Supervisor Approval" and "EXIT".  
4. The date fields should follow the format "dd-mm-yyyy".  
5. The checkboxes should have default values of "N" (unchecked) and should change to "Y" (checked) when selected.  
6. The "Supervisor Approval" button should trigger the approval process.  
7. The "EXIT" button should close the form or navigate the user away from the current screen.  
  
Definition of Done:  
- The user can view and interact with all the specified fields and buttons.  
- The date fields accept and display dates in the "dd-mm-yyyy" format.  
- The checkboxes correctly reflect the availability status of the PAN dates.  
- The "Supervisor Approval" button successfully initiates the approval process.  
- The "EXIT" button functions as expected, closing the form or navigating away.  
  
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 PAN Issuance Date Field Based on Checkbox Status

Type: BLK\_PAN\_APPROVAL

Detailed description: As a user, I want the system to manage the availability of the PAN issuance date field based on the status of a checkbox, so that the field is enabled or disabled appropriately.  
  
Acceptance criteria:  
1. When the checkbox labeled "PH PAN Date UnAvailable" is checked, the PAN issuance date field should be cleared and disabled.  
2. When the checkbox is unchecked, the PAN issuance date field should be enabled.  
  
Definition of Done:  
- The checkbox labeled "PH PAN Date UnAvailable" should control the state of the PAN issuance date field.  
- The PAN issuance date field should be cleared and disabled when the checkbox is checked.  
- The PAN issuance date field should be enabled when the checkbox is unchecked.  
- The changes should be reflected immediately upon checking or unchecking the checkbox.  
  
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 PAN Issuance Date

Type: BLK\_PAN\_APPROVAL

Title: Validate PAN Issuance Date  
  
Acceptance Criteria:  
1. The system should validate the PAN Issuance Date upon entry.  
2. If the entered PAN Issuance Date is greater than the current system date, an error message "PAN Issuance Date should not be less than system date" should be displayed.  
3. The user should not be able to proceed to the next field until a valid PAN Issuance Date is entered.  
  
Definition of Done:  
1. The validation logic for the PAN Issuance Date is implemented and tested.  
2. The error message is displayed correctly when the PAN Issuance Date is a future date.  
3. The user is prevented from moving to the next field until a valid date is entered.  
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 CRUD operations.

# Manage PAN Issuance Date Availability

Type: BLK\_PAN\_APPROVAL

Detailed description: As a user, I want to manage the availability status of the PAN issuance date so that I can accurately reflect whether the PAN issuance date is available or not.  
  
Acceptance criteria:  
1. When the checkbox for "IP PAN Date Unavailable" is checked:  
 - The PAN issuance date field should be cleared.  
 - The PAN issuance date field should be disabled.  
2. When the checkbox for "IP PAN Date Unavailable" is unchecked:  
 - The PAN issuance date field should be enabled.  
  
Definition of Done:  
- The functionality to manage the availability status of the PAN issuance date is implemented.  
- The PAN issuance date field is correctly enabled or disabled based on the checkbox status.  
- The PAN issuance date field is cleared when the checkbox is checked.  
- The 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 operations.

# Validate PAN Issuance Date

Type: BLK\_PAN\_APPROVAL

Title: Validate PAN Issuance Date  
  
Acceptance Criteria:  
1. The system should validate the PAN Issuance Date upon entry.  
2. If the entered PAN Issuance Date is greater than the current system date, an error message "PAN Issuance Date should not be less than system date" should be displayed.  
3. The user should be able to navigate to the next item in the form after entering a valid PAN Issuance Date.  
  
Definition of Done:  
1. The PAN Issuance Date field is validated to ensure it is not a future date.  
2. An appropriate error message is displayed if the validation fails.  
3. The user can successfully navigate to the next item in the form after entering a valid date.  
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.

# Exit Approval Screen and Navigate to Validation Screen

Type: BLK\_PAN\_APPROVAL

Detailed description: As a user, I want to be able to exit the current approval screen and navigate to the validation screen, ensuring that certain variables are set correctly to maintain the integrity of the approval process.  
  
Acceptance criteria:  
1. When the exit button is pressed, the current approval window should be hidden.  
2. The approval canvas should also be hidden.  
3. The variable `pan\_validate\_calling` should be set to 'Y'.  
4. The variable `pan\_validate\_continue` should be set to 'N'.  
5. The system should navigate to the validation screen.  
6. The trigger for the validation screen should be executed.  
  
Definition of Done:  
- The exit button hides the current approval window and canvas.  
- The specified variables are set correctly.  
- The system navigates to the validation screen and executes the necessary trigger.  
  
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 Button Functionality

Type: BLK\_PAN\_APPROVAL

Title: Supervisor Approval Button Functionality  
  
Acceptance Criteria:  
1. When the "Supervisor Approval" button is pressed, the system should check if the "UW Reason" and "Supervisor Comment" fields are not empty.  
2. The system should validate that the "UW Reason" and "Supervisor Comment" fields do not contain any special characters.  
3. If the validation fails, an error message should be displayed indicating the specific issue (e.g., "Special characters are not allowed in Reason!" or "Special characters are not allowed in Comments!").  
4. If the validation passes, the system should call a specific form and update the approval status based on the global approval variable.  
5. If the approval status is not "Y" and the approval count is 1, the system should navigate to the "result" block and update the "RESULT" fields accordingly.  
6. The system should handle record navigation and updates within the "result" block based on the cursor position and the "res\_type" field.  
  
Definition of Done:  
- The "Supervisor Approval" button functionality is implemented and tested.  
- The input fields are validated for non-emptiness and absence of special characters.  
- Appropriate error messages are displayed for validation failures.  
- The approval logic is executed correctly, and the "result" block is updated as per the specified conditions.  
- All acceptance criteria are met, and the feature is reviewed and approved by stakeholders.

# Manage Rider Details in Proposal Form

Type: AZBJ\_RIDER\_DETAILS

Detailed description: As a user, I want to manage rider details within the proposal form, including viewing, adding, and deleting rider information such as cover code, cover description, sum assured, benefit term, premium term, interest rate, and entry age.  
  
Acceptance criteria:  
1. The user should be able to view rider details including cover code, cover description, sum assured, benefit term, premium term, interest rate, and entry age.  
2. The user should be able to add new rider details with the following fields:  
 - Cover Code  
 - Cover Description  
 - Sum Assured  
 - Benefit Term  
 - Premium Term  
 - Interest Rate  
 - Entry Age  
3. The user should be able to delete rider details by selecting the delete checkbox and clicking the delete button.  
4. The populate button should allow the user to auto-fill certain fields based on predefined logic.  
5. The fields for cover code and cover description should be read-only.  
6. The fields for sum assured, benefit term, premium term, and interest rate should be editable.  
7. The entry age field should be read-only.  
8. The visibility and navigability of the interest rate, premium term, and benefit term fields should be controlled based on specific conditions (e.g., product definition).  
  
Definition of Done:  
1. The user interface for managing rider details is implemented and accessible within the proposal form.  
2. The user can view, add, and delete rider details as per the acceptance criteria.  
3. The populate button functionality is implemented and working as expected.  
4. The visibility and navigability of specific fields are controlled based on predefined conditions.  
5. All fields are displayed with appropriate labels and are aligned correctly.  
6. The implementation is tested and verified to ensure it meets the acceptance criteria.  
7. The feature is reviewed and approved by the stakeholders.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Not applicable as the provided XML content does not include specific database queries.

# Interest Rate Input for Loan

Type: AZBJ\_RIDER\_DETAILS

Title: Interest Rate Input for Loan  
  
Acceptance Criteria:  
1. The interest rate field should accept numeric values only.  
2. The interest rate field should be displayed with a white background and positioned correctly on the form.  
3. The interest rate field should be labeled as "LoanInt. Rt" and the label should be bold and centered above the field.  
4. The interest rate field should allow for insertion and updates.  
5. Upon entering the interest rate, the system should navigate to the next item in the sequence, which is the "CV\_CHK\_DEL" field.  
  
Definition of Done:  
1. The interest rate field is implemented and displayed correctly on the form.  
2. The field accepts numeric values and restricts non-numeric input.  
3. The label "LoanInt. Rt" is displayed correctly above the field.  
4. The field allows users to insert and update values.  
5. The navigation to the "CV\_CHK\_DEL" field works as expected after entering the interest rate.  
  
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 and Navigate Rider Cover Details

Type: AZBJ\_RIDER\_DETAILS

Title: Display and Navigate Rider Cover Details  
  
Acceptance Criteria:  
1. The rider cover details should be displayed in a text item field.  
2. The text item field should have a maximum length of 200 characters and should be displayed in uppercase.  
3. The field should be read-only, meaning users cannot insert or update the information directly.  
4. The field should be positioned appropriately within the form and should have a white background.  
5. The field should have a prompt labeled "Cover Description" that is centered and styled with a bold font.  
6. Navigation within the form should allow users to move to the next item, which is the rider cover sum assured, upon pressing the next item key.  
  
Definition of Done:  
- The rider cover details are displayed correctly in the form.  
- The field adheres to the specified design and read-only constraints.  
- The navigation logic to the next item is implemented and functional.  
- The form is tested to ensure that the rider cover details are displayed and navigated 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.

# Navigation and Display for Rider Cover Code

Type: AZBJ\_RIDER\_DETAILS

Title: Navigation and Display for Rider Cover Code  
  
Acceptance Criteria:  
1. When the user is on the "RIDER\_COVER\_CODE" field and presses the key to move to the next item, the system should automatically navigate to the "RIDER\_COVER1" field.  
2. The "RIDER\_COVER\_CODE" field should be displayed with a white background and should be positioned correctly within the form.  
3. The "RIDER\_COVER\_CODE" field should be in uppercase and should not allow insertions or updates.  
4. The "RIDER\_COVER\_CODE" field should have a prompt labeled "Cover Code" that is centered and bold.  
5. The "RIDER\_COVER\_CODE" field should have predefined list items with values 5, 8, 10, and "No".  
  
Definition of Done:  
- The navigation from "RIDER\_COVER\_CODE" to "RIDER\_COVER1" works as specified.  
- The "RIDER\_COVER\_CODE" field is displayed with the correct properties and restrictions.  
- The prompt for the "RIDER\_COVER\_CODE" field is correctly labeled and formatted.  
- The predefined list items are available for selection in the "RIDER\_COVER\_CODE" field.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include specific database queries.

# Manage Rider Details with Deletion Checkbox

Type: AZBJ\_RIDER\_DETAILS

Detailed description: As a user, I want to manage the rider details in the proposal form, including the ability to mark a rider for deletion using a checkbox, so that I can efficiently handle the rider information.  
  
Acceptance criteria:  
1. When the checkbox labeled "Delete" is checked, the value should be set to 'Y'.  
2. When the checkbox labeled "Delete" is unchecked, the value should be set to 'N'.  
3. If the number of riders matches the current cursor record, the focus should move to the delete item.  
4. If the number of riders does not match the current cursor record, the focus should move to the next rider cover item.  
5. If the rider cover code is null, the focus should move to the delete item.  
  
Definition of Done:  
1. The checkbox for marking a rider for deletion is functional and correctly updates its value based on user interaction.  
2. Navigation logic is implemented to move the focus based on the number of riders and the presence of a rider cover code.  
3. The form is tested to ensure that the checkbox and navigation logic work as expected.  
4. The form is free of any Oracle Forms-specific terminology and is independent of the underlying technology.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- No specific DB queries provided in the XML content.

# Input and Navigation for RIDER\_COVER1\_SA Field

Type: AZBJ\_RIDER\_DETAILS

Title: Input and Navigation for RIDER\_COVER1\_SA Field  
  
Acceptance Criteria:  
1. When the "RIDER\_COVER1\_SA" field is accessed, it should load data from an image file based on the current context and item.  
2. Navigation to the previous item should move to the "CORRECT" field within the "Rider Details" section.  
3. Navigation to the next item should:  
 - Move to the "BENEFIT\_TERM" field if the product definition is one of the specified types (e.g., 'ISECURE\_LOAN', 'ISECURE\_MORE', etc.).  
 - Otherwise, move to the "CV\_CHK\_DEL" field.  
4. The visual attributes of the current item should be updated based on the context (e.g., different visual attributes for different blocks).  
  
Definition of Done:  
- The "RIDER\_COVER1\_SA" field can be accessed and displays data loaded from an image.  
- Navigation between items works as specified, with appropriate visual feedback.  
- The field supports both insertion and updates.  
- The field's data type is validated as a number.  
- The field's visual attributes change based on the current block context.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- The following query is used to determine the maximum strip number for loading the field from an image:  
 ```sql  
 SELECT MAX(strip\_no)  
 INTO v\_strip\_no  
 FROM azbj\_strip\_field\_mapping a  
 WHERE block\_name = v\_current\_block  
 AND field\_name = v\_current\_item  
 AND proposal\_type = PK\_VARS.v\_proposal\_type  
 AND EXISTS  
 (SELECT 1  
 FROM azbj\_proposal\_stripes b  
 WHERE a.proposal\_type = b.proposal\_type  
 AND INSTR(stripe\_string, '~' || a.strip\_no || '~') > 0);  
 ```

# Populate Rider Details Based on Selected Package and Product Code

Type: AZBJ\_RIDER\_DETAILS

Detailed description: As a user, I want to populate rider details based on the selected package and product code, so that the correct rider information is displayed and updated accordingly.  
  
Acceptance criteria:  
1. When the "POPULATE" button is pressed, the system should:  
 - Check various conditions based on the insured status, age, and product type to set flags (`v\_ip\_ph\_flag`, `v\_minor\_flag`, `v\_joint\_flag`).  
 - Log the cover details issue using the `bbu\_ins\_log` utility.  
 - If the group product flag is 'Y', clear the `AZBJ\_RIDER\_DETAILS` block and populate it with rider details from the database based on the selected package and product code.  
 - For each rider cover, set the cover code, description, sum assured, premium term, benefit term, entry age, and rider partner based on specific conditions.  
 - Delete records if certain conditions are met (e.g., top-ups are zero, specific cover codes for certain product codes).  
 - Adjust premium and benefit terms based on product and cover codes.  
 - Refresh policy member details if the product code is 315.  
 - Refresh child covers if the product code is 321 and the package code starts with 'SMART\_CHILD\_'.  
  
Definition of Done:  
- The "POPULATE" button functionality is implemented and tested.  
- The rider details are correctly populated based on the selected package and product code.  
- All conditions and validations are handled as per the acceptance criteria.  
- The system logs appropriate messages and handles exceptions gracefully.  
- The feature is reviewed and approved by stakeholders.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- The following query is used to fetch rider details based on the selected package and product code:  
 ```sql  
 SELECT a.package\_code, a.package\_desc, 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.package\_code = :correct\_value.package\_code  
 AND a.product\_id = :correct\_value.product\_code  
 AND a.product\_id = c.product\_id  
 AND b.cover\_code = c.cover\_code  
 ORDER BY b.cover\_code;  
 ```

# Delete Rider Details with Validations

Type: AZBJ\_RIDER\_DETAILS

Detailed description: As a user, I want to be able to delete rider details from the rider details section, ensuring that all necessary validations and conditions are checked before the deletion is performed.  
  
Acceptance criteria:  
1. When the delete button is pressed, the system should check if the product is a group product. If it is, a specific group deletion procedure should be called.  
2. If the product is not a group product, the system should:  
 - Navigate to the rider details section and iterate through all records.  
 - Check if the rider cover code starts with 'L'. If it does, set the entry age.  
 - If the rider cover code starts with 'R' and the product code is 4 and the delete check flag is 'Y', display an error message indicating that the selected riders are compulsory with the package selected.  
 - Set various flags based on the rider cover code.  
 - For specific product codes, ensure that certain rider cover codes are always deleted together.  
 - If the product code is 315, ensure that no policy members exist for the rider cover code before allowing deletion.  
 - Iterate through the records again to set the delete check flag based on the previously set flags and conditions.  
 - If the product code is not in a specific list and the rider cover code starts with 'R' and the delete check flag is 'Y', delete the record.  
 - If the rider cover code starts with 'T' and the delete check flag is 'Y' and the product code is less than 183, delete the record.  
 - If the rider cover code starts with 'T' and the delete check flag is 'Y' and the product code is 183 or greater, set the delete check flag to 'N'.  
3. After all deletions, if the product code is 315, update the rider partner field based on the rider cover code and other conditions.  
  
Definition of Done:  
- The delete button functionality is implemented and tested.  
- All specified conditions and validations are checked before performing deletions.  
- Appropriate error messages are displayed when necessary.  
- The rider partner field is updated correctly for product code 315.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- The following query is used to count deferred annuity products:  
 ```sql  
 SELECT COUNT (1)  
 INTO v\_deferred\_annuity\_cnt  
 FROM AZBJ\_SYSTEM\_CONSTANTS  
 WHERE SYS\_TYPE = 'DEFERRED\_ANNUITY'  
 AND SYS\_CODE = 'DEFERRED\_ANNUITY\_PRODUCT'  
 AND NUM\_VALUE = :correct\_value.product\_code;  
 ```

# View and Select Fund Details

Type: AZBJ\_FUND\_DETAILS2

Title: View and Select Fund Details  
  
Acceptance Criteria:  
1. The system should display a list of available funds for selection.  
2. The list of funds should be fetched based on the following SQL query:  
 ```sql  
 SELECT fund\_short\_name AS FUND\_ID, 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 = :correct\_value.product\_CODE  
 AND :susac.sa\_daterecd >= NVL(start\_date, '01-apr-2005')  
 AND :susac.sa\_daterecd <= NVL(end\_date, '01-jan-3000')  
 ```  
3. The fund details should include the fund ID and fund name.  
4. The user should be able to select a fund from the list.  
5. The selected fund details should be displayed in the appropriate fields.  
  
Definition of Done:  
- The list of funds is displayed correctly based on the query.  
- The user can select a fund from the list.  
- The selected fund details are displayed in the designated fields.  
- The functionality is tested and verified to work as expected.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- The provided SQL query is used to fetch the list of available funds based on specific conditions.

# Load and Display Image for FUNDNAME2 Field

Type: AZBJ\_FUND\_DETAILS2

Title: Load and Display Image for FUNDNAME2 Field  
  
Acceptance Criteria:  
1. When the user navigates to the "FUNDNAME2" field:  
 - The system should determine the current block and item.  
 - If the proposal type is 'N' and the number of pages is 50, the system should load the image for the proposal form.  
 - If the form version is '10.%', the system should zoom the image to fit the display.  
  
2. The system should handle different blocks and items appropriately:  
 - For blocks like 'AZBJ\_FUND\_DETAILS1', 'AZBJ\_SSO\_FUND\_DETAILS1', 'AZBJ\_RIDER\_DETAILS1', etc., the system should set the visual attribute to 'VA\_CORRECT\_TEXT'.  
 - For blocks like 'CORRECT\_VALUE' and 'CORRECT', the system should adjust the current block and item accordingly.  
  
3. The system should fetch the maximum strip number based on the block, item, and proposal type, and check if it exists in the proposal stripes.  
  
4. If the current item is related to family details and the proposal type is 'N' with 50 pages, the system should set the strip number based on the family member type.  
  
5. The system should read and display the image file based on the strip number and form version:  
 - For form version '10.%', the system should read the image file and zoom it to fit the display.  
 - For other versions, the system should adjust the image path and read the image using a different method.  
  
Definition of Done:  
- The system correctly loads and displays the image for the "FUNDNAME2" field based on the current item, block, and proposal type.  
- The system handles different blocks and items as specified.  
- The system fetches and uses the correct strip number for loading the image.  
- The image is displayed correctly based on the form version and zoom settings.  
- All acceptance criteria are met, and the functionality is tested and verified.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- No direct DB queries are provided in the XML content for CRUD operations. The logic involves fetching data and setting properties based on the current item and block, which is handled within the application logic.

# Manage Rider Details in Insurance Proposal

Type: AZBJ\_RIDER\_DETAILS2

User Story: Manage Rider Details in Insurance Proposal  
  
Detailed description:   
As an insurance agent, I want to manage rider details within an insurance proposal so that I can accurately capture and update information related to additional coverage options for the policyholder.  
  
Acceptance criteria:  
1. The system should allow the agent to view and update the following fields for each rider:  
 - Cover Code  
 - Cover Description  
 - Sum Assured (SA)  
 - Benefit Term (BT)  
 - Premium Term (PT)  
 - Interest Rate  
 - Entry Age  
 - Total Rider Sum Assured  
 - Cover Description (display only)  
2. The system should ensure that the "Cover Code" and "Cover Description" fields are displayed in uppercase.  
3. The "Sum Assured", "Benefit Term", "Premium Term", and "Interest Rate" fields should accept numeric values only.  
4. The "Entry Age" field should be displayed but not editable.  
5. The "Total Rider Sum Assured" and "Cover Description" fields should be displayed but not editable.  
6. The system should provide a checkbox to mark a rider for deletion.  
7. The system should include a button to navigate to the next tab.  
8. The system should handle mouse click events to navigate to the clicked item if it is not a button.  
  
Definition of Done:  
- The user interface for managing rider details is implemented and accessible within the insurance proposal module.  
- All specified fields are displayed and function as described in the acceptance criteria.  
- The system correctly handles uppercase conversion for specified fields.  
- Numeric fields validate input to ensure only numbers are accepted.  
- Non-editable fields are displayed correctly and cannot be modified by the user.  
- The delete checkbox and navigation button are functional.  
- Mouse click events are handled appropriately to navigate to the clicked item.  
- The feature 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 database queries or table references.

# Efficient Navigation for Rider Cover Code

Type: AZBJ\_RIDER\_DETAILS2

Title: Efficient Navigation for Rider Cover Code  
  
Acceptance Criteria:  
1. When the user navigates to the "RIDER\_COVER\_CODE" field, the system should automatically move to the next item, "RIDER\_COVER1", upon pressing the next item key.  
2. The system should store the next item bean value as 'AZBJ\_RIDER\_DETAILS.RIDER\_COVER1' for further processing.  
  
Definition of Done:  
- The navigation from "RIDER\_COVER\_CODE" to "RIDER\_COVER1" should be seamless and without errors.  
- The next item bean value should be correctly set and retrievable for subsequent operations.  
- The "RIDER\_COVER\_CODE" field should be displayed with the specified properties (e.g., width, height, position, etc.) and should not allow insert or update operations.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include specific database queries.

# Navigation and Display of Cover Description in Rider Details

Type: AZBJ\_RIDER\_DETAILS2

Title: Navigation and Display of Cover Description in Rider Details  
  
Acceptance Criteria:  
1. When the user navigates to the cover description field, they should be able to move to the next record and automatically focus on the total rider investment field.  
2. The system should ensure that the cover description field is displayed with a maximum length of 500 characters and is right-justified.  
3. The cover description field should be displayed on the "Total Ride Sum Assured For Solution" canvas.  
  
Definition of Done:  
1. The cover description field is implemented and displayed correctly on the specified canvas.  
2. Navigation from the cover description field to the total rider investment field works as expected.  
3. The cover description field adheres to the specified visual attributes and constraints.  
4. All acceptance criteria are met and tested successfully.

# Manage Rider Details with Deletion Checkbox

Type: AZBJ\_RIDER\_DETAILS2

Title: Manage Rider Details with Deletion Checkbox  
  
Acceptance Criteria:  
1. When the checkbox labeled "Delete" is checked, the value should be set to 'Y'.  
2. When the checkbox labeled "Delete" is unchecked, the value should be set to 'N'.  
3. If the number of riders matches the current record, the focus should move to the delete item.  
4. If the number of riders does not match the current record, the focus should move to the next rider cover item.  
5. If the rider cover code is null, the focus should move to the delete item.  
  
Definition of Done:  
1. The checkbox functionality for marking a rider for deletion is implemented.  
2. The navigation logic based on the number of riders and rider cover code is implemented.  
3. The form is tested to ensure that the checkbox and navigation logic work as expected.  
4. The form is free of any Oracle Forms-specific terminology and is independent of the underlying technology.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- No specific DB queries provided in the XML content.

# View and Navigate Rider Cover Details

Type: AZBJ\_RIDER\_DETAILS2

Title: View and Navigate Rider Cover Details  
  
Acceptance Criteria:  
1. The rider cover details should be displayed in a text field with a maximum length of 200 characters.  
2. The text field should be positioned appropriately on the form and should be read-only.  
3. The text field should display the cover description in uppercase.  
4. The navigation within the form should allow moving to the next item seamlessly.  
5. The form should include a list of predefined values for the rider cover, such as 5, 8, 10, and "No".  
6. The form should have a tabbed interface for better organization and navigation.  
  
Definition of Done:  
1. The rider cover details are displayed correctly in the text field.  
2. The text field is read-only and displays the cover description in uppercase.  
3. The navigation within the form works as expected, allowing users to move to the next item.  
4. The predefined values for the rider cover are available for selection.  
5. The form has a tabbed interface for better organization and navigation.  
  
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.

# Interest Rate Field Functionality

Type: AZBJ\_RIDER\_DETAILS2

Title: Interest Rate Field Functionality  
  
Acceptance Criteria:  
1. The interest rate field should accept numeric values only.  
2. The interest rate field should allow users to insert and update values.  
3. The interest rate field should display a prompt labeled "LoanInt. Rt" above the field.  
4. Upon navigating away from the interest rate field, the system should automatically move the cursor to the next item in the sequence, specifically the "CV\_CHK\_DEL" field.  
5. The interest rate field should have predefined options for selection, including values 5, 8, and 10.  
  
Definition of Done:  
1. The interest rate field is implemented and accepts numeric input.  
2. Users can successfully insert and update the interest rate.  
3. The prompt "LoanInt. Rt" is displayed correctly above the interest rate field.  
4. Navigation from the interest rate field to the "CV\_CHK\_DEL" field works as specified.  
5. The predefined options (5, 8, 10) are available for selection in the interest rate field.  
  
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.

# Automatic Image Loading Based on Current Item and Block Context

Type: AZBJ\_RIDER\_DETAILS2

Detailed description: As a user, I want the system to automatically load and display the appropriate image based on the current item and block context, so that I can visually verify the data associated with the current item.  
  
Acceptance criteria:  
1. When navigating to the item, the system should load the image associated with the current item and display it.  
2. The system should determine the correct image to load based on the current block and item context.  
3. If the current block is 'PROPOSALFORM' or 'CORRECT\_VALUE', the system should apply a specific visual attribute to the item.  
4. The system should handle different blocks and items by setting the appropriate visual attributes and determining the correct image file name.  
5. The system should handle exceptions gracefully and log any errors encountered during the image loading process.  
  
Definition of Done:  
1. The image associated with the current item is correctly loaded and displayed.  
2. The correct visual attributes are applied based on the block and item context.  
3. The system handles all specified blocks and items as per the logic.  
4. Error handling and logging are implemented to capture any issues during the image loading process.  
5. The functionality 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 determine the maximum strip number for the current block and item:  
 ```sql  
 SELECT MAX(strip\_no)  
 INTO v\_strip\_no  
 FROM azbj\_strip\_field\_mapping a  
 WHERE block\_name = v\_current\_block  
 AND field\_name = v\_current\_item  
 AND proposal\_type = PK\_VARS.v\_proposal\_type  
 AND EXISTS  
 (SELECT 1  
 FROM azbj\_proposal\_stripes b  
 WHERE a.proposal\_type = b.proposal\_type  
 AND INSTR(stripe\_string, '~' || a.strip\_no || '~') > 0);  
 ```

# Navigation and Field Validation for Proposal Form

Type: AZBJ\_RIDER\_DETAILS2

Title: Navigation and Field Validation for Proposal Form  
  
Acceptance Criteria:  
1. When the "NEXT TAB" button is pressed, the system should:  
 - Navigate to the 'AZBJ\_RIDER\_DETAILS' section and display the first record.  
 - If the product group flag is 'Y', enable the 'COVERS\_GROUP' tab and navigate to the 'COVER\_TYPE' field.  
 - If the unit link flag is 'Y' or 'P', enable the 'UNIT\_LINK' tab and:  
 - If the 'PORTFOLIO\_STRATEGY' field is enabled, navigate to it, execute the 'WHEN-LIST-CHANGED' trigger, and navigate back to 'PORTFOLIO\_STRATEGY'.  
 - Otherwise, navigate to the 'CORRECT\_FUNDNAME' field in the 'AZBJ\_FUND\_DETAILS1' section.  
 - If the product code is 301, navigate to the 'RESET' button in the 'AZBJ\_SSO\_FUND\_DETAILS' section, execute its 'WHEN-BUTTON-PRESSED' trigger, and disable the 'CORRECT\_FUNDLIST' and 'CORRECT\_APPORT' fields in the 'AZBJ\_SSO\_FUND\_DETAILS1' section.  
 - If the unit link flag is not 'Y' or 'P', enable the 'BANK\_DETAILS' tab and:  
 - If the penny drop status is 'Penny Drop Successful', navigate to the 'PROPOSAL\_SIGN\_DATE' field.  
 - Otherwise, navigate to the 'BANK\_ACCOUNT\_NO' field.  
 - If the master policy number is null and the insured person's age is greater than 18, determine the safe district flag based on the address fields and update the 'SAFE\_DISTRICT\_FLAG' and 'SUSPECTED\_CASE' fields accordingly.  
 - Log debug information and check if the master policy number should disable the 'LOAN\_NUMBER' field based on system constants.  
  
2. When navigating to the next item, the system should:  
 - If the combo flag for the application number is 'Y', navigate to the 'PRODUCT\_ID\_PROD2' field and set the next item bean accordingly.  
 - Otherwise, navigate to the 'PROPOSAL\_DEPOSIT' field and set the next item bean accordingly.  
  
Definition of Done:  
- The navigation and field enabling/disabling logic should be implemented as described.  
- The system should correctly handle the conditions and navigate to the appropriate fields.  
- Debug information should be logged as specified.  
- The 'LOAN\_NUMBER' field should be disabled based on the system constants check.  
- The 'SAFE\_DISTRICT\_FLAG' and 'SUSPECTED\_CASE' fields should be updated based on the safe district determination logic.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Retrieve the master policy number for the given application number:  
 ```sql  
 SELECT a.master\_policy\_no  
 INTO v\_master\_policy\_no  
 FROM azbj\_proposal\_appln\_det a  
 WHERE a.appln\_no = TO\_NUMBER (:correct\_value.appln\_no)  
 AND de\_flag = 'D2';  
 ```  
  
- Count the number of system constants for disabling the loan number:  
 ```sql  
 SELECT COUNT ()  
 INTO v\_cnt  
 FROM azbj\_system\_constants  
 WHERE sys\_type = 'GROUP' AND sys\_code = 'DISABLE\_LOAN\_NUMBER'  
 AND sys\_desc = '|' || NVL (:correct\_value.master\_policy\_no, v\_master\_policy\_no) || '|';  
 ```

# Input Field for Total Rider Investment

Type: AZBJ\_RIDER\_DETAILS2

Title: Input Field for Total Rider Investment  
  
Acceptance Criteria:  
1. The input field for the total rider investment should only accept numerical values.  
2. The field should have a minimum value of 0 and a maximum length of 15 characters.  
3. When navigating through the form, if the user is on the last record and presses the "Next" key, the focus should move to the "OK" button in the solution product section.  
4. If the user is not on the last record and presses the "Next" key, the focus should move to the next record in the total rider investment field.  
5. The field should be visually styled with a specific font and alignment as per the design requirements.  
  
Definition of Done:  
- The input field for the total rider investment is implemented and meets the specified acceptance criteria.  
- The navigation logic for the "Next" key is correctly implemented and tested.  
- The field is visually styled according to the design specifications.  
- All functionality 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 direct database operations.

# Manage Apportionment of Funds within a Portfolio

Type: UL\_AT\_PORTFOL\_2

Title: Manage Apportionment of Funds within a Portfolio  
  
Acceptance Criteria:  
1. The system should allow users to input and update the apportionment percentage for each fund within the portfolio.  
2. The system should display the fund ID and fund name for each fund in the portfolio.  
3. The system should provide a button to delete a fund from the portfolio.  
4. The system should display a checkbox to mark a fund for deletion.  
5. The system should calculate and display the total fund value based on the apportionment percentages.  
6. The system should provide a list of available funds for selection, filtered by specific criteria:  
 - The fund must be active within a specified date range.  
 - The fund must be associated with a specific product ID.  
  
Definition of Done:  
- Users can successfully input and update the apportionment percentage for each fund.  
- The fund ID and fund name are displayed correctly for each fund.  
- Users can delete a fund from the portfolio using the provided button.  
- Users can mark a fund for deletion using the checkbox.  
- The total fund value is calculated and displayed correctly.  
- The list of available funds is filtered based on the specified criteria and displayed for user selection.  
  
DB queries for Table reference CRUD operations only (With Usage):  
```sql  
SELECT fund\_short\_name AS FUND\_ID, 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 = :correct\_value.product\_CODE  
AND :susac.sa\_daterecd >= NVL(start\_date, '01-apr-2005')  
AND :susac.sa\_daterecd <= NVL(end\_date, '01-jan-3000');  
```  
  
This query is used to fetch the list of available funds for selection, filtered by the specified criteria.

# Validate Apportionment Percentage and Perform Actions Based on Total

Type: UL\_AT\_PORTFOL\_2

Title: Validate Apportionment Percentage and Perform Actions Based on Total  
  
Acceptance Criteria:  
1. When the user enters an apportionment percentage, the system should validate that the value is between 0 and 100.  
2. If the entered value is outside this range, an error message should be displayed: "Apportionment percentage should be between 0 and 100".  
3. The system should sum the apportionment percentages across all records in the 'AZBJ\_FUND\_DETAILS' block.  
4. If the total apportionment percentage equals 100, the system should enable specific fields based on the product code and product definition.  
5. If the total apportionment percentage does not equal 100, the system should navigate to the next record and set focus on a specific field.  
  
Definition of Done:  
1. The apportionment percentage validation is implemented and tested.  
2. Error messages are displayed correctly when the validation fails.  
3. The system correctly sums the apportionment percentages and performs the necessary actions based on the total.  
4. The specific fields are enabled or focus is set as per the defined conditions.  
5. 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 direct database queries that can be executed independently of Oracle Forms constructs.

# Delete Fund from Portfolio

Type: UL\_AT\_PORTFOL\_2

Title: Delete Fund from Portfolio  
  
Acceptance Criteria:  
1. When the delete fund button is pressed, the system should navigate to the portfolio block and identify the total number of records.  
2. If there is only one fund in the portfolio, the system should display a message indicating that at least one fund is required.  
3. If there are multiple funds, the system should check each record to see if it is marked for deletion.  
4. If a fund is marked for deletion and there are more than one fund, the system should delete the record and update the total number of records.  
5. The system should loop through all records to ensure all marked funds are deleted.  
6. After processing, the system should navigate back to the first record.  
  
Definition of Done:  
- The delete fund functionality works as described in the acceptance criteria.  
- The system ensures that at least one fund remains in the portfolio.  
- Appropriate messages are displayed to the user when attempting to delete the last remaining fund.  
- The system correctly updates the portfolio after deletion of marked funds.  
- The functionality is tested and verified to work without errors.  
  
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.

# Delete Fund from Portfolio

Type: UL\_AT\_PORTFOL\_2

Title: Delete Fund from Portfolio  
  
Acceptance Criteria:  
1. When the delete button is pressed, the system should check the total number of records in the portfolio.  
2. If there is only one fund, the system should display a message indicating that at least one fund is required under Auto Transfer.  
3. If there are multiple funds and the selected fund is marked for deletion, the system should:  
 - Decrease the total number of records by one.  
 - Delete the selected fund.  
 - Update the portfolio to reflect the deletion.  
4. The system should navigate to the first record after the deletion process is complete.  
  
Definition of Done:  
- The delete functionality should be implemented and tested.  
- The system should display appropriate messages based on the number of funds.  
- The portfolio should be updated correctly after a fund is deleted.  
- The system should navigate to the first record after the deletion process.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- No specific DB queries provided in the XML content that can be directly executed without modification.

# Display and Handle Fund Name in FUND\_NAME\_UL Field

Type: UL\_AT\_PORTFOL\_2

Title: Display and Handle Fund Name in FUND\_NAME\_UL Field  
  
Acceptance Criteria:  
1. When the "FUND\_NAME\_UL" field is double-clicked, the system should display a list of available funds for selection.  
2. Upon selecting a fund, the "FUND\_NAME\_UL" field should display the selected fund name.  
3. The "FUND\_NAME\_UL" field should be read-only and not allow direct input or modification.  
4. When the "FUND\_NAME\_UL" field is focused, the system should load the corresponding image based on the proposal type and number of pages.  
5. If the proposal type is 'N' and the number of pages is 50, the system should zoom into a specific image.  
6. When the user navigates to the next item, the system should clear any visual attributes from the current item and move to the "APPORTIONMENT" field.  
  
Definition of Done:  
- The "FUND\_NAME\_UL" field displays the selected fund name from the LOV.  
- The field is read-only and does not allow direct input.  
- The system loads and displays the appropriate image when the field is focused.  
- The system handles double-click and navigation actions as specified.  
- All acceptance criteria are met, and the functionality is tested and verified.  
  
DB queries for Table reference CRUD operations only (With Usage):  
```sql  
SELECT fund\_short\_name AS FUND\_ID, 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 = :correct\_value.product\_CODE  
 AND :susac.sa\_daterecd >= NVL(start\_date, '01-apr-2005')  
 AND :susac.sa\_daterecd <= NVL(end\_date, '01-jan-3000');  
```  
- This query is used to fetch the list of available funds for the LOV based on the product code and date range.

# Manage Apportionment of Funds within a Portfolio

Type: UL\_AT\_PORTFOL

Title: Manage Apportionment of Funds within a Portfolio  
  
Acceptance Criteria:  
1. The system should allow users to input and update the apportionment percentage for each fund.  
2. The system should display the fund name and ID for each fund in the portfolio.  
3. The system should provide a checkbox to mark a fund for deletion.  
4. The system should include a button to delete the selected fund from the portfolio.  
5. The system should calculate and display the total apportionment value for all funds in the portfolio.  
6. The system should provide a list of available funds for selection, filtered based on specific criteria (e.g., product code and date range).  
  
Definition of Done:  
1. Users can successfully input and update the apportionment percentage for each fund.  
2. The fund name and ID are displayed correctly for each fund in the portfolio.  
3. Users can mark a fund for deletion using a checkbox.  
4. The delete button successfully removes the selected fund from the portfolio.  
5. The total apportionment value is calculated and displayed correctly.  
6. The list of available funds is filtered and displayed based on the specified criteria.  
  
DB queries for Table reference CRUD operations only (With Usage):  
```sql  
SELECT fund\_short\_name AS FUND\_ID, 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 = :correct\_value.product\_CODE  
AND :susac.sa\_daterecd >= NVL(start\_date, '01-apr-2005')  
AND :susac.sa\_daterecd <= NVL(end\_date, '01-jan-3000');  
```  
  
This query is used to fetch the list of available funds based on the product code and date range criteria.

# Delete Fund Functionality

Type: UL\_AT\_PORTFOL

Detailed description: As a user, I want to be able to delete a fund from the portfolio under certain conditions, 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 last record in the portfolio block.  
2. The system should determine the total number of records in the portfolio.  
3. If there is only one fund in the portfolio, the system should display a message indicating that at least one fund is required.  
4. If there are multiple funds, the system should loop through the records and delete the ones marked for deletion.  
5. The system should ensure that the portfolio block is updated and the first record is displayed after the deletion process.  
  
Definition of Done:  
- The delete fund functionality is implemented and tested.  
- The system correctly handles the deletion of funds while ensuring at least one fund remains.  
- Appropriate messages are displayed to the user when attempting to delete the last remaining fund.  
- The portfolio block is updated and displays the first record after the deletion process.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include direct database queries.

# Validate Apportionment Percentage and Navigate Based on Sum

Type: UL\_AT\_PORTFOL

Title: Validate Apportionment Percentage and Navigate Based on Sum  
  
Acceptance Criteria:  
1. When the user enters an apportionment percentage, the system should validate that the value is between 0 and 100.  
2. If the entered apportionment percentage is outside the range of 0 to 100, the system should display a warning message: "Apportionment percentage should be between 0 and 100" and prevent further actions.  
3. The system should calculate the sum of all apportionment percentages in the current block and ensure it equals 100.  
4. If the sum of apportionment percentages equals 100 and the product code is 71 with a specific product definition, the system should enable certain fields and navigate to the appropriate block and item.  
5. If the sum of apportionment percentages does not equal 100, the system should navigate to a specific item in the current block.  
  
Definition of Done:  
1. The apportionment percentage validation logic is implemented and tested.  
2. The system displays the appropriate warning message when the apportionment percentage is outside the valid range.  
3. The system correctly calculates the sum of apportionment percentages and takes the appropriate actions based on the sum and product code.  
4. The navigation logic to the appropriate block and item is implemented and tested.  
5. 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 for CRUD operations.

# Delete Fund and Navigate to Next Item

Type: UL\_AT\_PORTFOL

Detailed description: As a user, I want to be able to delete a fund from the portfolio, ensuring that at least one fund remains under Auto Transfer. Additionally, I want the system to handle navigation and validation when moving to the next item in the form, based on specific conditions related to rider cover codes and portfolio strategies.  
  
Acceptance criteria:  
1. When the "DELETE FUND" button is pressed:  
 - The system should navigate to the last record in the portfolio block.  
 - If there is only one fund, display a message indicating that at least one fund is required under Auto Transfer.  
 - If there are multiple funds and the fund marked for deletion is flagged, delete the record and navigate back to the portfolio block.  
 - Ensure the system loops through all records to check for deletions and navigates to the first record after processing.  
  
2. When navigating to the next item:  
 - The system should count the number of records in the rider details block.  
 - For each record, check the rider cover code against specific conditions.  
 - Update a variable to indicate the presence of certain rider cover codes.  
 - Based on the value of a flag, navigate to the appropriate item in the form (e.g., nominee name, portfolio strategy, or fund name).  
 - Enable or disable the portfolio strategy item based on product definitions and other conditions.  
  
Definition of Done:  
- The "DELETE FUND" button functionality is implemented and tested to ensure it meets the acceptance criteria.  
- The navigation and validation logic for moving to the next item is implemented and tested.  
- All related messages and validations are displayed correctly.  
- The system handles all specified conditions and navigations without errors.  
- User acceptance testing is completed, and the feature is approved by stakeholders.  
  
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.

# Display Fund Name Based on Proposal Type and Pages

Type: UL\_AT\_PORTFOL

Detailed description: As a user, I want the system to display the appropriate fund name in the "FUND\_NAME" field based on the selected proposal type and the number of pages, so that I can accurately view and select the fund details.  
  
Acceptance criteria:  
1. When the "FUND\_NAME" field is accessed, the system should load the field from the image based on the current item.  
2. If the proposal type is 'N' and the number of pages is 50, the system should zoom into the image 'PROPOSALFORM.image51' at a specified zoom percentage.  
3. On double-clicking the "FUND\_NAME" field, the system should display a list of values (LOV) for fund selection.  
4. When the user presses a key to list values, the system should show the LOV for fund selection.  
5. Navigating to the next item should clear any visual attributes set on the current item and move the focus to the "APPORTIONMENT" field.  
6. When the "FUND\_NAME" field is instantiated, it should load the field from the image based on the current item.  
  
Definition of Done:  
- The "FUND\_NAME" field should correctly display the fund name based on the proposal type and number of pages.  
- The image zoom functionality should work as expected when the conditions are met.  
- The LOV for fund selection should be displayed on double-click or key press.  
- Navigation to the next item should work seamlessly, clearing any visual attributes from the current item.  
- The field should load from the image correctly upon instantiation.  
  
DB queries for Table reference CRUD operations only (With Usage):  
```sql  
SELECT fund\_short\_name AS FUND\_ID, 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 = :correct\_value.product\_CODE  
 AND :susac.sa\_daterecd >= NVL(start\_date, '01-apr-2005')  
 AND :susac.sa\_daterecd <= NVL(end\_date, '01-jan-3000');  
```  
- This query is used to fetch the fund short name and full name for the LOV based on the product code and date received.

# Manage Fund Details

Type: AZBJ\_FUND\_DETAILS

Title: Manage Fund Details  
  
Acceptance Criteria:  
1. The system should display a list of fund details, including the total fund value, fund ID, apportionment percentage, and fund name.  
2. The total fund value should be automatically calculated as the sum of the apportionment values.  
3. Users should be able to add new fund details by entering the fund ID and apportionment percentage.  
4. Users should be able to update existing fund details, including the apportionment percentage.  
5. Users should be able to delete fund details by selecting the corresponding checkbox and clicking the delete button.  
6. The system should provide a reset button to clear the current form inputs.  
7. The fund details should be displayed in a tabbed interface for better organization.  
8. The system should ensure that the fund ID is unique and follows the specified format.  
9. The system should validate that the apportionment percentage is a number and within the allowed range.  
10. The system should fetch fund details from the database based on specific criteria, including product code and date range.  
  
Definition of Done:  
- The user interface for managing fund details is implemented and accessible.  
- The total fund value is correctly calculated and displayed.  
- Users can add, update, and delete fund details as specified.  
- The reset functionality works as expected.  
- The tabbed interface is implemented and functional.  
- Validation rules for fund ID and apportionment percentage are enforced.  
- The system fetches and displays fund details based on the specified criteria.  
- All acceptance criteria are met, and the functionality is tested and verified.  
  
DB queries for Table reference CRUD operations only(With Usage):  
```sql  
SELECT fund\_short\_name AS FUND\_ID, 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 = :correct\_value.product\_CODE  
AND :susac.sa\_daterecd >= NVL(start\_date, '01-apr-2005')  
AND :susac.sa\_daterecd <= NVL(end\_date, '01-jan-3000');  
```  
  
This query is used to fetch fund details based on the product code and date range criteria.

# Reset Fund Details in Proposal Form

Type: AZBJ\_FUND\_DETAILS

Detailed description: As a user, I want to reset the fund details in the proposal form so that I can clear all the existing data and start with a fresh entry.  
  
Acceptance criteria:  
1. When the reset button is pressed, the system should navigate to the fund details section.  
2. The system should clear all the fields in the fund details section without validation.  
3. The fields `CORRECT\_APPORT`, `correct\_fund\_id`, and `CORRECT\_FUNDNAME` should be set to null.  
4. The system should navigate to the first record in the `AZBJ\_FUND\_DETAILS1` block.  
5. The system should navigate to the first record in the `AZBJ\_FUND\_DETAILS2` block.  
  
Definition of Done:  
- The reset functionality should be implemented and tested.  
- The system should clear the specified fields and navigate to the appropriate records as described in the acceptance criteria.  
- The functionality should be independent of the underlying technology and should not reference 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 direct database queries.

# Delete Specific Fund Records

Type: AZBJ\_FUND\_DETAILS

Title: Delete Specific Fund Records  
  
Acceptance Criteria:  
1. When the delete button is pressed, the system should navigate to the fund details section.  
2. The system should identify the total number of records in the fund details section.  
3. The system should iterate through each record in the fund details section.  
4. If a record has the fund validation flag set to 'Y', the system should delete that record.  
5. The system should continue to the next record if the fund validation flag is not set to 'Y'.  
6. After processing all records, the system should navigate back to the first record in the fund details section.  
  
Definition of Done:  
- The delete functionality is implemented and tested.  
- The system correctly navigates to the fund details section and processes each record.  
- Records with the fund validation flag set to 'Y' are deleted.  
- The system navigates back to the first record after processing all records.  
- The functionality is verified through unit tests and user acceptance tests.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- No direct database queries are provided in the XML content. The logic is implemented through navigation and record manipulation within the application.

# Delete Fund Records Based on Validation Flag

Type: AZBJ\_FUND\_DETAILS

Title: Delete Fund Records Based on Validation Flag  
  
Acceptance Criteria:  
1. When the delete button is pressed, the system should navigate to the fund details section and identify the total number of records.  
2. The system should iterate through each record in the fund details section.  
3. If a record has the fund validation flag set to 'Y', the system should delete that record.  
4. The system should continue to the next record if the fund validation flag is not set to 'Y'.  
5. After processing all records, the system should navigate back to the first record in the fund details section.  
  
Definition of Done:  
- The delete button functionality is implemented and tested.  
- The system correctly identifies and deletes records with the fund validation flag set to 'Y'.  
- The system navigates back to the first record after processing all records.  
- 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 logic is specific to Oracle Forms and involves navigation and record manipulation within the form, which cannot be directly translated to SQL queries without modification.

# Manage Fund Validation Process with Checkbox and Navigation

Type: AZBJ\_FUND\_DETAILS

Title: Manage Fund Validation Process with Checkbox and Navigation  
  
Acceptance Criteria:  
1. When the checkbox is checked, it should set the value to 'Y' indicating the fund is marked for deletion.  
2. When the checkbox is unchecked, it should set the value to 'N' indicating the fund is not marked for deletion.  
3. Upon changing the checkbox state, no additional action should be triggered.  
4. When navigating to the next item:  
 - If the current record is the last fund validation record, the focus should move to the command button for fund validation.  
 - If there are more fund validation records, the focus should move to the fund name field of the next record.  
 - If the fund name field of the next record is empty, the focus should again move to the command button for fund validation.  
  
Definition of Done:  
- The checkbox should correctly reflect the deletion status of the fund.  
- Navigation logic should work as specified, ensuring smooth user experience.  
- The functionality should be 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 direct database queries.

# Validate Apportionment Percentage and Navigate Based on Total Apportionment

Type: AZBJ\_FUND\_DETAILS

Title: Validate Apportionment Percentage and Navigate Based on Total Apportionment  
  
Acceptance Criteria:  
1. When the user enters an apportionment percentage, the system should validate that the value is between 0 and 100.  
2. If the entered value is outside this range, an error message should be displayed: "Apportionment percentage should be between 0 and 100".  
3. The system should calculate the total apportionment percentage for all records in the fund details section.  
4. If the total apportionment percentage equals 100, the system should enable specific fields and navigate to the appropriate section based on the product code and product definition.  
5. If the total apportionment percentage does not equal 100, the system should navigate to the next item in the fund details section.  
  
Definition of Done:  
- The apportionment percentage validation is implemented and tested.  
- Error messages are displayed correctly when the validation fails.  
- The total apportionment percentage calculation is accurate.  
- The system navigates to the correct sections based on the total apportionment percentage and other conditions.  
- All acceptance criteria are met and verified through testing.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Not applicable as the provided XML content does not include direct database CRUD operations that can be executed independently of Oracle Forms constructs.

# Reset Fund Details and Navigate Through Proposal Form

Type: AZBJ\_FUND\_DETAILS

Title: Reset Fund Details and Navigate Through Proposal Form  
  
Acceptance Criteria:  
1. When the "RESET" button is pressed, the system should:  
 - Navigate to the fund details section.  
 - Clear all fields in the fund details section without validation.  
 - Set the fields `CORRECT\_APPORT`, `correct\_fund\_id`, and `CORRECT\_FUNDNAME` to null.  
 - Navigate to the first record in the fund details section.  
 - Navigate to the first record in the secondary fund details section.  
  
2. When navigating to the next item:  
 - The system should count the number of records in the rider details section.  
 - For each record, if the `RIDER\_COVER\_CODE` does not start with 'L' and is not in a specified list, it should append 'Y' to a variable.  
 - If the `RIDER\_COVER\_CODE` is in another specified list, it should append 'FIB' to the same variable.  
 - Depending on the value of a flag, the system should navigate to different items in the proposal form.  
 - If certain conditions are met, the system should enable or disable the `PORTFOLIO\_STRATEGY` item and navigate to it.  
  
Definition of Done:  
- The "RESET" button functionality is implemented and tested.  
- The system correctly navigates and clears the fund details as specified.  
- The next item navigation logic is implemented and tested.  
- All specified conditions and navigations are handled correctly.  
- The functionality is verified through user acceptance testing.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- No direct database queries are provided in the XML content.

# Display and Navigate Fund Details

Type: AZBJ\_FUND\_DETAILS

Title: Display and Navigate Fund Details  
  
Acceptance Criteria:  
1. When the user navigates to the fund name field, the system should load the field from the image if the proposal type is 'N' and the number of pages is 50.  
2. If the form version is 10.x, the system should zoom into the image at 50% zoom level.  
3. On double-clicking the fund name field, the system should display a list of values (LOV) for fund selection.  
4. When the user presses the key to list values, the system should show the LOV for fund selection.  
5. On pressing the key to move to the next item, the system should clear the visual attribute of the current item and navigate to the 'Apportionment' field.  
6. When the user navigates to the fund name field, the system should load the field from the image.  
  
Definition of Done:  
- The fund name field should be non-editable and should not allow insert or update operations.  
- The system should correctly load and display the fund details based on the selected fund name.  
- The LOV for fund selection should be displayed correctly on double-click or key press.  
- Navigation to the next item should work seamlessly, clearing the visual attribute of the current item and moving to the 'Apportionment' field.  
  
DB queries for Table reference CRUD operations only (With Usage):  
```sql  
SELECT fund\_short\_name AS FUND\_ID, 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 = :correct\_value.product\_CODE  
 AND :susac.sa\_daterecd >= NVL(start\_date, '01-apr-2005')  
 AND :susac.sa\_daterecd <= NVL(end\_date, '01-jan-3000');  
```  
- This query is used to fetch the fund details for the LOV based on the product code and date received.

# Manage Fund Details

Type: AZBJ\_SSO\_FUND\_DETAILS1

Title: Manage Fund Details  
  
Acceptance Criteria:  
1. The system should allow users to view a list of funds.  
2. The system should allow users to insert new fund details, including fund ID, fund name, and apportionment percentage.  
3. The system should allow users to update existing fund details.  
4. The system should validate fund details against a predefined list to ensure data accuracy.  
5. The system should restrict certain fields from being updated or inserted based on predefined conditions.  
  
Definition of Done:  
1. Users can successfully view a list of funds.  
2. Users can insert new fund details and the data is saved correctly.  
3. Users can update existing fund details and the changes are saved correctly.  
4. The system validates fund details against a predefined list and prevents invalid data entry.  
5. The system restricts updates and inserts for certain fields based on predefined conditions.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- The system should use the following query to fetch fund details for validation:  
 ```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 = :correct\_value.product\_code   
 AND COVER\_CODE = :azbj\_rider\_details.rider\_cover\_code;  
 ```  
  
- The system should use the following query to fetch fund details for the list of values (LOV):  
 ```sql  
 SELECT fund\_short\_name FUND\_ID, 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 = :correct\_value.product\_CODE   
 AND :susac.sa\_daterecd >= NVL(start\_date, '01-apr-2005')   
 AND :susac.sa\_daterecd <= NVL(end\_date, '01-jan-3000');  
 ```

# Input and Validate Apportionment Percentage for Fund Details

Type: AZBJ\_SSO\_FUND\_DETAILS1

Detailed description: As a user, I want to input and validate the apportionment percentage for fund details, ensuring that the value is within the acceptable range and correctly updates the related fund details.  
  
Acceptance criteria:  
1. The system should prompt the user to enter a percentage if the apportionment field is left empty.  
2. The system should validate that the entered percentage is between 0 and 100. If the value is outside this range, a warning message should be displayed, and the process should be halted.  
3. Upon entering a valid percentage, the system should update the related fund details with the entered percentage and other associated values.  
4. The system should clear the input fields after updating the fund details.  
5. The system should handle image loading based on the proposal type and display the appropriate image.  
  
Definition of Done:  
- The user can successfully input a percentage value.  
- The system validates the input and provides appropriate feedback if the value is invalid.  
- The related fund details are updated correctly with the entered percentage and associated values.  
- The input fields are cleared after the update.  
- The appropriate image is loaded and displayed based on the proposal type.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- No direct database queries are provided in the XML content that can be executed without modification. The logic involves form-specific constructs and procedures that need to be translated into appropriate database operations if required.

# Fund Selection and Validation for 'Name of Fund' Field

Type: AZBJ\_SSO\_FUND\_DETAILS1

Title: Fund Selection and Validation for 'Name of Fund' Field  
  
Acceptance Criteria:  
1. When the 'Name of Fund' field is double-clicked, a list of values (LOV) should be displayed, allowing the user to select a fund.  
2. When the 'Name of Fund' field is navigated to using the keyboard, the LOV should be displayed if the field is empty.  
3. The LOV should be populated with fund names based on the product code and cover code.  
4. The selected fund should be validated against the list of available funds.  
  
Definition of Done:  
1. The 'Name of Fund' field should display an LOV when double-clicked or when navigated to using the keyboard if the field is empty.  
2. The LOV should be correctly populated with fund names based on the product code and cover code.  
3. The selected fund should be validated and saved correctly.  
4. The functionality should be tested and verified to ensure it meets the acceptance criteria.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- The LOV for the 'Name of Fund' field should be populated using the following query:  
 ```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 = :correct\_value.product\_code   
 AND COVER\_CODE = :azbj\_rider\_details.rider\_cover\_code;  
 ```

# Validate and Process Correct Apportionment Percentage

Type: AZBJ\_SSO\_FUND\_DETAILS1

Title: Validate and Process Correct Apportionment Percentage  
  
Acceptance Criteria:  
1. If the "Correct Apportionment" field is left empty, an error message should be displayed prompting the user to enter a percentage.  
2. If the entered percentage is less than 0 or greater than 100, a warning message should be displayed indicating that the apportionment percentage should be between 0 and 100.  
3. Upon entering a valid percentage, the system should update the corresponding fields in the "Fund Details" section with the values from the "Correct Apportionment" field and clear the input fields in the "Correct Apportionment" section.  
4. The system should navigate to the next item in the form after processing the valid input.  
  
Definition of Done:  
- The validation logic for the "Correct Apportionment" field is implemented and tested.  
- Error and warning messages are displayed as per the acceptance criteria.  
- The corresponding fields in the "Fund Details" section are updated correctly.  
- The input fields in the "Correct Apportionment" section are cleared after a valid input is processed.  
- The form navigates to the next item after processing the input.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- No direct database queries are mentioned in the provided XML content.

# Fund Selection and Validation for 'Name of Fund' Field

Type: AZBJ\_SSO\_FUND\_DETAILS1

Title: Fund Selection and Validation for 'Name of Fund' Field  
  
Acceptance Criteria:  
1. When the 'Name of Fund' field is double-clicked, a list of values (LOV) should be displayed, allowing the user to select a fund.  
2. When the 'Name of Fund' field is navigated to using the keyboard, the LOV should be displayed if the field is empty.  
3. The LOV should be populated with fund names based on the product code and cover code.  
4. The selected fund should be validated and updated in the 'Name of Fund' field.  
  
Definition of Done:  
1. The 'Name of Fund' field should display an LOV when double-clicked or when navigated to using the keyboard if the field is empty.  
2. The LOV should be correctly populated with fund names based on the product code and cover code.  
3. The selected fund should be validated and updated in the 'Name of Fund' 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 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 = :correct\_value.product\_code   
AND COVER\_CODE = :azbj\_rider\_details.rider\_cover\_code;  
```  
  
```sql  
SELECT fund\_short\_name FUND\_ID, 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 = :correct\_value.product\_CODE   
AND :susac.sa\_daterecd >= NVL(start\_date, '01-apr-2005')   
AND :susac.sa\_daterecd <= NVL(end\_date, '01-jan-3000');  
```

# Automatic Image Loading for Fund List Field

Type: AZBJ\_SSO\_FUND\_DETAILS1

Title: Automatic Image Loading for Fund List Field  
  
Acceptance Criteria:  
1. When the user navigates to the "Fund List" field, the system should trigger the loading of the appropriate image based on the current item and block context.  
2. The system should determine the current block and item, and set the visual attributes accordingly.  
3. The system should identify the correct image file based on the proposal type and other conditions, and display it in the designated image area.  
4. If the current block is related to family details and the proposal type is 'N', the system should adjust the image strip number based on specific conditions.  
5. The system should handle exceptions gracefully and log any errors encountered during the image loading process.  
  
Definition of Done:  
- The image is loaded and displayed correctly when the user navigates to the "Fund List" field.  
- The visual attributes of the current item are set appropriately based on the block context.  
- The correct image file is identified and displayed without errors.  
- All specified conditions and exceptions are handled as per the acceptance criteria.  
- The functionality is tested and verified to ensure it meets the user requirements.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- The following query is used to determine the maximum strip number for the current block and item:  
 ```sql  
 SELECT MAX(strip\_no)  
 INTO v\_strip\_no  
 FROM azbj\_strip\_field\_mapping a  
 WHERE block\_name = v\_current\_block  
 AND field\_name = v\_current\_item  
 AND proposal\_type = PK\_VARS.v\_proposal\_type  
 AND EXISTS  
 (SELECT 1  
 FROM azbj\_proposal\_stripes b  
 WHERE a.proposal\_type = b.proposal\_type  
 AND INSTR(stripe\_string, '~' || a.strip\_no || '~') > 0);  
 ```

# Validate and Process 'Correct' Field Input in Fund Details

Type: AZBJ\_SSO\_FUND\_DETAILS1

Detailed description: As a user, I want to ensure that the system validates the input values for the 'Correct' field in the fund details section, so that only valid values are accepted and appropriate actions are taken based on the input.  
  
Acceptance criteria:  
1. The system should display an error message if the input value for the 'Correct' field is not one of the valid options (1, 2, 3, 4, 7, 8).  
2. The system should prompt the user to enter a value if the 'Correct' field is left empty.  
3. If the 'Correct' field value is 1 or 2 and the fund list values in two different sections are the same, an error message should be displayed.  
4. If the 'Correct' field value is 3 and the fund list values in two different sections are different, an error message should be displayed.  
5. If the 'Correct' field value is 4, 7, or 8, the system should display a confirmation alert asking if the user wishes to add more funds.  
6. Based on the user's response to the confirmation alert, the system should navigate to the appropriate section or item.  
7. The system should update the fund details based on the value of the 'Correct' field:  
 - If the value is 1, update the fund details with the first set of fund list values.  
 - If the value is 2 or 3, update the fund details with the second set of fund list values.  
8. The system should allow updates and insertions for specific fields if the 'Correct' field value is not 1, 2, or 3.  
9. The system should navigate to the appropriate section or item based on the value of the 'Correct' field and the presence of fund list values.  
  
Definition of Done:  
- The system correctly validates the input values for the 'Correct' field.  
- Appropriate error messages and alerts are displayed based on the input values.  
- The system navigates to the correct sections or items based on user input and confirmation responses.  
- Fund details are updated correctly based on the value of the 'Correct' field.  
- 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 database queries for CRUD operations.

# View and Select Fund Details

Type: AZBJ\_SSO\_FUND\_DETAILS2

Title: View and Select Fund Details  
  
Acceptance Criteria:  
1. The system should display a list of funds filtered by the following criteria:  
 - The fund's short name must match the product code.  
 - The current date must fall within the fund's start and end date range.  
2. The list of funds should include both the short name and full name of each fund.  
3. The user should be able to view the list of funds in a dropdown or similar selection component.  
  
Definition of Done:  
- The list of funds is displayed correctly based on the specified filtering criteria.  
- The user can select a fund from the list.  
- The selected fund's details are correctly captured and stored.  
  
DB queries for Table reference CRUD operations only(With Usage):  
```sql  
SELECT fund\_short\_name AS FUND\_ID, 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 = :correct\_value.product\_CODE  
AND :susac.sa\_daterecd >= NVL(start\_date, '01-apr-2005')  
AND :susac.sa\_daterecd <= NVL(end\_date, '01-jan-3000');  
```  
  
This query is used to fetch the list of funds based on the specified criteria.

# Automatic Image Loading for FUNDLIST2 Field

Type: AZBJ\_SSO\_FUND\_DETAILS2

Detailed description: As a user, I want the system to automatically load and display the appropriate image based on the current item and block context when I navigate to the "FUNDLIST2" field, so that I can view the relevant image data without manual intervention.  
  
Acceptance criteria:  
1. When the user navigates to the "FUNDLIST2" field, the system should trigger the loading of the appropriate image based on the current item and block context.  
2. The system should determine the current block and item, and set visual attributes accordingly.  
3. The system should identify the correct image file based on the proposal type and other conditions, and display it in the designated image area.  
4. If the current block is related to family details and the proposal type is 'N', the system should set specific strip numbers based on the family member type.  
5. The system should handle exceptions gracefully and log any errors encountered during the image loading process.  
  
Definition of Done:  
- The image loading functionality is implemented and tested.  
- The system correctly identifies and displays the appropriate image based on the current item and block context.  
- All acceptance criteria are met, and the functionality is verified through testing.  
- Error handling and logging are implemented to capture any issues during the image loading process.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- The system should execute a query to fetch the maximum strip number from the `azbj\_strip\_field\_mapping` table based on the current block, item, and proposal type.  
- The system should check for the existence of related records in the `azbj\_proposal\_stripes` table to determine the appropriate strip number.  
  
Example query:  
```sql  
SELECT MAX(strip\_no)  
INTO v\_strip\_no  
FROM azbj\_strip\_field\_mapping a  
WHERE block\_name = v\_current\_block  
 AND field\_name = v\_current\_item  
 AND proposal\_type = PK\_VARS.v\_proposal\_type  
 AND EXISTS (  
 SELECT 1  
 FROM azbj\_proposal\_stripes b  
 WHERE a.proposal\_type = b.proposal\_type  
 AND INSTR(stripe\_string, '~' || a.strip\_no || '~') > 0  
);  
```

# View and Manage Membership Details

Type: AZBJ\_MEMBERSHIP\_DETAILS

Title: View and Manage Membership Details  
  
Acceptance Criteria:  
1. The membership number should be displayed in uppercase and should not be editable.  
2. The membership sum assured should be displayed in uppercase, should be a numeric value, and should not be editable.  
3. There should be a button labeled "POPULATE" that, when clicked, performs a specific action related to populating membership details. This button should be visible only under certain conditions.  
  
Definition of Done:  
1. The membership number and sum assured fields are displayed correctly and meet the specified criteria.  
2. The "POPULATE" button is implemented and performs the intended action when clicked.  
3. The user interface is intuitive and user-friendly.  
4. All functionalities are tested and verified to be working as expected.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- No specific DB queries provided in the XML content.

# Display and Navigation of Membership Number

Type: AZBJ\_MEMBERSHIP\_DETAILS

Title: Display and Navigation of Membership Number  
  
Acceptance Criteria:  
1. The membership number field should be displayed with a white background and positioned at coordinates (85, 32) on the screen.  
2. The membership number should be displayed in uppercase letters.  
3. The field should be read-only, meaning users cannot insert or update the membership number.  
4. The field should have a prompt labeled "Membership No" that is centered and bold.  
5. Navigation should be enabled such that pressing the key to move to the next item should take the user to the "MEM\_SUM\_ASSURED" field.  
  
Definition of Done:  
1. The membership number field is displayed correctly with the specified properties.  
2. The field is read-only and displays the membership number in uppercase.  
3. The prompt for the membership number is correctly labeled and formatted.  
4. Navigation to the "MEM\_SUM\_ASSURED" field works as expected when the user attempts to move 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.

# Populate Membership Details

Type: AZBJ\_MEMBERSHIP\_DETAILS

Detailed description: As a user, I want to populate membership details based on the application number and proposal type, so that I can view and manage the membership information efficiently.  
  
Acceptance criteria:  
1. When the populate button is pressed, the system should:  
 - Clear the current membership details.  
 - Retrieve membership details (membership number and sum assured) from the database where the application number matches the provided value, the de\_flag is either 'D1' or 'D2', and the proposal type is 'PSSS'.  
 - Populate the retrieved membership details into the membership details section.  
 - Display a message indicating the status of the proposal.  
  
2. If an error occurs during the process, the system should display an error message with the SQL error message.  
  
Definition of Done:  
- The populate button functionality is implemented and tested.  
- The membership details are correctly retrieved and displayed based on the specified criteria.  
- Error handling is in place and tested to ensure appropriate error messages are displayed.  
- The functionality is independent of any specific technology or Oracle Forms terminology.  
  
DB queries for Table reference CRUD operations only(With Usage):  
```sql  
SELECT MEMBERSHIP\_NO, MEM\_SUM\_ASSURED  
FROM AZBJ\_MEMBERSHIP\_DET  
WHERE appln\_no = :correct\_value.appln\_no  
AND de\_flag IN ('D1', 'D2')  
AND PROPOSAL\_TYPE = 'PSSS'  
ORDER BY PROPOSAL\_TYPE;  
```

# View and Navigate Membership Sum Assured

Type: AZBJ\_MEMBERSHIP\_DETAILS

Title: View and Navigate Membership Sum Assured  
  
Acceptance Criteria:  
1. The "Membership Sum Assured" field should be displayed with a white background and should be positioned correctly on the screen.  
2. The field should be read-only and should not allow any updates or insertions.  
3. The field should display the data in uppercase and should be of numeric type.  
4. Navigation within the form should be seamless:  
 - When navigating to the previous item, the focus should move to the "CORRECT" field in the "AZBJ\_MEMBERSHIP\_DETAILS1" block.  
 - When navigating to the next item, the system should iterate through all records in the "AZBJ\_MEMBERSHIP\_DETAILS" block and set the focus to the "CORRECT" field in the "AZBJ\_RIDER\_DETAILS1" block for each record.  
5. The field should load data from an image based on the current item and block context.  
  
Definition of Done:  
- The "Membership Sum Assured" field is displayed correctly and adheres to the specified properties.  
- The field is non-editable and displays data in uppercase.  
- Navigation to the previous and next items works as described.  
- The field correctly loads data from an image based on the current context.  
- 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 CRUD operations directly executable in the database.

# Manage Fund Details

Type: AZBJ\_FUND\_DETAILS1

Title: Manage Fund Details  
  
Acceptance Criteria:  
1. The system should allow users to view fund details, including fund ID, fund name, and apportionment percentage.  
2. Users should be able to insert new fund details, ensuring that the fund ID and fund name are validated against a predefined list.  
3. Users should be able to update existing fund details, including the fund ID, fund name, and apportionment percentage.  
4. The system should restrict the fund ID and fund name to uppercase characters.  
5. The system should ensure that the apportionment percentage is a numeric value and within the allowed range.  
6. The system should provide a list of valid funds for selection when inserting or updating fund details.  
  
Definition of Done:  
- The functionality to view, insert, and update fund details is implemented and tested.  
- Validation rules for fund ID, fund name, and apportionment percentage are enforced.  
- The user interface is updated to reflect the changes and is user-friendly.  
- The system is tested to ensure that it meets the acceptance criteria and works as expected.  
  
DB queries for Table reference CRUD operations only (With Usage):  
```sql  
SELECT fund\_short\_name AS FUND\_ID, 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 = :correct\_value.product\_CODE  
AND :susac.sa\_daterecd >= NVL(start\_date, '01-apr-2005')  
AND :susac.sa\_daterecd <= NVL(end\_date, '01-jan-3000');  
```  
  
This query is used to fetch valid fund IDs and fund names for validation and selection purposes.

# Fund Selection and Navigation

Type: AZBJ\_FUND\_DETAILS1

Title: Fund Selection and Navigation  
  
Acceptance Criteria:  
1. The system should display a list of available funds when the user interacts with the fund selection field.  
2. The list of available funds should be filtered based on the product code and the date received.  
3. The system should navigate to the fund name field after a fund is selected.  
  
Definition of Done:  
1. The fund selection field is implemented and displays a list of available funds.  
2. The list of funds is correctly filtered based on the product code and date received.  
3. The system successfully navigates to the fund name field after a fund is selected.  
4. All functionalities are tested and verified to be working as expected.  
  
DB queries for Table reference CRUD operations only(With Usage):  
```sql  
SELECT fund\_short\_name AS FUND\_ID, 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 = :correct\_value.product\_CODE  
AND :susac.sa\_daterecd >= NVL(start\_date, '01-apr-2005')  
AND :susac.sa\_daterecd <= NVL(end\_date, '01-jan-3000');  
```  
  
This query is used to fetch the list of available funds based on the product code and date received.

# Validate and Update Apportionment Percentage

Type: AZBJ\_FUND\_DETAILS1

Title: Validate and Update Apportionment Percentage  
  
Acceptance Criteria:  
1. The apportionment percentage must be entered by the user.  
2. The apportionment percentage must be between 0 and 100.  
3. If the apportionment percentage is not entered or is out of the valid range, an appropriate error message should be displayed.  
4. Upon entering a valid apportionment percentage, the system should update the fund details with the entered percentage and related fund information.  
5. The system should clear the input fields after updating the fund details.  
  
Definition of Done:  
- The user is prompted to enter an apportionment percentage if it is not provided.  
- The system validates that the apportionment percentage is within the range of 0 to 100.  
- Appropriate error messages are displayed for invalid input.  
- The fund details are updated with the correct apportionment percentage and related fund information.  
- The input fields are cleared after the update.  
- The user is navigated to the next item after the update.  
  
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 Available Funds for Fund Name Field

Type: AZBJ\_FUND\_DETAILS1

Title: Display Available Funds for Fund Name Field  
  
Acceptance Criteria:  
1. When the user focuses on the "Fund Name" field, the system should automatically display a list of available funds if the field is empty.  
2. The list of available funds should be fetched based on the product code and the date received.  
3. If the user selects a fund from the list, the selected fund name should be populated in the "Fund Name" field.  
4. The system should highlight the "Fund Name" field with a specific visual attribute when it is in focus.  
  
Definition of Done:  
1. The "Fund Name" field should trigger the display of a list of available funds when it is empty.  
2. The list of available funds should be correctly filtered based on the product code and date received.  
3. The selected fund name should be correctly populated in the "Fund Name" field.  
4. The "Fund Name" field should have a distinct visual attribute when in focus.  
5. All functionalities should be tested and verified to ensure they work as expected.  
  
DB queries for Table reference CRUD operations only (With Usage):  
```sql  
SELECT fund\_short\_name AS FUND\_ID, 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 = :correct\_value.product\_CODE  
AND :susac.sa\_daterecd >= NVL(start\_date, '01-apr-2005')  
AND :susac.sa\_daterecd <= NVL(end\_date, '01-jan-3000');  
```  
  
This query is used to fetch the list of available funds based on the product code and the date received.

# Fund ID Input and Validation

Type: AZBJ\_FUND\_DETAILS1

Title: Fund ID Input and Validation  
  
Acceptance Criteria:  
1. When the user inputs the fund ID, the system should validate it against the list of available funds.  
2. The system should display the fund name corresponding to the entered fund ID.  
3. The fund ID should be converted to uppercase automatically.  
4. The fund ID field should be hidden from view.  
5. The system should navigate to the fund name field after the fund ID is entered.  
  
Definition of Done:  
1. The fund ID input field is implemented and hidden from the user interface.  
2. The system validates the fund ID against the available funds.  
3. The fund name is displayed correctly based on the entered fund ID.  
4. The fund ID is automatically converted to uppercase.  
5. The system navigates to the fund name field after the fund ID is entered.  
  
DB queries for Table reference CRUD operations only(With Usage):  
```sql  
SELECT fund\_short\_name AS FUND\_ID, 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 = :correct\_value.product\_CODE  
AND :susac.sa\_daterecd >= NVL(start\_date, '01-apr-2005')  
AND :susac.sa\_daterecd <= NVL(end\_date, '01-jan-3000');  
```

# Validate and Process Apportionment Percentage

Type: AZBJ\_FUND\_DETAILS1

Title: Validate and Process Apportionment Percentage  
  
Acceptance Criteria:  
1. The apportionment percentage field should not be left empty. If it is empty, the system should display an error message: "Please enter the percentage."  
2. The apportionment percentage should be a number between 0 and 100. If the input is outside this range, the system should display a warning message: "Apportionment percentage should be between 0 and 100."  
3. Upon entering a valid apportionment percentage, the system should:  
 - Navigate to the "Fund Details" section.  
 - If the "Fund Name" field in the "Fund Details" section is not empty, move to the last record and then to the next record.  
 - Copy the apportionment percentage, fund ID, and fund name from the "Fund Details1" section to the corresponding fields in the "Fund Details" section.  
 - Clear the apportionment percentage, fund ID, and fund name fields in the "Fund Details1" section.  
 - Move the cursor to the next item in the "Fund Details1" section.  
  
Definition of Done:  
- The user can input a valid apportionment percentage.  
- The system validates the input and displays appropriate messages for invalid inputs.  
- The system correctly navigates and updates the "Fund Details" section based on the input.  
- The fields in the "Fund Details1" section are cleared after the update.  
- The cursor moves to the next item in the "Fund Details1" section after the update.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include any direct database queries.

# Fund Name Selection and Validation

Type: AZBJ\_FUND\_DETAILS1

Title: Fund Name Selection and Validation  
  
Acceptance Criteria:  
1. When the user navigates to the fund name field, the system should load the field from an image if the proposal type is 'N' and the number of pages is 50.  
2. If the fund name field is empty, the system should automatically display a list of values (LOV) for the user to select from.  
3. The LOV should be populated with fund names and their corresponding full names based on the product code and the date received.  
4. The system should allow the user to insert and update the fund name.  
5. The system should apply specific visual attributes to the fund name field based on the current block and item properties.  
  
Definition of Done:  
1. The fund name field should display a list of values (LOV) when the user navigates to it or when it is empty.  
2. The LOV should be correctly populated with relevant fund names and full names.  
3. The user should be able to select a fund name from the LOV and have it correctly inserted or updated in the field.  
4. The system should apply the appropriate visual attributes to the fund name field based on the current context.  
5. All functionalities should be tested and verified to ensure they work as expected.  
  
DB queries for Table reference CRUD operations only (With Usage):  
```sql  
SELECT fund\_short\_name AS FUND\_ID, 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 = :correct\_value.product\_CODE  
AND :susac.sa\_daterecd >= NVL(start\_date, '01-apr-2005')  
AND :susac.sa\_daterecd <= NVL(end\_date, '01-jan-3000');  
```  
  
This query is used to populate the list of values (LOV) for the fund name field based on the product code and the date received.

# Validation and Navigation for 'Correct' Field in Fund Details

Type: AZBJ\_FUND\_DETAILS1

Detailed description: As a user, I want to ensure that the system validates the input values for the 'Correct' field in the fund details section, so that only valid values are accepted and appropriate actions are taken based on the input.  
  
Acceptance criteria:  
1. The system should display an error message if the input value for the 'Correct' field is not among the valid options (1, 2, 3, 4, 7, 8).  
2. The system should prompt the user to enter a value if the 'Correct' field is left empty.  
3. If the 'Correct' field value is 1 or 2 and the fund names in the two fund details sections match, an error message should be displayed.  
4. If the 'Correct' field value is 3 and the fund names in the two fund details sections do not match, an error message should be displayed.  
5. If the 'Correct' field value is 4, 7, or 8, the system should display a confirmation alert asking if the user wishes to add more funds.  
6. Based on the user's response to the confirmation alert, the system should navigate to the appropriate section or block.  
7. The system should update the fund details based on the value of the 'Correct' field and the corresponding fund names and IDs.  
8. If the total fund value equals 100, the system should navigate to the appropriate section based on the product code.  
9. If the total fund value does not equal 100, the system should navigate to the next record or the appropriate section based on the input values.  
  
Definition of Done:  
- The system correctly validates the input values for the 'Correct' field.  
- Appropriate error messages and alerts are displayed based on the input values.  
- The system navigates to the correct sections or blocks based on the user's input and responses.  
- The fund details are updated accurately based on the input values.  
- 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 specific CRUD operations that can be directly executed in the database.

# Automatic Image Loading for Fund Name Field

Type: AZBJ\_FUND\_DETAILS1

Title: Automatic Image Loading for Fund Name Field  
  
Acceptance Criteria:  
1. When the user navigates to the "Fund Name" field, the system should check the proposal type and the number of pages.  
2. If the proposal type is 'N' and the number of pages is 50, the system should load the image corresponding to the current item.  
3. The system should determine the correct image file name based on the current block and item.  
4. The image should be displayed in the designated area, and the zoom level should be adjusted to fit the image properly.  
5. If the image cannot be loaded, an appropriate error message should be logged for debugging purposes.  
  
Definition of Done:  
- The image loading functionality is implemented and tested.  
- The system correctly identifies the proposal type and number of pages.  
- The correct image is displayed when the user navigates to the "Fund Name" field.  
- Error handling and logging are in place for any issues encountered during image loading.  
- The functionality is verified through user acceptance testing.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- The following query is used to fetch the fund details for the LOV (List of Values) associated with the "Fund Name" field:  
 ```sql  
 SELECT fund\_short\_name FUND\_ID, 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 = :correct\_value.product\_CODE  
 AND :susac.sa\_daterecd >= NVL(start\_date, '01-apr-2005')  
 AND :susac.sa\_daterecd <= NVL(end\_date, '01-jan-3000')  
 ```

# Manage SSO Fund Details

Type: AZBJ\_SSO\_FUND\_DETAILS

Title: Manage SSO Fund Details  
  
Acceptance Criteria:  
1. The system should display a list of SSO funds with the following details:  
 - Total Fund Value (calculated as the sum of the apportionment values).  
 - Fund ID (hidden from view but used for internal processing).  
 - Apportionment percentage.  
 - Name of the Fund.  
 - A checkbox to mark funds for deletion.  
2. The system should allow users to delete selected SSO funds by clicking a "DELETE SSO & FUND" button.  
3. The system should provide a "RESET" button to clear any selections or inputs made by the user.  
4. The system should ensure that the apportionment percentage is editable, while the name of the fund is read-only.  
5. The system should fetch fund details from the database based on specific criteria, including product code and date range.  
  
Definition of Done:  
- The user can view a list of SSO funds with the specified details.  
- The user can delete selected funds using the provided button.  
- The user can reset the form using the reset button.  
- The apportionment percentage is editable, and the name of the fund is read-only.  
- The system fetches and displays fund details based on the defined criteria.  
  
DB queries for Table reference CRUD operations only (With Usage):  
```sql  
SELECT fund\_short\_name AS FUND\_ID, 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 = :correct\_value.product\_CODE  
AND :susac.sa\_daterecd >= NVL(start\_date, '01-apr-2005')  
AND :susac.sa\_daterecd <= NVL(end\_date, '01-jan-3000');  
```  
  
This query is used to fetch the fund details based on the product code and date range criteria.

# Manage Fund Details within Proposal Form

Type: AZBJ\_SSO\_FUND\_DETAILS

Title: Manage Fund Details within Proposal Form  
  
Acceptance Criteria:  
1. When the checkbox for a fund is checked, the value should be set to 'Y'.  
2. When the checkbox for a fund is unchecked, the value should be set to 'N'.  
3. If the number of funds matches the current record, the system should navigate to the fund command item.  
4. If the number of funds does not match the current record, the system should navigate to the fund list item and move to the next record.  
5. If the fund list item is null, the system should navigate to the fund command item.  
  
Definition of Done:  
1. The checkbox functionality for managing fund details is implemented and tested.  
2. The navigation logic based on the number of funds and the current record is implemented and tested.  
3. The system correctly handles null values in the fund list item.  
4. All acceptance criteria are met, and the functionality is verified through user acceptance testing.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include specific database queries.

# Reset and Navigate Fund Details

Type: AZBJ\_SSO\_FUND\_DETAILS

Title: Reset and Navigate Fund Details  
  
Acceptance Criteria:  
1. When the "RESET" button is pressed, the system should:  
 - Navigate to the fund details section.  
 - Clear all fields related to fund details without validation.  
 - Set the fields `CORRECT\_APPORT`, `correct\_fund\_id`, and `CORRECT\_fundlist` to null.  
 - Navigate to the first record in the fund details section.  
 - Navigate to the first record in the secondary fund details section.  
  
2. When navigating to the next item, the system should:  
 - Check the number of records in the rider details section.  
 - Loop through each record and update the `v\_hc\_ci\_present` variable based on the `RIDER\_COVER\_CODE` field.  
 - If the `RIDER\_COVER\_CODE` matches specific values, update the `v\_hc\_ci\_present` variable accordingly.  
 - Depending on the value of `v\_unitlink\_flag`, navigate to either the nominee name or portfolio strategy fields.  
 - Enable or disable the portfolio strategy field based on the value of `v\_get\_portfolio\_strategy` and `v\_product\_defn`.  
  
Definition of Done:  
- The reset functionality should be implemented and tested to ensure it clears the fund details as specified.  
- The navigation and field updates should work correctly when moving to the next item.  
- All acceptance criteria should be met, and the functionality should be verified through testing.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- No specific DB queries are provided in the XML content for CRUD operations.

# Delete Selected Fund Details

Type: AZBJ\_SSO\_FUND\_DETAILS

User Story: Delete Selected Fund Details  
  
Detailed description:   
As a user, I want to be able to delete selected fund details from the list of fund records, so that I can manage and update the fund information efficiently.  
  
Acceptance criteria:  
1. When the "Delete SSO & Fund" button is pressed, the system should navigate to the fund details section.  
2. The system should identify the total number of fund records.  
3. The system should iterate through each fund record.  
4. If a fund record is marked for deletion (indicated by a specific field value), the system should delete that record.  
5. The system should continue to the next record if the current record is not marked for deletion.  
6. After processing all records, the system should navigate back to the first record.  
  
Definition of Done:  
- The "Delete SSO & Fund" button functionality is implemented and tested.  
- The system correctly navigates to the fund details section and processes each record as per the acceptance criteria.  
- The system successfully deletes the marked records and retains the unmarked ones.  
- The system navigates back to the first record after processing all records.  
- Unit tests and integration tests are written and passed.  
- User acceptance testing (UAT) is completed and signed off.  
  
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 Manage Fund List in Fund Details Section

Type: AZBJ\_SSO\_FUND\_DETAILS

Title: Display and Manage Fund List in Fund Details Section  
  
Acceptance Criteria:  
1. The "Fund Details" section should display a list of available funds.  
2. The list should include the following options: 5, 8, 10, No, and SSO.  
3. The field should be read-only and not allow any updates or insertions.  
4. When the user navigates to the "Fund Details" section, the system should load the field from an image based on the current item.  
5. Upon moving to the next item, the system should clear any visual attributes and navigate to the "Apportionment" field.  
  
Definition of Done:  
1. The "Fund Details" section displays the correct list of available funds.  
2. The field is read-only and does not allow updates or insertions.  
3. The system correctly loads the field from an image when the user navigates to it.  
4. The system clears visual attributes and navigates to the "Apportionment" field upon moving to the next item.  
5. All acceptance criteria are met, and the functionality is tested and verified.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- No specific CRUD operations are mentioned in the provided XML content.

# Validate Apportionment Percentage in Fund Details

Type: AZBJ\_SSO\_FUND\_DETAILS

Detailed description: As a user, I want to ensure that the apportionment percentage entered in the fund details section is validated to be between 0 and 100, so that the data integrity is maintained and the total apportionment sums up to 100%.  
  
Acceptance criteria:  
1. When the user enters an apportionment percentage, the system should validate that the value is between 0 and 100.  
2. If the entered value is outside this range, an error message should be displayed indicating that the apportionment percentage should be between 0 and 100.  
3. The system should calculate the total apportionment percentage for all records in the fund details section.  
4. If the total apportionment percentage equals 100, the system should navigate to the next relevant section.  
5. If the total apportionment percentage does not equal 100, the system should prompt the user to correct the entries.  
  
Definition of Done:  
- The validation logic for the apportionment percentage is implemented and tested.  
- Error messages are displayed correctly when the apportionment percentage is outside the valid range.  
- The total apportionment percentage calculation is accurate and triggers the appropriate navigation based on the result.  
- The user interface reflects the validation and navigation logic 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 direct database CRUD operations.

# Delete SSO & Fund Records Based on Condition

Type: AZBJ\_SSO\_FUND\_DETAILS

Title: Delete SSO & Fund Records Based on Condition  
  
Acceptance Criteria:  
1. When the "DELETE SSO & FUND" button is pressed, the system should navigate to the SSO Fund Details section.  
2. The system should identify the total number of records in the SSO Fund Details section.  
3. The system should iterate through each record in the SSO Fund Details section.  
4. If a record has the "CHK\_FUND" field set to 'Y', the system should delete that record.  
5. The system should continue to the next record if the "CHK\_FUND" field is not set to 'Y'.  
6. After processing all records, the system should navigate back to the first record in the SSO Fund Details section.  
  
Definition of Done:  
- The "DELETE SSO & FUND" button functionality is implemented and tested.  
- The system correctly navigates to the SSO Fund Details section and processes records as per the acceptance criteria.  
- The system successfully deletes records with the "CHK\_FUND" field set to 'Y'.  
- The system navigates back to the first record after processing all records.  
- All unit tests and integration tests pass.  
- The feature is reviewed and approved by the stakeholders.  
  
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 record manipulation within the form, which cannot be directly translated to SQL queries without modification.

# Reset Fund Details Form

Type: AZBJ\_SSO\_FUND\_DETAILS

Detailed description: As a user, I want to reset the fund details form so that I can clear all the current entries and start with a fresh form.  
  
Acceptance criteria:  
1. When the "RESET" button is pressed, the system should:  
 - Navigate to the "Fund Details" section.  
 - Clear all the fields in the "Fund Details" section without validation.  
 - Set the fields `CORRECT\_APPORT`, `correct\_fund\_id`, and `CORRECT\_fundlist` to null.  
 - Navigate to the first record in the "Fund Details" section.  
 - Navigate to the first record in the "Additional Fund Details" section.  
  
2. When navigating to the next item:  
 - The system should count the number of records in the "Rider Details" section.  
 - For each record, if the `RIDER\_COVER\_CODE` does not start with 'L' and is not in a specified list, it should append 'Y' to `v\_hc\_ci\_present`.  
 - If the `RIDER\_COVER\_CODE` is in another specified list, it should append 'FIB' to `v\_hc\_ci\_present`.  
 - Depending on the value of `v\_unitlink\_flag`, the system should navigate to either the "Nominee Name" or "Portfolio Strategy" fields.  
 - If `v\_get\_portfolio\_strategy` is 'Y' or `v\_product\_defn` is in a specified list, the "Portfolio Strategy" field should be enabled and navigated to; otherwise, it should be disabled.  
  
Definition of Done:  
- The "RESET" button functionality is implemented and tested.  
- The form fields are cleared and reset as per the acceptance criteria.  
- The navigation logic for the next item is implemented and tested.  
- All specified conditions and field updates are verified to work correctly.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- No direct database queries are provided in the XML content that can be executed without modification.

# Manage Image Details for Proposals and Applications

Type: IMAGE\_DET

Title: Manage Image Details for Proposals and Applications  
  
Acceptance Criteria:  
1. The system should allow the user to input and display the following details for each image:  
 - Proposal Number  
 - Application Number  
 - Image Type  
 - Scan Time  
 - Image Size (in KB)  
 - Image Path  
 - Number of Pages  
2. The system should provide a button to view the image.  
3. The system should provide a button to hide the image details.  
4. The system should ensure that the input fields for image details are visible and properly aligned on the user interface.  
  
Definition of Done:  
- The user can successfully input and view all the required image details.  
- The "View Image" button functions correctly and displays the image.  
- The "Hide" button functions correctly and hides the image details.  
- All input fields are visible and properly aligned on the user interface.  
- The system meets all the acceptance criteria and passes user acceptance testing.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Not applicable as the provided XML content does not include specific database queries.

# Implement Hide Button Functionality

Type: IMAGE\_DET

Title: Implement Hide Button Functionality  
  
User Story:  
As a user, I want to be able to hide certain elements on the screen by pressing a button, so that I can focus on the relevant information.  
  
Acceptance Criteria:  
1. The "Hide" button should be visible on the user interface.  
2. When the "Hide" button is pressed, the system should navigate to the "CORRECT\_VALUE.PROPTYPE" field.  
3. The button should have the following properties:  
 - Label: "Hide"  
 - Background Color: Gray  
 - Foreground Color: Dark Blue  
 - Font: Tahoma, Bold, Size 8  
4. The button should be positioned at X: 73, Y: 155 on the screen.  
  
Definition of Done:  
- The "Hide" button is implemented with the specified properties.  
- Pressing the "Hide" button successfully navigates to the "CORRECT\_VALUE.PROPTYPE" field.  
- The functionality is tested and verified to work as expected.

# View Image Functionality

Type: IMAGE\_DET

Detailed description: As a user, I want to be able to view images associated with a proposal, including those in PDF format, directly from the application interface. This functionality should allow me to download the image to a specified local directory and open it using the appropriate viewer based on the file type.  
  
Acceptance criteria:  
1. When the "View Image" button is pressed, the application should retrieve the image path and proposal number from the current record.  
2. The application should construct a local file path using the proposal number and image name.  
3. The image should be transferred from the server to the specified local directory.  
4. If the image is in PDF format, it should be opened using the default PDF viewer.  
5. If the image is in any other format, it should be opened using the default image viewer.  
6. The application should handle any exceptions that occur during the process gracefully, without crashing.  
  
Definition of Done:  
- The "View Image" button functionality is implemented and tested.  
- The image is successfully transferred to the local directory.  
- The image opens in the appropriate viewer based on its format.  
- Exception handling is in place to manage any errors during the process.  
- The feature is reviewed and approved by stakeholders.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Not applicable as the provided XML content does not include any direct database queries for CRUD operations.

# Manage AML Information for Documents

Type: AML

Detailed description: As a user, I want to manage the Anti-Money Laundering (AML) information for various documents, so that I can ensure compliance with regulatory requirements.  
  
Acceptance criteria:  
1. The system should allow the user to input and update the following fields for each document:  
 - Document Type  
 - Request Code  
 - Document Description  
 - Information  
 - Proof Type  
 - Document ID  
 - Document Date  
 - Old Policy Reference  
 - Proof Description  
 - Document Remarks  
  
2. The system should validate the "Information" and "Proof Type" fields against predefined lists to ensure data integrity.  
  
3. The system should display the "Proof Description" based on the selected "Proof Type".  
  
4. The system should handle errors gracefully, particularly for specific error codes (40212, 40102, 41058), by adjusting the system message level.  
  
5. The system should allow navigation to the clicked item if it is not a button.  
  
Definition of Done:  
- The user can successfully input and update AML information for documents.  
- The system validates the "Information" and "Proof Type" fields against predefined lists.  
- The "Proof Description" is displayed based on the selected "Proof Type".  
- Error handling is implemented for specific error codes.  
- Navigation to clicked items is functional.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- The system should fetch proof types and descriptions from the `azbj\_aml\_proof\_master` table where the `active\_flag` is 'Y' and the `partner\_type` and `document\_type` match the provided values.  
  
```sql  
SELECT DISTINCT Proof\_type, proof\_desc   
FROM azbj\_aml\_proof\_master  
WHERE active\_flag = 'Y'   
 AND partner\_type = NVL(:control.cn\_partner\_type, partner\_type)  
 AND document\_type = NVL(:AML.document\_type, document\_type);  
```

# Navigate to Next Item on Old Policy Reference Input

Type: AML

Title: Navigate to Next Item on Old Policy Reference Input  
  
Acceptance Criteria:  
1. 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.  
2. The system should update the next item bean to "Agents.bi\_no" upon navigation.  
  
Definition of Done:  
- The old policy reference field is present and functional.  
- The system successfully navigates to the "bi\_no" field in the "Agents" section when the user moves to the next item.  
- The next item bean is updated to "Agents.bi\_no" upon navigation.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- No specific DB queries are provided in the XML content for CRUD operations.

# Input and View Document Remarks in AML Section

Type: AML

Detailed description: As a user, I want to be able to input and view document remarks within the AML section of the application, so that I can provide additional context or comments related to the document.  
  
Acceptance criteria:  
1. The input field for document remarks should allow a maximum length of 300 characters.  
2. The field should be positioned appropriately within the AML section and should be clearly labeled as "Document Remarks".  
3. Upon pressing the key to move to the next item, the focus should shift to the "bi\_no" field in the Agents section.  
4. The document remarks field should be displayed on the "AML\_KYC" tab page.  
  
Definition of Done:  
1. The document remarks field is implemented and visible within the AML section.  
2. The field accepts up to 300 characters and is labeled correctly.  
3. Navigation to the next item works as specified, moving focus to the "bi\_no" field in the Agents section.  
4. The field is displayed on the correct tab page and is functional.  
  
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.

# Control Editability of VALUE Field Based on Document Type and Control Flag

Type: AML

Title: Control Editability of VALUE Field Based on Document Type and Control Flag  
  
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 and non-insertable.  
2. If the control flag `CHK\_EDIT\_AML` is set to 'Y', then the "VALUE" field should be editable and insertable.  
3. If neither of the above conditions is met, the "VALUE" field should be non-editable and non-insertable.  
  
Definition of Done:  
- The "VALUE" field's editability and insertability are correctly controlled based on the document type and the control flag.  
- The changes are tested and verified to ensure the field behaves as expected under different conditions.  
- The implementation is reviewed and approved by the project stakeholders.

# Document ID Field Validation and Processing

Type: AML

Detailed description: As a user, I want the system to validate and process the "Document ID" field based on specific conditions and trigger appropriate actions, so that the data integrity and workflow are maintained.  
  
Acceptance criteria:  
1. When the "Document ID" field is entered:  
 - If the "Document ID" is 'PP', the system should navigate to the "Passport Details" section and execute the related actions.  
 - If the "Document ID" is 'DL', the system should navigate to the "Driver's License Details" section and execute the related actions.  
 - If the "Document ID" is 'VI', the system should navigate to the "Voter's ID Details" section and execute the related actions.  
2. If the "Document ID" is related to 'IDENTITY PROOF' and the proof type is 'PC':  
 - The system should validate the PAN number using the `VALIDATE\_PAN` procedure.  
 - If the PAN number is valid, it should be stored in the appropriate field.  
 - If the PAN number is invalid, an error message should be displayed, and the user should be prompted to re-enter the PAN number.  
3. If the "Document ID" is not provided, an error message should be displayed prompting the user to enter the "Document ID".  
4. The system should ensure that the "Document ID" field is editable or non-editable based on the value of the "CHK\_EDIT\_AML" control.  
  
Definition of Done:  
- The "Document ID" field validation and processing logic is implemented as per the acceptance criteria.  
- The system navigates to the appropriate sections and executes related actions based on the "Document ID" value.  
- Error messages are displayed correctly for invalid PAN numbers and missing "Document ID".  
- The "Document ID" field's editability is controlled based on the "CHK\_EDIT\_AML" control value.  
- All functionalities are tested and verified to ensure they work as expected.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- No specific DB queries are mentioned in the provided XML content that can be executed directly in the database without modification.

# Dynamic Proof Type Selection for AML Details

Type: AML

Detailed description: As a user, I want to be able to select a proof type from a list of values (LOV) when entering AML (Anti-Money Laundering) details, so that I can ensure the correct proof type is associated with the AML record.  
  
Acceptance criteria:  
1. When the user double-clicks on the proof type field, the system should dynamically populate the LOV based on the document type and partner type.  
2. If the EIA (Electronic Identification Authentication) exists, the LOV should be populated with values from the system constants table where the system type is 'EIA' and the system code is 'ADDR\_PROOF'.  
3. If the document type is not null, the LOV should be populated with distinct proof types and descriptions from the AML proof master table where the partner type and document type match the current values.  
4. The system should validate the selected proof type against the populated LOV.  
5. Upon selecting a proof type, the system should clear the expiry date and other fields.  
6. The system should automatically populate additional details based on the selected proof type, such as proof description, proof ID, and expiry date, from the relevant tables.  
  
Definition of Done:  
- The proof type field should display a list of values when double-clicked.  
- The LOV should be populated dynamically based on the conditions specified.  
- The system should validate the selected proof type against the LOV.  
- Additional details should be automatically populated based on the selected proof type.  
- The expiry date and other fields should be cleared upon selecting a proof type.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Populate LOV with proof types and descriptions:  
 ```sql  
 SELECT DISTINCT proof\_type, proof\_desc   
 FROM azbj\_aml\_proof\_master   
 WHERE partner\_type = :control.cn\_partner\_type   
 AND document\_type = :AML.document\_type;  
 ```  
  
- Populate LOV with system constants for EIA:  
 ```sql  
 SELECT char\_value, sys\_desc   
 FROM azbj\_system\_constants   
 WHERE sys\_type = 'EIA'   
 AND sys\_code = 'ADDR\_PROOF';  
 ```  
  
- Auto-populate additional details based on proof type:  
 ```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 = :correct\_value.appln\_no   
 AND proof\_type = 'DRIVING LICENSE'   
 AND top\_indicator = 'Y'   
 AND partner\_type = CASE   
 WHEN :azbj\_ul\_checker.chk\_insured = '0' THEN 'IP'   
 ELSE 'PH'   
 END;  
 ```

# Implement Signature Capture and Display Area

Type: IC\_SUBCODE\_SIGNATURE

Title: Implement Signature Capture and Display Area  
  
Acceptance Criteria:  
1. The signature area should be clearly visible and have a white background.  
2. The signature area should have horizontal and vertical scrollbars to allow for navigation within the signature space.  
3. The signature area should be of sufficient size to capture detailed signatures (Width: 536, Height: 143).  
4. There should be a radio button labeled "NA" that is not visible by default.  
5. The signature area should be displayed within a specific canvas and window, ensuring it is properly contained and viewable.  
  
Definition of Done:  
1. The signature area is implemented and can display signatures accurately.  
2. The scrollbars within the signature area function correctly, allowing users to navigate the entire signature space.  
3. The radio button labeled "NA" is present in the code but not visible to the user by default.  
4. The signature area is contained within the specified canvas and window, ensuring proper display and functionality.  
5. All acceptance criteria are met, and the feature is tested and verified for accuracy and usability.

# Manage Reasons for Proposals

Type: REASON

Detailed description: As a user, I want to manage reasons for proposals within a dedicated section, so that I can view, add, and update reasons efficiently.  
  
Acceptance criteria:  
1. The section should display up to 9 records at a time.  
2. The section should include the following fields:  
 - A display-only counter field.  
 - A text field for the reason, which should be read-only and have a maximum length of 200 characters.  
 - A checkbox to mark the reason as checked or unchecked, with default values set to 'N' (unchecked) and 'Y' (checked).  
3. The text field for the reason should be visually distinct with a gray background and bold font.  
4. The checkbox should also have a gray background and bold font.  
5. The section should allow for scrolling to view additional records if more than 9 are present.  
6. The section should support adding new reasons and updating existing ones.  
  
Definition of Done:  
- The section is implemented and displays up to 9 records at a time.  
- The counter field is display-only.  
- The reason text field is read-only, with a maximum length of 200 characters, and styled as specified.  
- The checkbox is functional with default values set to 'N' and 'Y', and styled as specified.  
- Scrolling functionality is implemented to view additional records.  
- The section supports adding new reasons and updating existing ones.  
  
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 Insurance Details

Type: AZBJ\_MLQ\_INSURENCE

Title: Manage Insurance Details  
  
Acceptance Criteria:  
1. The system should allow the user to select a relationship from a predefined list.  
2. The system should provide fields to input the company name, sum assured, and premium.  
3. The company name field should accept a maximum of 100 characters.  
4. The sum assured and premium fields should accept numerical values only.  
5. When a user clicks on any item that is not a button, the system should focus on the clicked item.  
  
Definition of Done:  
1. The user can successfully select a relationship from a list.  
2. The user can input and save the company name, sum assured, and premium.  
3. The system validates the input data according to the specified criteria.  
4. The system correctly handles user interactions, focusing on the clicked item if it is not a button.  
5. 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 CRUD operations or SQL queries.

# Automatic Navigation Based on Field Completion

Type: AZBJ\_MLQ\_INSURENCE

Title: Automatic Navigation Based on Field Completion  
  
Acceptance Criteria:  
1. If the fields "Premium", "Sum Assured", and "Company Name" are all filled out, the system should automatically navigate to the next record.  
2. If any of the fields "Premium", "Sum Assured", or "Company Name" are not filled out, the system should navigate to a specific item labeled "Correct" in the questionnaire section.  
  
Definition of Done:  
- The system correctly identifies whether the "Premium", "Sum Assured", and "Company Name" fields are filled out.  
- The system navigates to the next record if all fields are filled.  
- The system navigates to the "Correct" item in the questionnaire section if any of the fields are not filled.  
- The functionality is tested and verified 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 any direct database queries.

# Implement Relationship Type Dropdown

Type: AZBJ\_MLQ\_INSURENCE

Title: Implement Relationship Type Dropdown  
  
Acceptance Criteria:  
1. The relationship type field should be a dropdown list.  
2. The dropdown list should contain the following options:  
 - 5  
 - 8  
 - LA  
 - No  
3. The dropdown list should be positioned appropriately on the form.  
4. The form should display the relationship type field with a label "Relation" above it.  
5. The form should be visually consistent with the rest of the application, using the specified font and color settings.  
  
Definition of Done:  
1. The relationship type dropdown list is implemented and functional.  
2. The dropdown list contains the specified options.  
3. The dropdown list is correctly positioned on the form.  
4. The label "Relation" is displayed above the dropdown list.  
5. The form's appearance matches the specified design requirements.  
6. 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 any specific database queries.

# Add Exit Button to Policy Details Others Section

Type: POLICY\_DETAILS\_OTHERS

Title: Add Exit Button to Policy Details Others Section  
  
Acceptance Criteria:  
1. The button should be labeled "Exit".  
2. The button should be positioned at the specified coordinates on the form.  
3. The button should have a specific font style, size, and color.  
4. The button should be placed on the "Policy Details Others" section of the form.  
  
Definition of Done:  
- The "Exit" button is visible on the "Policy Details Others" section.  
- The button meets the specified design requirements (label, position, font style, size, and color).  
- The button functions correctly, allowing the user to exit the form.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as there are no CRUD operations mentioned in the provided XML content.

# Implement Exit Button Functionality

Type: POLICY\_DETAILS\_OTHERS

Title: Implement Exit Button Functionality  
  
Acceptance Criteria:  
1. If the global view image count is zero, the current window is 'IP', and the policy reference is not null:  
 - The system should prompt the user to select "VIEW IMAGES" at least once.  
 - The system should navigate to the "VIEW IMAGES" item in the policy details section.  
2. If the insured check is '1', the global view image count for 'PH' is zero, the current window is 'PH', and the policy reference is not null:  
 - The system should prompt the user to select "VIEW IMAGES" at least once.  
 - The system should navigate to the "VIEW IMAGES" item in the policy details section.  
3. The system should hide the current view and window.  
4. If the current window is 'PH':  
 - The system should navigate to the policy details in the 'CP\_SEARCH\_RESULT\_PH' section.  
5. If the current window is not 'PH':  
 - The system should navigate to the policy details in the 'CP\_SEARCH\_RESULT' section.  
6. The system should reset the current window indicator to 'N'.  
  
Definition of Done:  
- The "Exit" button functionality is implemented as per the acceptance criteria.  
- The system performs the necessary checks and navigations based on the context and data.  
- The user is appropriately prompted and navigated based on the conditions.  
- The current view and window are hidden as required.  
- The current window indicator is reset to 'N'.  
- All changes are tested and verified to ensure they work as expected.

# View and Manage Search Results for Customer Proposals

Type: CP\_SEARCH\_RESULT

Title: View and Manage Search Results for Customer Proposals  
  
Acceptance Criteria:  
1. The search results should display the following fields:  
 - Partner ID  
 - First Name  
 - Middle Name  
 - Last Name  
 - Father's Name  
 - Spouse's Name  
 - Gender  
 - Date of Birth  
 - Permanent Address  
 - Telephone  
 - Mobile  
 - Confidence Parameter  
 - Number of Policies  
 - Occupation  
 - Aadhaar Number  
 - PAN Number  
2. Each field should be non-editable to ensure data integrity.  
3. A checkbox should be available to select individual records.  
4. A button labeled "Pol Dtls" should be available to view policy details for the selected record.  
5. The search results should be displayed in a scrollable format with a vertical scrollbar.  
6. The user should be able to click on any field to navigate to that specific item.  
  
Definition of Done:  
- The search results display all the specified fields.  
- All fields are non-editable.  
- The checkbox and "Pol Dtls" button are functional.  
- The search results are scrollable with a vertical scrollbar.  
- Clicking on any field navigates to that specific item.  
  
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.

# Checkbox Selection and Partner Information Update

Type: CP\_SEARCH\_RESULT

Title: Checkbox Selection and Partner Information Update  
  
Acceptance Criteria:  
1. When the checkbox is checked:  
 - If the partner ID matches any of the predefined partner IDs (CP\_1\_PARTID, CP\_2\_PARTID, CP\_3\_PARTID, CP\_4\_PARTID, CP\_5\_PARTID) and the corresponding check flag (CHK\_CP\_1, CHK\_CP\_2, CHK\_CP\_3, CHK\_CP\_4, CHK\_CP\_5) is 'N', the flag should be updated to 'Y'.  
 - The system should update the partner's first name, middle name, last name, father's name, spouse's name, and occupation with the correct values.  
 - The system should set the update name, father name, husband name, and occupation name flags to '1'.  
  
2. When the checkbox is unchecked:  
 - If the partner ID matches any of the predefined partner IDs and the corresponding check flag is 'Y', the flag should be updated to 'N'.  
 - The system should revert the partner's first name, middle name, last name, father's name, spouse's name, and occupation to their backup values.  
 - The system should set the update name, father name, husband name, and occupation name flags to '0'.  
  
3. The system should count the number of checked checkboxes with non-null partner IDs and update the merge status (CHOOSE\_MERGE) as follows:  
 - 'N\_C' if no checkboxes are checked.  
 - 'S\_C' if exactly one checkbox is checked.  
 - 'M\_C' if more than one checkbox is checked.  
  
4. The system should update the signature check, age proof check, and address proof check flags based on the number of checked checkboxes:  
 - 'Y' if one or more checkboxes are checked.  
 - 'NA' if no checkboxes are checked.  
  
Definition of Done:  
- The checkbox functionality is implemented and tested.  
- The system correctly updates the partner information and merge status based on the checkbox state.  
- The system correctly counts the number of checked checkboxes and updates the merge status and proof check flags accordingly.  
- All acceptance criteria are met and verified through testing.

# View Policy Details for Selected Partner

Type: CP\_SEARCH\_RESULT

Title: View Policy Details for Selected Partner  
  
Acceptance Criteria:  
1. When the "Policy Details" button is pressed, the system should navigate to the policy details section and clear any existing data.  
2. The system should retrieve and display the following information for the selected partner:  
 - Partner ID  
 - Policy Reference  
 - Contract Status  
 - Role Type  
 - Sum Assured  
 - Start Date  
 - Risk Date  
 - Product ID  
 - Rated Up status (YES/NO)  
3. The system should determine the rated up status based on the values of ml\_perc, oc\_perc, nri\_perc, and sr\_perc fields.  
4. If any of the fields (ml\_perc, oc\_perc, nri\_perc, sr\_perc) have values greater than 0, the system should concatenate the reasons and values into the "Reason" and "Value" fields respectively.  
5. The system should handle any errors gracefully and display an appropriate error message.  
  
Definition of Done:  
- The "Policy Details" button correctly navigates to the policy details section and clears existing data.  
- The system retrieves and displays the required policy information for the selected partner.  
- The rated up status and reasons are correctly calculated and displayed.  
- Error handling is implemented, and appropriate error messages are shown when necessary.  
- The functionality is tested and verified to work as expected.  
  
DB queries for Table reference CRUD operations only (With Usage):  
```sql  
SELECT partner\_id, azbj\_pk0\_acc.get\_policy\_ref(a.contract\_id) policy\_ref, c.contract\_status status, d.role\_type role\_type, e.sum\_insured\_whole\_cover sum\_assured, f.term\_start\_date start date, h.DATE\_OF\_RECEIPT risk\_date, c.product\_id Product, CASE WHEN g.ml\_perc>0 OR g.oc\_perc>0 OR g.nri\_perc>0 OR g.sr\_perc>0 THEN 'YES' ELSE 'NO' END rated\_up, g.ml\_perc, g.oc\_perc, g.nri\_perc, g.sr\_perc  
FROM ocp\_interested\_parties a, cp\_partners b, ocp\_policy\_versions c, ocp\_ip\_links d, ocp\_policy\_covers e, ocp\_policy\_bases f, azbj\_policy\_covers\_ext g, azbj\_policy\_contract\_ext h  
WHERE a.partner\_id=:CP\_SEARCH\_RESULT.partner\_id  
AND a.partner\_id=b.part\_id  
AND a.contract\_id=c.contract\_id  
AND d.contract\_id=c.contract\_id  
AND a.contract\_id=c.contract\_id  
AND a.ip\_no=d.ip\_no  
AND a.top\_indicator='Y'  
AND a.action\_code <> 'D'  
AND d.top\_indicator='Y'  
AND d.action\_code <> 'D'  
AND c.top\_indicator='Y'  
AND e.contract\_id=c.contract\_id  
AND e.contract\_id=d.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 f.contract\_id=d.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 g.contract\_id=c.contract\_id  
AND g.contract\_id=d.contract\_id  
AND a.contract\_id=g.contract\_id  
AND g.contract\_id=f.contract\_id  
AND g.action\_code <> 'D'  
AND g.top\_indicator='Y'  
AND g.cover\_code like 'L%'  
AND h.contract\_id=c.contract\_id  
AND h.contract\_id=d.contract\_id  
AND a.contract\_id=h.contract\_id  
AND h.contract\_id=f.contract\_id  
UNION  
SELECT partner\_id, azbj\_pk0\_acc.get\_policy\_ref(a.contract\_id) policy\_ref, c.contract\_status status, d.role\_type role\_type, e.sum\_insured\_whole\_cover sum\_assured, f.term\_start\_date start date, h.date\_of\_receipt risk date, c.product\_id product, CASE WHEN g.ml\_perc > 0 OR g.oc\_perc > 0 OR g.nri\_perc > 0 OR g.sr\_perc > 0 THEN 'YES' ELSE 'NO' END rated up, g.ml\_perc, g.oc\_perc, g.nri\_perc, g.sr\_perc  
FROM wip\_interested\_parties a, cp\_partners b, wip\_policy\_versions c, wip\_ip\_links d, wip\_policy\_covers e, wip\_policy\_bases f, wip\_azbj\_policy\_covers\_ext g, wip\_azbj\_policy\_contract\_ext h  
WHERE a.partner\_id = :cp\_search\_result.partner\_id  
AND a.partner\_id = b.part\_id  
AND a.contract\_id = c.contract\_id  
AND d.contract\_id = c.contract\_id  
AND a.contract\_id = c.contract\_id  
AND a.ip\_no = d.ip\_no  
AND e.contract\_id = c.contract\_id  
AND e.contract\_id = d.contract\_id  
AND a.contract\_id = e.contract\_id  
AND e.cover\_code LIKE 'L%'  
AND f.contract\_id = c.contract\_id  
AND f.contract\_id = d.contract\_id  
AND a.contract\_id = f.contract\_id  
AND e.contract\_id = f.contract\_id  
AND g.contract\_id = c.contract\_id  
AND g.contract\_id = d.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 h.contract\_id = d.contract\_id  
AND a.contract\_id = h.contract\_id  
AND h.contract\_id = f.contract\_id;  
```

# Display Current Applicant's Personal Information

Type: CP\_CURRENT\_APPLN

Title: Display Current Applicant's Personal Information  
  
Acceptance Criteria:  
1. The system should display the following fields for the current applicant:  
 - First Name  
 - Middle Name  
 - Last Name  
 - Occupation  
 - Father's Name  
 - Spouse's Name  
 - Gender  
 - Date of Birth (formatted as DD/MM/YYYY)  
 - Place of Birth  
 - Permanent Address  
 - Telephone Number (if available)  
 - Mobile Number (if available)  
 - Number of Policies  
  
2. All fields should be non-editable and displayed in a read-only format.  
  
3. The fields should be visually aligned and properly labeled for clarity.  
  
4. The date of birth should be displayed in the format DD/MM/YYYY.  
  
5. The telephone number, mobile number, and number of policies fields should be hidden if they are not available.  
  
Definition of Done:  
- The user can view all the specified fields for the current applicant.  
- All fields are displayed in a read-only format.  
- The date of birth is correctly formatted.  
- The telephone number, mobile number, and number of policies fields are hidden if not available.  
- The user interface is visually aligned and clear.  
  
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.

# Customer Profile Search and Update Form

Type: CP\_SEARCH

Detailed description: As a user, I want to search for customer profiles using various criteria such as first name, last name, date of birth, gender, father name, and spouse name, so that I can efficiently find and manage customer information.  
  
Acceptance criteria:  
1. The search form should allow input for the following fields:  
 - First Name  
 - Last Name  
 - Date of Birth (in DD/MM/YYYY format)  
 - Gender (with options such as Male, Female)  
 - Father Name  
 - Spouse Name  
2. The form should include buttons for:  
 - Manual Search  
 - Clear  
 - Apply and Save  
3. The form should display the number of records found.  
4. The form should allow the user to update specific fields in the master customer profile, such as:  
 - Name  
 - Occupation  
 - Father Name  
 - Husband Name  
5. The form should include checkboxes for selecting which fields to update in the master customer profile.  
6. The form should include radio buttons for selecting options related to signature, age proof, and address proof.  
7. The form should display images associated with the customer profile.  
8. The form should handle errors gracefully and provide appropriate error messages.  
  
Definition of Done:  
1. The search form is implemented and allows users to input search criteria.  
2. The form includes all specified buttons and checkboxes.  
3. The form displays the number of records found based on the search criteria.  
4. The form allows users to update specific fields in the master customer profile.  
5. The form includes radio buttons for additional options.  
6. The form displays customer profile images.  
7. The form handles errors gracefully and provides appropriate error messages.  
8. The form 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.

# Clear Button Functionality in Search Interface

Type: CP\_SEARCH

Title: Clear Button Functionality in Search Interface  
  
Acceptance Criteria:  
1. When the clear button is pressed, the system should navigate to the search results section.  
2. The system should clear all the data in the search results section without validating the current data.  
  
Definition of Done:  
- The clear button is visible and accessible on the search interface.  
- Pressing the clear button successfully navigates to the search results section.  
- All data in the search results section is cleared without any validation 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.

# Gender Dropdown List Implementation

Type: CP\_SEARCH

Title: Gender Dropdown List Implementation  
  
Acceptance Criteria:  
1. The gender field should be a dropdown list.  
2. The dropdown list should contain the following options:  
 - IP  
 - 5  
 - 8  
 - 10  
 - SL  
3. The gender field should be located at the specified position on the form.  
4. The gender field should be styled with the specified font, size, and color attributes.  
  
Definition of Done:  
1. The gender dropdown list is implemented and displays the correct options.  
2. The dropdown list is positioned correctly on the form.  
3. The styling of the gender field matches the specified attributes.  
4. The functionality is tested and verified to ensure it meets the acceptance criteria.

# Update Occupation Information Based on Checkbox State

Type: CP\_SEARCH

Detailed description: As a user, I want to update the occupation information in the master CP based on the selected occupation type, so that the correct occupation details are reflected in the search results.  
  
Acceptance criteria:  
1. When the "Update Occupation in Master CP" checkbox is checked:  
 - If the partner type is 'IP', the occupation in the search results should be updated to the IP occupation value.  
 - If the partner type is not 'IP', the occupation should remain unchanged.  
2. When the "Update Occupation in Master CP" checkbox is unchecked:  
 - The occupation in the search results should revert to its backup value.  
  
Definition of Done:  
- The checkbox functionality should be implemented and tested.  
- The occupation information should update correctly based on the checkbox state and partner type.  
- The changes should be reflected in the search results 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.

# Update Father's Name Based on Partner Type

Type: CP\_SEARCH

Title: Update Father's Name Based on Partner Type  
  
Acceptance Criteria:  
1. When the checkbox for "Correction in Father Name in Master CP" is checked:  
 - If the partner type is 'IP', the father's name in the search results should be updated with the corrected value for 'IP'.  
 - If the partner type is not 'IP', the father's name should remain unchanged.  
2. When the checkbox is unchecked, the father's name in the search results should revert to its original value.  
  
Definition of Done:  
- The functionality to update the father's name based on the checkbox and partner type is implemented.  
- The father's name is correctly displayed in the search results according to the specified conditions.  
- The feature is tested and verified 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.

# Initialize Search Results Based on Radio Group Selection

Type: CP\_SEARCH

Title: Initialize Search Results Based on Radio Group Selection  
  
Acceptance Criteria:  
1. When the user selects the 'N\_C' option from the radio group, the system should navigate to the search results section.  
2. The system should iterate through all records in the search results and set a specific field (`to\_chk`) to '0' for each record.  
3. The iteration should stop when the last record in the search results is reached.  
  
Definition of Done:  
- The radio group should be displayed correctly in the search module.  
- Selecting the 'N\_C' option should trigger the navigation to the search results section.  
- All records in the search results should have the `to\_chk` field set to '0'.  
- The process should complete without errors and stop at the last record.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Not applicable as the provided logic does not include direct database CRUD operations.

# Update Name in Master CP based on Partner Type

Type: CP\_SEARCH

Detailed description: As a user, I want the system to update the name fields in the master CP based on the selected partner type when I check the "Update Name in Master CP" checkbox, so that the correct name details are displayed.  
  
Acceptance criteria:  
1. When the "Update Name in Master CP" checkbox is checked:  
 - If the partner type is 'IP', the system should update the first name, middle name, and last name fields with the corresponding values from the correct value fields for 'IP'.  
 - If the partner type is not 'IP', the system should not update the name fields.  
2. When the "Update Name in Master CP" checkbox is unchecked:  
 - The system should revert the first name, middle name, and last name fields to their backup values.  
  
Definition of Done:  
- The functionality is implemented and tested.  
- The name fields are correctly updated or reverted based on the checkbox state and partner type.  
- The changes are reviewed and approved.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided logic does not include direct database operations.

# Update Husband Name Based on Checkbox Selection

Type: CP\_SEARCH

Title: Update Husband Name Based on Checkbox Selection  
  
Acceptance Criteria:  
1. When the checkbox labeled "Update Husband Name in Master CP" is checked:  
 - If the partner type is 'IP', the spouse's name should be updated to the value of `IP\_SPOUSE`.  
 - If the partner type is not 'IP', the spouse's name should remain unchanged.  
2. When the checkbox is unchecked, the spouse's name should revert to its backup value.  
  
Definition of Done:  
- The checkbox functionality should be implemented and tested.  
- The spouse's name should update correctly based on the partner type when the checkbox is checked.  
- The spouse's name should revert to its backup value when the checkbox is unchecked.  
- All changes should be reflected in the user interface without any errors.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- No specific DB queries provided in the XML content.

# Validate Checkbox Selections on Signature Radio Button Change

Type: CP\_SEARCH

Title: Validate Checkbox Selections on Signature Radio Button Change  
  
Acceptance Criteria:  
1. If the image count is zero and the partner ID is not null, the system should automatically set the values of the Signature, Age Proof, and Address Proof checkboxes to 'NA'.  
2. The system should prompt the user to check the policy details if the above condition is met.  
3. If any of the checkboxes (Signature, Age Proof, Address Proof) are not selected (i.e., their values are not 'Y'), the system should display an error message prompting the user to select all required checkboxes.  
4. If all checkboxes are selected (i.e., their values are 'Y'), the system should allow the user to proceed without any error messages.  
  
Definition of Done:  
- The system correctly sets the values of the checkboxes to 'NA' when the image count is zero and the partner ID is not null.  
- The system displays a prompt to check the policy details under the specified condition.  
- The system displays an error message if any of the required checkboxes are not selected.  
- The system allows the user to proceed if all required checkboxes are selected.  
- All functionalities are tested and verified to work as expected.

# Apply and Save CP Merging Details for IP Records

Type: CP\_SEARCH

Title: Apply and Save CP Merging Details for IP Records  
  
Acceptance Criteria:  
1. When the "Apply and Save" button is pressed, the system should navigate to the CP Search Result section.  
2. The system should start from the first record and loop through each record in the CP Search Result section.  
3. If a checkbox in the CP Search Result section is checked, the system should set a merge flag to 'Y' and display a success message indicating that the CP merging details are captured successfully for IP.  
4. The loop should exit after processing the last record or upon encountering a checked checkbox.  
  
Definition of Done:  
- The "Apply and Save" button is functional and triggers the described behavior.  
- The system correctly navigates to the CP Search Result section and processes each record as specified.  
- The merge flag is set correctly, and the success message is displayed when applicable.  
- The loop exits appropriately based on the conditions specified.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include specific database queries.

# Manage Checkbox Selection for IP Details or Desktop Data

Type: CP\_SEARCH

Title: Manage Checkbox Selection for IP Details or Desktop Data  
  
Acceptance Criteria:  
1. When the checkbox is deselected (value 'N'):  
 - Display a warning message: "You have De-Selected the CP, So fill all the details of IP OR Choose the Desktop Data."  
 - If a specific partner ID is present, update the record color to indicate deselection.  
2. When the checkbox is selected (value 'Y'):  
 - If a specific partner ID is present, update the record color to indicate selection.  
  
Definition of Done:  
- The system correctly displays the warning message when the checkbox is deselected.  
- The system updates the record color based on the checkbox state and the presence of a partner ID.  
- 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 Additional Proof Checkbox

Type: CP\_SEARCH

Title: Validate Additional Proof Checkbox  
  
Acceptance Criteria:  
1. If the global view image count is zero and the partner ID is not null, the "Additional Proof" checkbox should automatically be set to 'NA'.  
2. The user should be prompted to check the "Policy Details" at least once if the above condition is met.  
3. If the "Signature Check", "Age Proof Check", and "Additional Proof Check" are all set to 'Y', no further action is required.  
4. If any of the above checks are not set to 'Y', an error message should be displayed prompting the user to select all required questions.  
  
Definition of Done:  
- The "Additional Proof" checkbox is automatically set to 'NA' under the specified conditions.  
- The user is prompted to check the "Policy Details" if the global view image count is zero and the partner ID is not null.  
- An error message is displayed if all required checks are not selected.  
- The form logic is thoroughly tested 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.

# Age Proof Check Validation

Type: CP\_SEARCH

Title: Age Proof Check Validation  
  
Acceptance Criteria:  
1. If the global view image count is zero and the partner ID is not null, the age proof check should be set to 'NA'.  
2. The user should be prompted to check the policy details at least once if the above condition is met.  
3. If the signature check, age proof check, and address proof check are all marked as 'Y', no further action is required.  
4. If any of the checks (signature, age proof, or address proof) are not marked as 'Y', the user should receive an error message prompting them to select all questions.  
  
Definition of Done:  
- The age proof check validation logic is implemented and tested.  
- The user receives appropriate prompts and error messages based on the validation conditions.  
- The functionality is verified to ensure it meets the acceptance criteria.  
- The changes are 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 specific CRUD operations or direct database queries.

# Implement Exit Button Functionality

Type: CP\_SEARCH

Detailed description: As a user, I want to be able to clear the search fields and reset the form when I press the "Exit" button, so that I can start a new search or exit the form without retaining any previous data.  
  
Acceptance criteria:  
1. When the "Exit" button is pressed, the following fields should be cleared:  
 - First Name  
 - Last Name  
 - Date of Birth  
 - Gender  
 - Policy Reference  
  
2. The system should navigate to the "Previous Policy" section and iterate through all records to collect policy numbers based on their type (IP or PH). These policy numbers should be concatenated into separate strings for IP and PH types.  
  
3. The system should delete any existing merge requests related to the current application number and partner type from the database.  
  
4. If the user has not selected an option for exiting without merging, an error message should be displayed.  
  
5. Depending on the partner type (IP or PH), specific fields in the "Correct Value" section should be reset to NULL, and certain control flags should be set to 'N'.  
  
6. If there are no records in the search results, the system should delete any merging reasons related to the current application number and partner type from the database.  
  
7. If the partner type is IP and the insured checkbox is checked, the system should update the scrutiny partner form with the provided details and display a popup for PH.  
  
8. If the partner type is IP and the insured checkbox is not checked, the system should validate the RCU and navigate to the appropriate item based on the validation result.  
  
9. If the partner type is PH, the system should update the scrutiny partner form with the provided details and display a popup for PH.  
  
10. If the user has selected the option for exiting without merging, the system should display the "Reason" window and populate it with reasons from the database.  
  
Definition of Done:  
- The "Exit" button functionality is implemented as per the acceptance criteria.  
- All specified fields are cleared and reset appropriately.  
- Database operations for deleting merge requests and merging reasons are executed correctly.  
- Appropriate error messages and popups are displayed based on the conditions.  
- The form navigation and item focus are handled as specified.  
- The feature is tested and verified to ensure it meets the requirements.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- DELETE FROM azbj\_cp\_merge WHERE request\_id IN (SELECT request\_id FROM azbj\_cp\_merge\_request WHERE appln\_no = TO\_NUMBER (:correct\_value.appln\_no) AND partner\_type = :cp\_search.partner\_type);  
- DELETE FROM azbj\_cp\_merge\_request WHERE appln\_no = TO\_NUMBER (:correct\_value.appln\_no) AND partner\_type = :cp\_search.partner\_type;  
- DELETE FROM azbj\_cp\_merging\_reason WHERE application\_no = :correct\_value.appln\_no AND partner\_type = 'IP';  
- DELETE FROM azbj\_cp\_merging\_reason WHERE application\_no = :correct\_value.appln\_no AND partner\_type = 'PH';

# Checkbox CP\_3 Functionality

Type: CP\_SEARCH

Title: Checkbox CP\_3 Functionality  
  
Acceptance Criteria:  
1. When the checkbox "CP\_3" is checked:  
 - If the related identifier (CP\_3\_PARTID) is not null, the system should:  
 - Generate a list number using a predefined function.  
 - Update the color of the matching record based on the generated list number.  
2. When the checkbox "CP\_3" is unchecked:  
 - If the related identifier (CP\_3\_PARTID) is not null, the system should:  
 - Update the color of the matching record to indicate it is not selected.  
  
Definition of Done:  
- The checkbox "CP\_3" should be visible and functional in the user interface.  
- The system should correctly perform the specified actions when the checkbox is checked or unchecked.  
- The related identifier (CP\_3\_PARTID) should be considered in the logic.  
- The user interface should reflect the changes in record color based on the checkbox state.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided logic does not include direct database CRUD operations.

# Checkbox CP\_5 Functionality

Type: CP\_SEARCH

Title: Checkbox CP\_5 Functionality  
  
Acceptance Criteria:  
1. When the checkbox "CP\_5" is checked:  
 - If the related identifier (CP\_5\_PARTID) is not null, the system should:  
 - Generate a number using a specific function.  
 - Update the record color based on the generated number.  
2. When the checkbox "CP\_5" is unchecked:  
 - If the related identifier (CP\_5\_PARTID) is not null, the system should:  
 - Update the record color to indicate it is unchecked.  
  
Definition of Done:  
- The checkbox "CP\_5" should be functional and trigger the appropriate actions when checked or unchecked.  
- The system should handle the presence or absence of the related identifier (CP\_5\_PARTID) correctly.  
- The record color should update based on the checkbox state and the related identifier.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided logic does not include direct database CRUD operations.

# Checkbox CP\_4 Functionality

Type: CP\_SEARCH

Detailed description: As a user, I want to be able to check or uncheck a checkbox labeled "CP\_4" in the CP\_SEARCH section, so that the system can perform specific actions based on the checkbox state.  
  
Acceptance criteria:  
1. When the checkbox "CP\_4" is checked:  
 - If the field `CP\_4\_PARTID` is not null, the system should:  
 - Call a function to get a list number for deduplication.  
 - Change the color of the matching record based on the `CP\_4\_PARTID` and the list number.  
2. When the checkbox "CP\_4" is unchecked:  
 - If the field `CP\_4\_PARTID` is not null, the system should:  
 - Change the color of the matching record based on the `CP\_4\_PARTID` and set the list number to 0.  
  
Definition of Done:  
- The checkbox "CP\_4" should be functional and trigger the appropriate actions when checked or unchecked.  
- The system should handle exceptions gracefully without causing any errors.  
- The user interface should reflect the changes in record color based on the checkbox state and the `CP\_4\_PARTID` 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.

# Checkbox CP\_2 Functionality

Type: CP\_SEARCH

Title: Checkbox CP\_2 Functionality  
  
Acceptance Criteria:  
1. When the checkbox "CP\_2" is checked:  
 - If the field "CP\_2\_PARTID" is not null, the system should call a function to change the color of the matching record to indicate it is selected.  
2. When the checkbox "CP\_2" is unchecked:  
 - If the field "CP\_2\_PARTID" is not null, the system should call a function to change the color of the matching record to indicate it is not selected.  
3. The system should handle any exceptions gracefully without causing errors.  
  
Definition of Done:  
- The checkbox "CP\_2" is present and functional within the CP\_SEARCH section.  
- The system correctly performs the actions of changing the record color based on the checkbox state.  
- The system handles exceptions without causing any disruptions to the user experience.  
- 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 database CRUD operations.

# Save and Apply Button Functionality

Type: CP\_SEARCH

User Story: Save and Apply Button Functionality  
  
Detailed description:   
As a user, I want to be able to save and apply changes to the CP (Customer Profile) data, so that I can ensure the data is correctly updated and any necessary merging of CPs is performed accurately.  
  
Acceptance criteria:  
1. When the "Save and Apply" button is pressed, the system should:  
 - Initialize necessary variables for tracking selected CPs, merge status, and other relevant data.  
 - Navigate to the "Previous Policy" section and iterate through all records to collect previous policy numbers based on the type (IP or PH).  
 - Clear specific fields in the "CP Search" section.  
 - Delete existing merge requests from the database for the given application number and partner type.  
 - If the "Choose Merge" option is selected, validate the partner type and date of birth, and ensure the selected CPs meet the criteria for merging.  
 - If the "Single CP Matched" option is selected, ensure exactly one CP is selected and update the relevant fields in the "Correct Value" section.  
 - If the "Choose for Merging" option is selected, ensure at least two CPs are selected and create a merge request in the database.  
 - Update the "Control" section with the appropriate flags based on the selected options.  
 - Validate the RCU (Risk Control Unit) and navigate to the appropriate item based on the validation result.  
  
Definition of Done:  
- The "Save and Apply" button should perform all the specified actions without errors.  
- The system should correctly handle different scenarios based on the selected options (Single CP Matched, Choose for Merging).  
- All database operations (deletions and insertions) should be executed successfully.  
- The user should receive appropriate messages for any validation errors or successful operations.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Delete from `azbj\_cp\_merge` where `request\_id` matches the criteria.  
- Delete from `azbj\_cp\_merge\_request` where `appln\_no` and `partner\_type` match the criteria.  
- Select count from `AZBJ\_POLICY\_COVERS\_EXT` where conditions are met.  
- Insert into `azbj\_cp\_merging` for creating merge requests.  
- Update `azbj\_phub\_tracker` for proposal status.

# Manual Search for Partner Details

Type: CP\_SEARCH

Detailed description: As a user, I want to perform a manual search for partner details based on specific criteria such as date of birth, gender, first name, and last name, so that I can retrieve and update partner information accurately.  
  
Acceptance criteria:  
1. The system should display an error message if the date of birth is not provided.  
2. The system should display an error message if the gender is not provided.  
3. If the partner type is not specified, it should default to 'IP'.  
4. The system should display an error message if both the first name and last name are not provided.  
5. The system should retrieve the scrutiny number based on the application number.  
6. The system should update the scrutiny partner details with the provided information.  
7. If the partner type is 'PH', specific control fields should be set to 'N' and the partner details should be populated accordingly.  
8. If the partner type is 'IP', specific control fields should be set to 'N' and the partner details should be populated accordingly.  
9. The system should navigate to the search result block and clear any existing data.  
10. The system should perform a search for the partner based on the application number and display the results.  
11. If the merge flag or ixsight flag is set to 'Y', specific fields in the search result should be disabled.  
12. The system should count the number of policies and set the visual attributes of the search result fields based on the color code.  
13. The system should display a message if no data is found for the given search criteria.  
  
Definition of Done:  
- The manual search functionality is implemented and tested.  
- All acceptance criteria are met.  
- The system displays appropriate error messages and updates partner details as specified.  
- The search results are displayed correctly based on the search criteria.  
- The visual attributes of the search result fields are set based on the color code.  
- The functionality is reviewed and approved by the stakeholders.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Retrieve scrutiny number:  
 ```sql  
 SELECT scrutiny\_no  
 INTO v\_scrutiny\_no  
 FROM azbj\_phub\_scrutiny\_prop  
 WHERE application\_no = :correct\_value.appln\_no;  
 ```  
  
- Update scrutiny partner details:  
 ```sql  
 azbj\_pk\_scrutiny\_partner.update\_scrutiny\_partner\_form(  
 p\_verification\_no => :correct\_value.appln\_no,  
 p\_scrutiny\_no => v\_scrutiny\_no,  
 p\_first\_name => :cp\_search.first\_name,  
 p\_last\_name => :cp\_search.last\_name,  
 p\_date\_of\_birth => :cp\_search.date\_of\_birth,  
 p\_sex => :cp\_search.gender,  
 p\_address\_line4 => NULL,  
 p\_module => 'MANUAL',  
 p\_partner\_type => NVL(:cp\_search.partner\_type, 'IP'),  
 p\_policy\_ref => :cp\_search.POLICY\_REF,  
 p\_message => v\_message,  
 p\_father\_name => :cp\_search.FATHER\_NAME,  
 p\_spouse\_name => :cp\_search.SPOUSE\_NAME  
 );  
 ```  
  
- Count merge partners:  
 ```sql  
 SELECT COUNT(1)  
 INTO v\_merge\_partner  
 FROM azbj\_cp\_detail\_gpa\_tab  
 WHERE appln\_no = :correct\_value.appln\_no AND PART\_ID = cp\_record(i).partner\_id;  
 ```  
  
- Retrieve occupation description:  
 ```sql  
 SELECT occ\_desc  
 INTO :cp\_search\_result.OCCUPATION  
 FROM azbj\_occupation  
 WHERE occ\_code = cp\_record(i).OCCUPATION;  
 ```  
  
- Retrieve PAN and AADHAAR numbers:  
 ```sql  
 SELECT a.tax\_id, b.aadhar\_no  
 INTO :cp\_search\_result.PAN\_NO, :cp\_search\_result.AADHAAR\_NO  
 FROM cp\_partners a, azbj\_partner\_extn b  
 WHERE a.part\_id = b.part\_id AND a.part\_id = cp\_record(i).partner\_id;  
 ```

# Manage and Validate Quality Check Questions

Type: AZBJ\_QC\_QUESTIONS

Title: Manage and Validate Quality Check Questions  
  
Acceptance Criteria:  
1. The system should display a list of quality check questions with their descriptions and corresponding answer fields.  
2. The answer field should only accept values 'Y', 'N', or 'NA'. If an invalid value is entered, an error message should be displayed.  
3. The system should allow the user to populate questions based on specific criteria such as partner ID and application number.  
4. The system should handle the deletion of records based on specific conditions, such as the question ID and sub-question values.  
5. The system should ensure that certain questions are excluded based on the product code.  
  
Definition of Done:  
1. The user can view and interact with the quality check questions and their answers.  
2. The system validates the answers and displays appropriate error messages for invalid entries.  
3. The user can populate questions dynamically based on the provided criteria.  
4. The system correctly handles the deletion of records based on predefined conditions.  
5. The system excludes specific questions based on the product code.  
  
DB queries for Table reference CRUD operations only (With Usage):  
```sql  
-- Example query to fetch questions based on partner ID and application number  
SELECT question\_id, sub\_question, answer\_type, MAX(ans1) AS ans1, MAX(ans2) AS ans2, question\_desc, form\_question\_no, appln\_no  
FROM (  
 SELECT b.question\_id, b.sub\_question, a.answer\_type, b.answer AS ans1, NULL AS ans2, a.question\_desc, a.form\_question\_no, appln\_no  
 FROM azbj\_bbu\_questions a, azbj\_fcf\_questionnaire b  
 WHERE NVL(a.form\_question\_no, '0') = NVL(b.question\_no, '0')  
 AND a.question\_id = b.question\_id  
 AND a.sub\_question = b.sub\_question  
 AND b.de\_flag = 'D1'  
 AND partner\_no = :p\_partner\_no  
 AND member\_no = :p\_partner\_no  
 AND b.appln\_no = :correct\_value\_appln\_no  
 UNION ALL  
 SELECT b.question\_id, b.sub\_question, a.answer\_type, NULL AS ans1, b.answer AS ans2, a.question\_desc, a.form\_question\_no, appln\_no  
 FROM azbj\_bbu\_questions a, azbj\_fcf\_questionnaire b  
 WHERE NVL(a.form\_question\_no, '0') = NVL(b.question\_no, '0')  
 AND a.question\_id = b.question\_id  
 AND a.sub\_question = b.sub\_question  
 AND b.de\_flag = 'D2'  
 AND partner\_no = :p\_partner\_no  
 AND member\_no = :p\_partner\_no  
 AND b.appln\_no = :correct\_value\_appln\_no  
)  
GROUP BY question\_id, sub\_question, answer\_type, question\_desc, form\_question\_no, appln\_no  
ORDER BY question\_id, sub\_question;  
```

# Populate Questions Based on Criteria

Type: AZBJ\_QC\_QUESTIONS

Title: Populate Questions Based on Criteria  
  
Acceptance Criteria:  
1. If the product group flag is set to 'Y', the system should call a specific procedure to populate group-related questions.  
2. If the product group flag is not set to 'Y', the system should:  
 - Populate questions based on various criteria such as gender, age, occupation, income proof, nationality, marital status, etc.  
 - Display a warning message if no questions are found for the application.  
 - Clear the current block and start from the first record.  
 - Loop through the list of questions and populate the fields with question details.  
 - Delete specific records based on conditions such as premium amount, question ID, and other criteria.  
 - Display a warning message if an error occurs during the process.  
  
Definition of Done:  
1. The "Populate Questions" button should trigger the population of questions based on the defined criteria.  
2. The system should handle different scenarios and conditions as specified in the acceptance criteria.  
3. Appropriate messages should be displayed to the user in case of errors or specific conditions.  
4. The populated questions should be displayed correctly in the user interface.  
5. The functionality should be tested and verified to ensure it meets the requirements.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- The system should execute a query to fetch the partner type from the `DMT\_AGENTS` and `CP\_PARTNERS` tables based on the reference code and unique code.  
- The system should execute a query to count the number of term products from the `AZBJ\_SYSTEM\_CONSTANTS` table based on the product code.  
- The system should execute a query to fetch the acceptance type from the `azbj\_bi\_appln\_dtls` table based on the application number.  
  
```sql  
-- Fetch partner type  
SELECT b.PARTNER\_TYPE  
INTO v\_partner\_type  
FROM DMT\_AGENTS a, CP\_PARTNERS b  
WHERE a.REFERENCE\_CODE = :CORRECT\_VALUE.FSC\_CODE  
AND a.CUST\_PART\_UNIQUE\_CODE = b.partner\_ref;  
  
-- Count term products  
SELECT COUNT()  
INTO V\_TERM\_PRODUCT\_CNT  
FROM AZBJ\_SYSTEM\_CONSTANTS  
WHERE SYS\_TYPE = 'PROD'  
AND SYS\_CODE = 'TERM\_PRODUCTS'  
AND NUM\_VALUE = :correct\_value.product\_code;  
  
-- Fetch acceptance type  
SELECT ACCEPT\_TYPE INTO v\_BI\_ACCEPT\_TYPE  
FROM azbj\_bi\_appln\_dtls  
WHERE application\_no = TO\_CHAR(:correct\_value.appln\_no)  
AND bi\_accept = 'Y'  
AND top\_indicator = 'Y'  
AND accept\_type = 'BI';  
```

# Validate and Manage Quality Check Answers

Type: AZBJ\_QC\_QUESTIONS

Detailed description: As a user, I want the system to validate the answer provided for a quality check question to ensure it is either 'Y', 'N', or 'NA'. If the answer is 'Y' or 'N', the system should dynamically manage dependent questions and update related fields accordingly.  
  
Acceptance criteria:  
1. The system should display an error message if the answer is not 'Y', 'N', or 'NA'.  
2. If the answer is 'Y' for the question "Do you want to refer this case to Risk Control Unit?", the system should enable the "RCU\_REASONS" field and set a flag indicating the case is referred to the Risk Control Unit.  
3. If the answer is 'N' for the same question, the system should disable the "RCU\_REASONS" field and reset the flag.  
4. For the question "Is Minor life Questionnaire received & complete?", if the answer is 'Y', the system should set a flag indicating the questionnaire is received; if 'N', the flag should be reset.  
5. The system should dynamically create or delete dependent questions based on the answer provided.  
6. If the answer to a specific question is 'N', the system should navigate to a different section and update related fields accordingly.  
7. The system should handle simultaneous proposals and update partner information based on the provided application number.  
  
Definition of Done:  
- The system correctly validates the answer and displays an error message for invalid inputs.  
- The system dynamically manages dependent questions and updates related fields based on the answer.  
- The system navigates to the appropriate sections and updates fields as specified.  
- All acceptance criteria are met, and the functionality is tested and verified.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- The system should use the following query to fetch dependent questions:  
 ```sql  
 SELECT FROM azbj\_qc\_questions WHERE depend\_on = :azbj\_qc\_questions.question\_id AND answer\_depend\_on = :azbj\_qc\_questions.answer;  
 ```  
- The system should use the following query to delete dependent questions:  
 ```sql  
 DELETE FROM azbj\_qc\_questions WHERE depend\_on = :azbj\_qc\_questions.question\_id AND answer\_depend\_on <> :azbj\_qc\_questions.answer;  
 ```  
- The system should use the following query to fetch scrutiny details:  
 ```sql  
 SELECT scrutiny\_no, first\_name, last\_name, date\_of\_birth, sex, address\_line4 FROM azbj\_phub\_scrutiny\_prop WHERE application\_no = :correct\_value.appln\_no;  
 ```

# Handle IT Declaration Process

Type: AZBJ\_QC\_QUESTIONS

Detailed description: As a user, I want to ensure that the IT Declaration process is correctly handled based on various conditions such as contract ID, insured status, nationality, and age proof, so that the system can accurately determine the mandatory status and proceed accordingly.  
  
Acceptance criteria:  
1. When the IT Declaration button is pressed, the system should:  
 - Set a flag indicating the IT rule is active.  
 - Check if the desktop flag is 'N'. If so, fetch the contract ID from the database using the application number and transaction type 'FRP'. If an error occurs, display an error message.  
 - Determine the insured person's number based on the insured status and age.  
 - Loop through the questions in the 'azbj\_qc\_questions' block to check if a specific question (ID 80) is answered 'Y'. If so, set the mandatory flag to 'Y'.  
 - Check the nationality and set the nationality flag accordingly.  
 - Check the age proof and set the passport proof flag accordingly.  
 - If the passport proof flag is 'N', loop through the 'AML' block to check if any proof type is 'PP'. If found, set the passport proof flag to 'Y'.  
 - If the mandatory flag is 'Y', create a parameter list with various parameters (application number, insured person number, contract ID, mandatory status, passport proof, nationality, IP nationality, and property type) and call the 'AZBJ\_INCOMETAX\_QUEST\_DTLS' form.  
 - If the mandatory flag is 'N', display a warning message indicating that the IT Declaration is not received.  
  
Definition of Done:  
- The IT Declaration process should be correctly triggered and handled based on the specified conditions.  
- Appropriate messages should be displayed for errors and warnings.  
- The system should accurately determine and set the mandatory status and other related flags.  
- The 'AZBJ\_INCOMETAX\_QUEST\_DTLS' form should be called with the correct parameters when the mandatory flag is 'Y'.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Fetch contract ID:  
 ```sql  
 SELECT cont\_id  
 INTO v\_contract\_id  
 FROM azbj\_batch\_items  
 WHERE application\_no = :correct\_value.appln\_no  
 AND transaction\_type = 'FRP'  
 AND rownum = 1;  
 ```  
  
- Note: The above query is provided for reference and should be adapted to the specific database environment and application context.

# Retrieve and Display Signature Based on IC Code and Sub IC Code

Type: AZBJ\_QC\_QUESTIONS

Detailed description: As a user, I want to be able to retrieve and display the signature of a specified person based on their IC code and sub IC code, so that I can verify their identity and proceed with the necessary actions.  
  
Acceptance criteria:  
1. When the user clicks the "SP/IC Sign" button, the system should check if the IC code starts with '200%'.  
 - If the IC code starts with '200%' and a signature exists for the specified sub IC code and IC code, the system should:  
 - Retrieve the signature from the database.  
 - Save the signature as a temporary file on the client machine.  
 - Display the signature in the designated area.  
 - If no signature is found, the system should display a warning message indicating that the signature is not available.  
2. If the IC code starts with '1%', the system should:  
 - Check if a signature exists in the external system (BALIC.DDT\_704).  
 - If a signature exists, generate a URL to display the signature and open it in a web browser.  
 - If no signature is found, the system should:  
 - Retrieve the application number for the specified IC code from the recruitment details.  
 - If an application number is found, display the agent's signature and zoom into the image.  
 - If an error occurs during the process, display a warning message.  
3. The system should handle any errors gracefully and display appropriate error messages.  
  
Definition of Done:  
- The "SP/IC Sign" button functionality is implemented and tested.  
- The system retrieves and displays signatures based on the specified conditions.  
- Appropriate warning and 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):  
- Retrieve count of signatures:  
 ```sql  
 SELECT COUNT(1) INTO v\_count   
 FROM AZBJ\_SIGNATURE   
 WHERE sub\_ic\_code = :correct\_value.SPECIFIED\_PERSON\_SUB\_IC\_CODE   
 AND IC\_CODE = :correct\_value.FSC\_CODE;  
 ```  
  
- Retrieve application number:  
 ```sql  
 SELECT APPLICATION\_NO   
 INTO v\_agent\_appln\_no   
 FROM azbj\_ic\_recruitement\_dtls   
 WHERE IC\_CODE = :correct\_value.FSC\_CODE   
 AND modified\_date = (SELECT MAX(modified\_date)   
 FROM azbj\_ic\_recruitement\_dtls   
 WHERE ic\_code = :correct\_value.FSC\_CODE);  
 ```  
  
- Check for signature in external system:  
 ```sql  
 SELECT COUNT()   
 INTO v\_omni\_count   
 FROM BALIC.DDT\_704   
 WHERE FIELD\_1209 = 'SIGNATURE'   
 AND FIELD\_1206 = :correct\_value.FSC\_CODE;  
 ```

# Manage Personal Details and Documents

Type: CP\_SEARCH\_PH

Detailed description: As a user, I want to search and manage personal details and documents for individuals, including their first name, last name, date of birth, gender, and other relevant information, so that I can ensure accurate and up-to-date records.  
  
Acceptance criteria:  
1. The system should allow users to input and update the following personal details:  
 - First Name  
 - Last Name  
 - Date of Birth  
 - Gender  
 - Father Name  
 - Spouse Name  
 - Policy Reference  
2. The system should provide options to check and update the following documents:  
 - Age Proof  
 - Address Proof  
 - Signature  
3. The system should include checkboxes for updating specific details in the master record:  
 - Update Name in Master CP  
 - Update Occupation in Master CP  
 - Correction in Father Name in Master CP  
 - Update Husband Name in Master CP  
4. The system should have buttons for the following actions:  
 - Manual Search  
 - Clear  
 - Apply and Save  
 - Save and Apply  
 - Exit  
5. The system should display the number of records found and the master partner ID.  
6. The system should handle the following triggers:  
 - When a new block instance is created, if the "Update Name in Master CP" checkbox is checked, the system should allow updates to the first and last names.  
 - When a mouse click occurs, if the clicked item is not a button, the system should navigate to the clicked item.  
  
Definition of Done:  
- The user interface allows for the input and update of personal details and documents.  
- Checkboxes and buttons function as described in the acceptance criteria.  
- Triggers are implemented to handle specific actions based on user interactions.  
- The system 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 database queries.

# Clear Button Functionality on Search Page

Type: CP\_SEARCH\_PH

Title: Clear Button Functionality on Search Page  
  
Acceptance Criteria:  
1. When the clear button is pressed, the system should navigate to the search results section.  
2. The system should clear all the data in the search results section without validating the current data.  
  
Definition of Done:  
- The clear button is visible and accessible on the search page.  
- Pressing the clear button navigates to the search results section.  
- All data in the search results section is cleared without any validation 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.

# Document Verification for Policy Details

Type: CP\_SEARCH\_PH

Title: Document Verification for Policy Details  
  
Acceptance Criteria:  
1. If the image count is zero and the partner ID is not null, the system should automatically set the signature, age proof, and address proof checks to 'NA'.  
2. The system should prompt the user to check the policy details at least once if the above condition is met.  
3. If all three checks (signature, age proof, and address proof) are not marked as 'Y', the system should display an error message prompting the user to select all required questions.  
  
Definition of Done:  
- The system correctly sets the document checks to 'NA' when the image count is zero and the partner ID is not null.  
- The user is prompted to check the policy details if the above condition is met.  
- An error message is displayed if all required document checks are not marked as 'Y'.  
- The functionality is tested and verified to ensure it works as expected without any Oracle Forms-specific dependencies.

# Additional Proof Check Validation

Type: CP\_SEARCH\_PH

Title: Additional Proof Check Validation  
  
Acceptance Criteria:  
1. If the global view image count is zero and the partner ID is not null, the "Additional Proof Check" should be set to 'NA'.  
2. The user should be navigated to the "Policy Details" section.  
3. An error message should be displayed prompting the user to check the policy details at least once.  
4. If the "Signature Check", "Age Proof Check", and "Additional Proof Check" are all set to 'Y', no further action is required.  
5. If any of the checks are not set to 'Y', an error message should be displayed prompting the user to select all required questions.  
  
Definition of Done:  
- The "Additional Proof Check" functionality is implemented and tested.  
- The system correctly sets the "Additional Proof Check" to 'NA' and navigates to the "Policy Details" section when conditions are met.  
- Appropriate error messages are displayed based on the conditions.  
- All acceptance criteria are met and verified through testing.

# Gender Dropdown Implementation

Type: CP\_SEARCH\_PH

Title: Gender Dropdown Implementation  
  
Acceptance Criteria:  
1. The gender field should be a dropdown list.  
2. The dropdown list should contain the following options:  
 - "PH"  
 - "5"  
 - "8"  
 - "10"  
 - "No"  
3. The gender field should be positioned at the top left of the form.  
4. The field should be styled with a white background, black text, and bold font.  
5. The field should be restricted to uppercase input.  
  
Definition of Done:  
- The gender dropdown list is implemented and displays the specified options.  
- The field is correctly positioned and styled as per the requirements.  
- The field restricts input to uppercase characters.  
- 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.

# Update Husband Name in Master CP based on Checkbox

Type: CP\_SEARCH\_PH

Title: Update Husband Name in Master CP based on Checkbox  
  
Acceptance Criteria:  
1. When the checkbox is checked:  
 - If the partner type is 'PH', the spouse's name should be updated to the value of `ph\_SPOUSE`.  
 - If the partner type is not 'PH', the spouse's name should remain unchanged.  
2. When the checkbox is unchecked:  
 - The spouse's name should revert to its original value stored in `SPOUSES\_NAME\_bkp`.  
  
Definition of Done:  
- The checkbox functionality is implemented and tested.  
- The spouse's name updates correctly based on the partner type when the checkbox is checked.  
- The spouse's name reverts to its original value when the checkbox is unchecked.  
- 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 CRUD operations or direct database queries.

# Correction in Father's Name Based on Partner Type

Type: CP\_SEARCH\_PH

Title: Correction in Father's Name Based on Partner Type  
  
Acceptance Criteria:  
1. When the checkbox labeled "Correction in Father Name in Master CP" is checked:  
 - If the partner type is 'PH', the father's name in the search result should be updated to the corrected value.  
 - If the partner type is not 'PH', the father's name should remain unchanged.  
2. When the checkbox is unchecked, the father's name in the search result should revert to its original value.  
  
Definition of Done:  
- The checkbox functionality should be implemented and tested to ensure it updates the father's name correctly based on the partner type.  
- The user interface should reflect the changes immediately when the checkbox state is changed.  
- The feature should be tested for both checked and unchecked states to ensure the father's name is updated or reverted correctly.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- No specific DB queries are provided in the XML content for CRUD operations.

# Merge Option Selection in Search Interface

Type: CP\_SEARCH\_PH

Detailed description: As a user, I want to be able to select a merge option in the search interface so that the system can reset specific fields in the search results.  
  
Acceptance criteria:  
1. When the user selects the "N\_C" option in the merge selection, the system should navigate to the search results section.  
2. The system should iterate through all records in the search results and reset the 'to\_chk' field to '0' for each record.  
3. The iteration should stop when the last record in the search results is reached.  
  
Definition of Done:  
- The merge option is selectable by the user.  
- Upon selecting the "N\_C" option, the system correctly navigates to the search results section.  
- All records in the search results have their 'to\_chk' field reset to '0'.  
- The process stops correctly at the last record in the search results.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- No specific database queries are provided in the XML content.

# Update Occupation in Master CP based on Checkbox Selection

Type: CP\_SEARCH\_PH

Title: Update Occupation in Master CP based on Checkbox Selection  
  
Acceptance Criteria:  
1. When the checkbox labeled "Update Occupation in Master CP" is checked:  
 - If the partner type is 'PH', the occupation should be updated to the value of `PH\_OCCUPATION`.  
 - If the partner type is not 'PH', the occupation should remain unchanged.  
2. When the checkbox is unchecked:  
 - The occupation should revert to its backup value.  
  
Definition of Done:  
- The checkbox functionality is implemented and tested.  
- The occupation information updates correctly based on the partner type when the checkbox is checked.  
- The occupation information reverts to its backup value when the checkbox is unchecked.  
- All changes are verified to ensure no unintended side effects on other functionalities.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- No specific DB queries provided in the XML content.

# Update Name in Master CP based on Partner Type

Type: CP\_SEARCH\_PH

Title: Update Name in Master CP based on Partner Type  
  
Acceptance Criteria:  
1. When the "Update Name in Master CP" checkbox is checked:  
 - If the partner type is 'PH', the first name, middle name, and last name fields should be updated with the corresponding 'PH' name values.  
 - If the partner type is not 'PH', the first name, middle name, and last name fields should be updated with the corresponding 'IP' name values.  
2. When the "Update Name in Master CP" checkbox is unchecked:  
 - The first name, middle name, and last name fields should revert to their original values.  
  
Definition of Done:  
- The functionality to update the name fields based on the partner type is implemented.  
- The checkbox correctly triggers the update and revert actions.  
- The changes are tested and verified to ensure they meet the acceptance criteria.  
- The user interface reflects the updated name information accurately.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- No specific DB queries provided in the XML content.

# Age Proof Check Validation

Type: CP\_SEARCH\_PH

Detailed description: As a user, I want to ensure that the age proof check is validated correctly when interacting with the system, so that I can be informed if any required information is missing.  
  
Acceptance criteria:  
1. If the global view image count is zero and the partner ID is not null, the age proof check should be set to 'NA'.  
2. The user should be navigated to the policy details section if the above condition is met.  
3. An error message should be displayed prompting the user to check the policy details at least once.  
4. If the signature check, age proof check, and address proof check are all marked as 'Y', no further action is required.  
5. If any of the checks (signature, age proof, or address proof) are not marked as 'Y', an error message should be displayed prompting the user to select all required questions.  
  
Definition of Done:  
- The age proof check validation logic is implemented and tested.  
- Error messages are displayed as per the acceptance criteria.  
- Navigation to the policy details section works as expected.  
- All checks (signature, age proof, and address proof) are validated correctly.  
- The feature is reviewed and approved by stakeholders.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include any direct database queries.

# Apply and Save CP Merging Details for PH

Type: CP\_SEARCH\_PH

Title: Apply and Save CP Merging Details for PH  
  
Acceptance Criteria:  
1. When the "Apply and Save" button is pressed, the system should navigate to the CP Search Result PH section.  
2. The system should start from the first record and loop through each record.  
3. If a checkbox in the CP Search Result PH section is checked, the system should set a merge flag to 'Y' and display a success message indicating that the CP merging details for PH have been captured successfully.  
4. The loop should exit after processing all records or upon encountering a checked checkbox.  
  
Definition of Done:  
- The "Apply and Save" button functionality is implemented and tested.  
- The system navigates to the CP Search Result PH section upon button press.  
- The system correctly processes each record and sets the merge flag when applicable.  
- A success message is displayed when the merging details are captured.  
- All acceptance criteria are met and verified through testing.

# Save and Apply CP Merging Process

Type: CP\_SEARCH\_PH

Title: Save and Apply CP Merging Process  
  
Acceptance Criteria:  
1. When the "Save and Apply" button is pressed, the system should:  
 - Initialize necessary variables and navigate to the 'Previous Policy' section.  
 - Loop through all records in the 'Previous Policy' section and concatenate policy numbers for policies of type 'PH'.  
 - Clear specific fields in the 'CP Search' section.  
 - Delete existing merge requests related to the current application number and partner type from the database.  
 - Commit the changes quietly.  
 - If the merge option selected is 'Single CP Matched' or 'Choose for Merging', perform additional checks and updates:  
 - If the partner type is 'PH', set the partner's date of birth.  
 - Navigate to the 'CP Search Result' section and loop through all records.  
 - For each selected record, perform various checks and updates, including:  
 - Ensuring the occupation name cannot be changed if there is a previous case with occupational loading.  
 - Validating that the date of birth matches the current proposal.  
 - Updating the 'correct\_value' fields based on the selected CP's details.  
 - If the merge option is 'Single CP Matched', ensure exactly one CP is selected.  
 - If the merge option is 'Choose for Merging', ensure at least two CPs are selected and generate a merge request.  
 - Update control fields based on the selected options.  
  
Definition of Done:  
- The "Save and Apply" button functionality is implemented and tested.  
- All specified validations and updates are performed correctly.  
- The system handles exceptions and displays appropriate error messages.  
- The changes are committed to the database as required.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Delete from `azbj\_cp\_merge` where `request\_id` matches the criteria.  
- Delete from `azbj\_cp\_merge\_request` where `appln\_no` and `partner\_type` match the criteria.  
- Select count from `AZBJ\_POLICY\_COVERS\_EXT` where conditions are met.  
- Select from `ocp\_interested\_parties` and `ocp\_policy\_bases` where conditions are met.

# Manual Search Functionality

Type: CP\_SEARCH\_PH

User Story: Manual Search Functionality  
  
Detailed description:   
As a user, I want to perform a manual search for partner details by entering specific criteria such as date of birth, gender, first name, and last name. The system should validate the input fields and provide appropriate error messages if mandatory fields are missing. Upon successful validation, the system should retrieve and display the relevant partner details from the database.  
  
Acceptance criteria:  
1. The system should display an error message if the date of birth is not provided.  
2. The system should display an error message if the gender is not provided.  
3. The system should display an error message if both the first name and last name are not provided.  
4. The system should retrieve the scrutiny number based on the application number.  
5. The system should update the scrutiny partner form with the provided details.  
6. If the partner type is 'PH', the system should set specific control flags and retrieve confidence partner details.  
7. The system should navigate to the search result block and clear any previous results.  
8. The system should search for partners based on the application number and confidence partner details.  
9. The system should display the number of records found and allow the user to choose a merge option.  
10. The system should highlight the search results based on specific conditions and set visual attributes accordingly.  
11. The system should display additional partner details such as occupation, PAN number, and Aadhaar number.  
12. The system should display a message if no data is found for the given search criteria.  
  
Definition of Done:  
- The manual search functionality is implemented and tested.  
- All acceptance criteria are met.  
- The system displays appropriate error messages for missing mandatory fields.  
- The system retrieves and displays the correct partner details based on the search criteria.  
- The system highlights the search results based on specific conditions.  
- The system displays additional partner details as required.  
- The functionality is reviewed and approved by the stakeholders.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Retrieve scrutiny number:  
 ```sql  
 SELECT scrutiny\_no  
 INTO v\_scrutiny\_no  
 FROM azbj\_phub\_scrutiny\_prop  
 WHERE application\_no = :correct\_value.appln\_no;  
 ```  
  
- Update scrutiny partner form:  
 ```sql  
 azbj\_pk\_scrutiny\_partner.update\_scrutiny\_partner\_form (  
 p\_verification\_no => :correct\_value.appln\_no,  
 p\_scrutiny\_no => v\_scrutiny\_no,  
 p\_first\_name => :cp\_search\_ph.first\_name,  
 p\_last\_name => :cp\_search\_ph.last\_name,  
 p\_date\_of\_birth => :cp\_search\_ph.date\_of\_birth,  
 p\_sex => :cp\_search\_ph.gender,  
 p\_address\_line4 => NULL,  
 p\_module => 'MANUAL',  
 p\_partner\_type => NVL (:cp\_search\_ph.partner\_type, 'IP'),  
 p\_policy\_ref => :cp\_search\_ph.POLICY\_REF,  
 p\_message => v\_message,  
 p\_father\_name => :cp\_search\_ph.FATHER\_NAME,  
 p\_spouse\_name => :cp\_search\_ph.SPOUSE\_NAME  
 );  
 ```  
  
- Retrieve partner details:  
 ```sql  
 SELECT a.tax\_id, b.aadhar\_no  
 INTO :cp\_search\_result\_ph.PAN\_NO, :cp\_search\_result\_ph.AADHAAR\_NO  
 FROM cp\_partners a, azbj\_partner\_extn b  
 WHERE a.part\_id = b.part\_id AND a.part\_id = cp\_record(i).partner\_id;  
 ```  
  
- Retrieve occupation description:  
 ```sql  
 SELECT occ\_desc  
 INTO :cp\_search\_result\_ph.OCCUPATION  
 FROM azbj\_occupation  
 WHERE occ\_code = cp\_record(i).OCCUPATION;  
 ```  
  
- Count merge partners:  
 ```sql  
 SELECT COUNT(1)  
 INTO v\_merge\_partner  
 FROM azbj\_cp\_detail\_gpa\_tab  
 WHERE appln\_no = :correct\_value.appln\_no AND PART\_ID = cp\_record(i).partner\_id;  
 ```  
  
Please review the user story and provide feedback. Once approved, it will be saved.

# Implement Exit Button Functionality

Type: CP\_SEARCH\_PH

Detailed description: As a user, I want to clear all search fields and reset the form when I press the "Exit" button, so that I can start a new search without any previous data.  
  
Acceptance criteria:  
1. When the "Exit" button is pressed, the following fields should be cleared:  
 - First Name  
 - Last Name  
 - Date of Birth  
 - Gender  
 - Policy Reference  
  
2. The system should navigate to the "Previous Policy" section and iterate through all records to collect policy numbers based on their type (IP or PH). These policy numbers should be concatenated into two separate strings: one for IP type and one for PH type.  
  
3. The system should delete any existing merge requests related to the current application number and partner type from the `azbj\_cp\_merge` and `azbj\_cp\_merge\_request` tables.  
  
4. If the user has not selected an option for exiting without merging, an error message should be displayed.  
  
5. Depending on the partner type (IP or PH), the system should reset specific fields and flags related to the partner's information.  
  
6. If there are no records in the search results, the system should delete any merging reasons related to the current application number and partner type from the `azbj\_cp\_merging\_reason` table.  
  
7. If the partner type is IP and the insured checkbox is selected, the system should update the scrutiny partner form with the provided details and show a popup for PH.  
  
8. If the user has selected the "N\_C" option for merging, the system should display the "Reason" window and populate it with reasons from the `azbj\_cp\_reason` table.  
  
Definition of Done:  
- The "Exit" button functionality is implemented as described.  
- All specified fields are cleared when the button is pressed.  
- The system navigates to the "Previous Policy" section and processes records as described.  
- Merge requests and merging reasons are deleted as specified.  
- Error messages are displayed when necessary.  
- The scrutiny partner form is updated and popups are shown as described.  
- The "Reason" window is displayed and populated correctly when the "N\_C" option is selected.  
- All changes are tested and verified to work as expected.  
  
DB queries for Table reference CRUD operations only (With Usage):  
```sql  
-- Delete from azbj\_cp\_merge  
DELETE FROM azbj\_cp\_merge  
WHERE request\_id IN (  
 SELECT request\_id  
 FROM azbj\_cp\_merge\_request  
 WHERE appln\_no = TO\_NUMBER(:correct\_value.appln\_no)  
 AND partner\_type = :cp\_search.partner\_type  
);  
  
-- Delete from azbj\_cp\_merge\_request  
DELETE FROM azbj\_cp\_merge\_request  
WHERE appln\_no = TO\_NUMBER(:correct\_value.appln\_no)  
AND partner\_type = :cp\_search.partner\_type;  
  
-- Delete from azbj\_cp\_merging\_reason for IP  
DELETE FROM azbj\_cp\_merging\_reason  
WHERE application\_no = :correct\_value.appln\_no  
AND partner\_type = 'IP';  
  
-- Delete from azbj\_cp\_merging\_reason for PH  
DELETE FROM azbj\_cp\_merging\_reason  
WHERE application\_no = :correct\_value.appln\_no  
AND partner\_type = 'PH';  
  
-- Select from azbj\_phub\_scrutiny\_prop  
SELECT scrutiny\_no, phfirst\_name, phlast\_name, phdate\_of\_birth, ph\_sex, phaddress\_line4  
INTO v\_ph\_scrutiny\_no, v\_phfirst\_name, v\_phlast\_name, v\_phdate\_of\_birth, v\_phsex, v\_phaddress\_line4  
FROM azbj\_phub\_scrutiny\_prop  
WHERE application\_no = :correct\_value.appln\_no;  
```

# Display Current Applicant's Details in Read-Only Format

Type: CP\_CURRENT\_APPLN\_PH

Title: Display Current Applicant's Details in Read-Only Format  
  
Acceptance Criteria:  
1. The interface should display the following details of the current applicant:  
 - First Name  
 - Middle Name  
 - Last Name  
 - Occupation  
 - Father's Name  
 - Spouse's Name  
 - Gender  
 - Date of Birth (formatted as DD/MM/YYYY)  
 - Place of Birth  
 - Permanent Address  
 - Telephone (if available)  
 - Mobile (if available)  
 - Number of Policies (if available)  
2. All fields should be non-editable to ensure data integrity.  
3. The information should be presented in a clear and organized manner, with appropriate labels for each field.  
4. The interface should be visually consistent with the rest of the application, using the same fonts, colors, and styles.  
  
Definition of Done:  
1. The user interface displays all the required fields in a read-only format.  
2. The data is presented in a clear and organized manner.  
3. The interface is visually consistent with the rest of the application.  
4. The feature has been tested and verified to ensure that all fields are non-editable and display the correct information.  
5. The feature has been reviewed and approved by the stakeholders.

# View and Manage Search Results for Customer Proposals

Type: CP\_SEARCH\_RESULT\_PH

Title: View and Manage Search Results for Customer Proposals  
  
Acceptance Criteria:  
1. The user should be able to view a list of search results with the following details for each record:  
 - Partner ID  
 - First Name  
 - Middle Name  
 - Last Name  
 - Father's Name  
 - Spouse's Name  
 - Gender  
 - Date of Birth  
 - Place of Birth  
 - Permanent Address  
 - Telephone  
 - Mobile  
 - Confidence Parameter  
 - Number of Policies  
 - Occupation  
 - Aadhaar Number  
 - PAN Number  
2. The user should be able to select records using a checkbox.  
3. The user should be able to click a button to view policy details for the selected record.  
4. All fields should be read-only and not allow updates or inserts.  
5. The user should be able to scroll through the list of search results.  
  
Definition of Done:  
1. The user interface displays a list of search results with all the specified details.  
2. The user can select records using a checkbox.  
3. The user can click a button to view policy details for the selected record.  
4. All fields are read-only and do not allow updates or inserts.  
5. The user can scroll through the list of search results.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- No specific DB queries provided in the XML content.

# Checkbox Selection for Updating Personal Information and Merge Options

Type: CP\_SEARCH\_RESULT\_PH

Title: Checkbox Selection for Updating Personal Information and Merge Options  
  
Acceptance Criteria:  
1. When the checkbox is selected:  
 - The fields for first name, middle name, last name, father's name, spouse's name, and occupation should be updated with predefined values.  
 - The merge options should be updated based on the count of selected checkboxes:  
 - If no checkboxes are selected, the merge option should be set to 'N\_C'.  
 - If one checkbox is selected, the merge option should be set to 'S\_C'.  
 - If more than one checkbox is selected, the merge option should be set to 'M\_C'.  
 - The proof checks (signature, age proof, and address proof) should be set to 'Y'.  
  
2. When the checkbox is deselected:  
 - The fields for first name, middle name, last name, father's name, spouse's name, and occupation should revert to their original values.  
 - The merge options should be updated based on the count of selected checkboxes as described above.  
 - The proof checks should be set to 'NA'.  
 - The update flags for name, father's name, spouse's name, and occupation should be set to '0'.  
  
Definition of Done:  
- The checkbox functionality should be implemented and tested to ensure it updates the personal information fields correctly.  
- The system should correctly count the number of selected checkboxes and update the merge options and proof checks accordingly.  
- All acceptance criteria should be met, and the feature should be free of bugs.  
  
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 Detailed Policy Information for Selected Partner

Type: CP\_SEARCH\_RESULT\_PH

Title: View Detailed Policy Information for Selected Partner  
  
Acceptance Criteria:  
1. When the user selects a partner and clicks the "Policy Details" button, the system should:  
 - Navigate to the policy details section.  
 - Clear any existing data in the policy details section.  
 - Retrieve and display the first record of policy details for the selected partner.  
2. The system should fetch policy details from the database, including:  
 - Partner ID  
 - Policy reference  
 - Contract status  
 - Role type  
 - Sum assured  
 - Start date  
 - Risk date  
 - Product ID  
 - Whether the policy is rated up (YES/NO)  
3. If the policy is rated up, the system should determine the rating reasons and values based on the following criteria:  
 - ML percentage  
 - OC percentage  
 - NRI percentage  
 - SR percentage  
4. The system should display the rating reasons and values in the policy details section.  
5. The system should handle any errors during the data retrieval process and display an appropriate error message.  
  
Definition of Done:  
- The user can successfully view detailed policy information for a selected partner.  
- The policy details section is populated with accurate data retrieved from the database.  
- The system correctly identifies and displays rating reasons and values if the policy is rated up.  
- Error handling is implemented, and appropriate error messages are displayed in case of data retrieval issues.  
  
DB queries for Table reference CRUD operations only (With Usage):  
```sql  
SELECT partner\_id, azbj\_pk0\_acc.get\_policy\_ref(a.contract\_id) policy\_ref, c.contract\_status status, d.role\_type role\_type, e.sum\_insured\_whole\_cover sum\_assured, f.term\_start\_date start\_date, h.DATE\_OF\_RECEIPT risk\_date, c.product\_id Product, CASE WHEN g.ml\_perc > 0 OR g.oc\_perc > 0 OR g.nri\_perc > 0 OR g.sr\_perc > 0 THEN 'YES' ELSE 'NO' END rated\_up, g.ml\_perc, g.oc\_perc, g.nri\_perc, g.sr\_perc  
FROM ocp\_interested\_parties a, cp\_partners b, ocp\_policy\_versions c, ocp\_ip\_links d, ocp\_policy\_covers e, ocp\_policy\_bases f, azbj\_policy\_covers\_ext g, azbj\_policy\_contract\_ext h  
WHERE a.partner\_id = :CP\_SEARCH\_RESULT\_PH.partner\_id  
AND a.partner\_id = b.part\_id  
AND a.contract\_id = c.contract\_id  
AND d.contract\_id = c.contract\_id  
AND a.contract\_id = c.contract\_id  
AND a.ip\_no = d.ip\_no  
AND a.top\_indicator = 'Y'  
AND a.action\_code <> 'D'  
AND d.top\_indicator = 'Y'  
AND d.action\_code <> 'D'  
AND c.top\_indicator = 'Y'  
AND e.contract\_id = c.contract\_id  
AND e.contract\_id = d.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 f.contract\_id = d.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 g.contract\_id = c.contract\_id  
AND g.contract\_id = d.contract\_id  
AND a.contract\_id = g.contract\_id  
AND g.contract\_id = f.contract\_id  
AND g.action\_code <> 'D'  
AND g.top\_indicator = 'Y'  
AND g.cover\_code LIKE 'L%'  
AND h.contract\_id = c.contract\_id  
AND h.contract\_id = d.contract\_id  
AND a.contract\_id = h.contract\_id  
AND h.contract\_id = f.contract\_id  
UNION  
SELECT partner\_id, azbj\_pk0\_acc.get\_policy\_ref(a.contract\_id) policy\_ref, c.contract\_status status, d.role\_type role\_type, e.sum\_insured\_whole\_cover sum\_assured, f.term\_start\_date start\_date, h.date\_of\_receipt risk date, c.product\_id product, CASE WHEN g.ml\_perc > 0 OR g.oc\_perc > 0 OR g.nri\_perc > 0 OR g.sr\_perc > 0 THEN 'YES' ELSE 'NO' END rated\_up, g.ml\_perc, g.oc\_perc, g.nri\_perc, g.sr\_perc  
FROM wip\_interested\_parties a, cp\_partners b, wip\_policy\_versions c, wip\_ip\_links d, wip\_policy\_covers e, wip\_policy\_bases f, wip\_azbj\_policy\_covers\_ext g, wip\_azbj\_policy\_contract\_ext h  
WHERE a.partner\_id = :CP\_SEARCH\_RESULT\_PH.partner\_id  
AND a.partner\_id = b.part\_id  
AND a.contract\_id = c.contract\_id  
AND d.contract\_id = c.contract\_id  
AND a.contract\_id = c.contract\_id  
AND a.ip\_no = d.ip\_no  
AND e.contract\_id = c.contract\_id  
AND e.contract\_id = d.contract\_id  
AND a.contract\_id = e.contract\_id  
AND e.cover\_code LIKE 'L%'  
AND f.contract\_id = c.contract\_id  
AND f.contract\_id = d.contract\_id  
AND a.contract\_id = f.contract\_id  
AND e.contract\_id = f.contract\_id  
AND g.contract\_id = c.contract\_id  
AND g.contract\_id = d.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 h.contract\_id = d.contract\_id  
AND a.contract\_id = h.contract\_id  
AND h.contract\_id = f.contract\_id;  
```

# 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 view 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.  
5. The screen layout and design match the specified requirements.  
6. All functionalities are tested and verified to be working correctly.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Not applicable as the provided XML content does not include any specific database queries or CRUD operations.

# Implement Exit Button Functionality

Type: BLK\_SIGN\_CONFID

Detailed description: As a user, I want to be able to exit the "Signature Confidence Details" screen by pressing a button labeled "Exit" so that I can easily navigate away from this screen when needed.  
  
Acceptance criteria:  
- When the "Exit" button is pressed, the system should navigate to the "Correct Value" screen.  
- The "Exit" button should be clearly labeled and easily identifiable.  
- The button should be positioned at coordinates (306, 272) 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 "Correct Value" screen when pressed.  
- 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 there are no direct database operations mentioned in the provided XML content.

# Generate Secure URL for Master Signature Document

Type: BLK\_SIGN\_CONFID

Title: Generate Secure URL for Master Signature Document  
  
Acceptance Criteria:  
1. When the "Master Signature" button is pressed, the system should generate a secure URL using the application number.  
2. The URL should be encrypted and should follow the format provided by the `azbj\_encrypt\_dms\_link` function.  
3. If the URL is successfully generated, it should be opened in a web browser.  
4. If the URL is not generated, no action should be taken.  
  
Definition of Done:  
- The "Master Signature" button is functional and triggers the URL generation process.  
- The URL is encrypted and follows the specified format.  
- The URL opens in a web browser if successfully generated.  
- 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 CRUD operations.

# View Policy Details

Type: POLICY\_DETAILS

Title: View Policy Details  
  
Acceptance Criteria:  
1. The policy details should be displayed in a structured format with the following fields:  
 - Partner ID  
 - Policy Reference  
 - Status  
 - Role  
 - Sum Assured  
 - Start Date (formatted as DD/MM/YYYY)  
 - Risk Date (formatted as DD/MM/YYYY)  
 - Product  
 - Rated Up  
 - Reason  
 - Value  
2. The fields should be read-only and not allow any modifications.  
3. There should be a button labeled "View Images" that allows the user to view associated images for the policy.  
  
Definition of Done:  
1. The policy details are displayed correctly with all the specified fields.  
2. The fields are non-editable and display accurate information.  
3. The "View Images" button is functional and allows the user to view images related to the policy.  
4. The user interface is intuitive and easy to navigate.  
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 specific database queries or operations.

# View Images Functionality

Type: POLICY\_DETAILS

Title: View Images Functionality  
  
Acceptance Criteria:  
1. When the "View Images" button is pressed, the system should increment a global counter for the number of times images have been viewed.  
2. If the insured check is true and the policy type is 'PH', an additional counter specific to 'PH' should be incremented.  
3. The system should generate a URL for the policy documents using an encryption function.  
4. If the URL is successfully generated, it should open the document in a web browser.  
5. If there is an error in generating the URL, an error message should be displayed to the user.  
  
Definition of Done:  
- The "View Images" button is functional and performs the described actions.  
- The counters for viewing images are correctly incremented.  
- The URL for the policy documents is correctly generated and opened in a web browser.  
- Appropriate error handling is in place, and error messages are displayed when necessary.  
  
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.

# Manage Duplicate Contact Information

Type: BLK\_DUPLICATE\_CONTACT\_NO

Title: Manage Duplicate Contact Information  
  
Acceptance Criteria:  
1. The system should display a list of duplicate contacts with the following details:  
 - Policy Number/Agent Code  
 - Name  
 - Contact Information  
2. The system should allow users to mark contacts with a checkbox for special handling.  
3. The system should ensure that the displayed information is read-only to prevent unauthorized modifications.  
4. The system should support scrolling through the list of duplicate contacts if the number of records exceeds the display limit.  
  
Definition of Done:  
1. The user interface displays a list of duplicate contacts with the specified details.  
2. Users can mark contacts for special handling using a checkbox.  
3. The contact information fields are read-only.  
4. The interface includes a scrollbar to navigate through the list of duplicate contacts.  
  
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 allow users to add new proposal records.  
2. The system should allow users to save the current proposal record.  
3. The system should allow users to delete selected proposal records.  
4. The system should provide a dropdown list for the "Proposal Form Field" which is populated with values from a predefined list.  
5. The system should allow users to enter comments for each proposal record.  
6. The system should provide a checkbox to mark records for deletion.  
7. The system should allow users to mark whether details have been received for each proposal record.  
8. The system should provide buttons for saving, adding new rows, rechecking, and navigating back.  
  
Definition of Done:  
1. The user interface should have fields for "Proposal Form Field", "Comments", and "Details Received".  
2. The user interface should have buttons labeled "SAVE", "ADD ROW", "RECHECK", "BACK", and "DELETE ROW".  
3. The "Proposal Form Field" dropdown should be populated with values from the database.  
4. The "Details Received" field should be a dropdown with predefined values.  
5. The "Delete Row" checkbox should allow users to mark records for deletion.  
6. The "SAVE" button should save the current record.  
7. The "ADD ROW" button should add a new empty row for data entry.  
8. The "RECHECK" button should trigger a recheck of the current data.  
9. The "BACK" button should navigate the user to the previous screen.  
10. The "DELETE ROW" button should delete the selected record(s).  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Populate "Proposal Form Field" dropdown:  
 ```sql  
 SELECT screen\_value  
 FROM inf\_dnm\_poplists  
 WHERE poplist\_code = 'PROPOSAL\_FORM\_FIELDS' AND LANGUAGE = 'US'  
 ORDER BY SCREEN\_VALUE ASC;  
 ```

# Delete Selected Rows from Incomplete Proposal Details

Type: AZBJ\_INCMPLT\_PROP\_DTLS

Title: Delete Selected Rows from Incomplete Proposal Details  
  
Acceptance Criteria:  
1. When the delete button is pressed, the system should navigate to the incomplete proposal details section.  
2. The system should iterate through all records in the incomplete proposal details section.  
3. For each record, if the delete checkbox is selected, the system should delete that record.  
4. The process should stop once all records have been checked.  
  
Definition of Done:  
- The delete button is functional and deletes only the selected rows.  
- The system correctly iterates through all records and stops after the last record.  
- The user interface updates to reflect the deletion of the selected rows.  
- 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 logic is specific to Oracle Forms and cannot be executed directly in the database without modification.

# Dropdown for Details Received

Type: AZBJ\_INCMPLT\_PROP\_DTLS

Detailed description: As a user, I want to select whether the details have been received or not from a dropdown list, so that I can accurately record the status of the details received for incomplete proposals.  
  
Acceptance criteria:  
1. The dropdown list should have the following options:  
 - "5"  
 - "8"  
 - "No"  
2. The default value of the dropdown list should be "N".  
3. The dropdown list should be labeled "Details Received" and should be positioned appropriately on the form.  
  
Definition of Done:  
- The dropdown list is implemented and displays the correct options.  
- The default value is set to "N".  
- The label "Details Received" is correctly displayed and aligned.  
- The dropdown list is functional and allows the user to select an option.  
- The form is tested to ensure the dropdown list 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.

# Recheck Incomplete Proposal Details

Type: AZBJ\_INCMPLT\_PROP\_DTLS

Detailed description: As a user, I want to be able to recheck the details of an incomplete proposal so that I can ensure all necessary information is accurate and complete.  
  
Acceptance criteria:  
1. When the "Recheck" button is pressed, the current view should be hidden.  
2. The system should navigate to the "Incomplete Proposal Details" section and clear any existing data.  
3. The system should then navigate to the "Quality Check Questions" section and focus on the "Answer" field.  
4. Two variables, `v\_recheck` and `v\_stp\_flg`, should be set to 'Y' to indicate that a recheck is in progress and to stop further processing if necessary.  
5. Any exceptions during this process should be handled gracefully without causing the application to crash.  
  
Definition of Done:  
- The "Recheck" button functionality is implemented and tested.  
- The system hides the current view and navigates to the appropriate sections as specified.  
- The variables `v\_recheck` and `v\_stp\_flg` are correctly set.  
- Exception handling is in place to ensure the application remains stable.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include any specific database queries.

# Add 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, a confirmation alert should appear asking, "Do you really want to add a new record for Incomplete Proposal Form Details?"  
2. If the user selects "No" on the confirmation alert, the action should be canceled, and no new record should be added.  
3. If the user selects "Yes" on the confirmation alert, the system should navigate to the Incomplete Proposal Form Details section, move to the last record, and create a new record.  
4. If any error occurs during the process, an error message should be displayed indicating "Exception in adding new rows" along with the specific error details.  
  
Definition of Done:  
- The "Add Row" button is functional and triggers the confirmation alert.  
- The confirmation alert correctly handles user input (Yes/No).  
- A new record is successfully created in the Incomplete Proposal Form Details section when the user confirms the action.  
- Appropriate error handling is in place, and error messages are displayed as needed.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Not applicable as the provided XML content does not include any direct database queries.

# Dropdown List for Proposal Form Field

Type: AZBJ\_INCMPLT\_PROP\_DTLS

Detailed description: As a user, I want to view and select options from a dropdown list in the proposal form field so that I can input relevant data for incomplete proposal details.  
  
Acceptance criteria:  
1. The dropdown list should display the following options:  
 - 8  
 - 10  
 - No  
2. The dropdown list should be labeled "Proposal Form Field" and should be positioned appropriately on the form.  
3. The dropdown list should be styled with the specified font and color settings.  
4. The dropdown list should be part of the "Incomplete Proposal Details" section.  
  
Definition of Done:  
- The dropdown list is implemented and displays the correct options.  
- The dropdown list is labeled and positioned as specified.  
- The dropdown list adheres to the specified styling.  
- The dropdown list is integrated into the "Incomplete Proposal Details" section and functions correctly.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Not applicable as the provided XML content does not include specific database queries.

# Save Incomplete Proposal Form Details

Type: AZBJ\_INCMPLT\_PROP\_DTLS

Detailed description: As a user, I want to be able to save incomplete proposal form details by clicking a "Save" button, which will prompt me with a confirmation message to ensure that I really want to save the details. If I confirm, the system should hide the current view and navigate to the next section for quality check questions.  
  
Acceptance criteria:  
1. When the "Save" button is clicked, a confirmation message should appear asking, "Do you really want to Save Incomplete Proposal Form Details?"  
2. If the user selects "No" on the confirmation message, the save operation should be canceled, and the form should remain in its current state.  
3. If the user selects "Yes" on the confirmation message, the system should:  
 - Mark the incomplete proposal details as saved.  
 - Hide the current view.  
 - Navigate to the quality check questions section.  
 - Set specific flags (`v\_recheck` and `v\_stp\_flg`) to 'Y' for further processing.  
  
Definition of Done:  
- The "Save" button triggers a confirmation message.  
- The system correctly handles the user's response to the confirmation message.  
- The current view is hidden, and the system navigates to the quality check questions section upon confirmation.  
- The specific flags are set correctly for further processing.  
  
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.

# Back Button Functionality in Incomplete Proposal Details

Type: AZBJ\_INCMPLT\_PROP\_DTLS

Detailed description: As a user, I want to navigate back from the incomplete proposal details section to the quality check questions section, ensuring that all required fields in the incomplete proposal details are filled out. If any required fields are missing, I should be prompted to complete them. If all required fields are filled, I should be asked whether I want to save the details before proceeding.  
  
Acceptance criteria:  
1. When the "Back" button is pressed, the system should check if the fields `prop\_form\_field\_scr`, `prop\_comments`, and `dtls\_rcvd` are filled.  
2. If any of these fields are empty, the system should display a message prompting the user to complete the incomplete proposal form details.  
3. If all required fields are filled, the system should check the flag `incmp\_prp\_dtls\_sv\_flg`.  
4. If the flag is not set to 'Y', the system should display an alert asking the user if they want to save the details.  
5. If the user chooses not to save, the system should hide the incomplete proposal details section, clear the block, and navigate to the quality check questions section.  
6. If the user chooses to save, the system should navigate to the save button for the incomplete proposal details.  
  
Definition of Done:  
- The "Back" button functionality is implemented as described.  
- The system correctly identifies and prompts for incomplete fields.  
- The system displays the save alert when required.  
- Navigation between sections works as expected based on user input.  
- 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 CRUD operations or SQL queries.

# Manage Solution Product Details

Type: SOLUTION\_PRODUCT

User Story: Manage Solution Product Details  
  
Detailed description:   
As a user, I want to manage the details of solution products within the system, so that I can accurately capture and update various attributes related to the products.  
  
Acceptance criteria:  
1. The system should allow the user to view and manage the following attributes for solution products:  
 - Frequency  
 - Accumulate  
 - Insurance for All  
 - Fully Vaccinated Flag  
 - Auto Pay Rebate Flag  
 - Salaried Rebate Flag  
 - Income Period  
 - Payout Frequency  
 - Death Benefit Option  
 - Cash Bonus  
 - Deferred Period  
 - Worksite Rebate Flag  
 - Life Stage Upgrade Flag  
 - Number of Premium Holidays  
 - Death Benefit Percentage  
 - GMI  
 - Add/Delete Loyalty  
 - Payout Date  
 - ADB Percentage  
 - EMR2 Value  
 - EMR1 Value  
 - EMR2  
 - EMR1  
 - Vesting Age  
 - Annuity Option  
 - Annuity Type  
 - Pension Option  
 - Percentage Increase in SA  
 - SPW Flag  
 - SPW Percentage  
 - SPW Frequency  
 - IB Amount  
 - Product ID  
 - Package Description  
 - Package Code  
 - Premium  
 - Sum Assured  
 - Multiplier  
 - Benefit Term  
 - Premium Term  
 - Money Booster  
 - Nominee Age  
 - Solution ID  
 - Solution Multiplier  
  
2. The system should ensure that certain fields are only viewable and not editable based on specific conditions (e.g., Fully Vaccinated Flag, Auto Pay Rebate Flag, Salaried Rebate Flag, etc.).  
  
3. The system should provide appropriate prompts and tooltips for each attribute to guide the user in entering the correct information.  
  
4. The system should validate the data entered by the user to ensure it meets the required format and constraints (e.g., numeric fields, date fields, etc.).  
  
5. The system should allow the user to insert new records and update existing records for solution products.  
  
Definition of Done:  
- The user can view and manage all specified attributes for solution products.  
- The system enforces view-only restrictions on specified fields.  
- The system provides prompts and tooltips for user guidance.  
- Data validation is implemented for all fields.  
- The user can successfully insert and update records for solution products.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- The system should use the following query to fetch package details:  
 ```sql  
 SELECT FROM azbj\_package\_master WHERE product\_id = :correct\_value.product\_code;  
 ```  
 This query is used to populate the list of available packages for the solution product.

# Enable/Disable Percentage Input Field Based on Control Flag

Type: SOLUTION\_PRODUCT

Title: Enable/Disable Percentage Input Field Based on Control Flag  
  
Acceptance Criteria:  
1. When the control flag is set to 'Y':  
 - The percentage input field should be enabled.  
2. When the control flag is set to 'N':  
 - The percentage input field should be disabled.  
  
Definition of Done:  
- The percentage input field is correctly enabled or disabled based on the control flag status.  
- The changes are reflected immediately when the control flag status is updated.  
- 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.

# GMI Value Validation and Sum Assured Calculation

Type: SOLUTION\_PRODUCT

Title: GMI Value Validation and Sum Assured Calculation  
  
Acceptance Criteria:  
1. When the product code is 299 and the GMI value is null, the system should display a warning message prompting the user to enter a valid GMI value.  
2. If the GMI value is provided, the system should calculate the sum assured by multiplying the GMI value by 144.  
3. The sum assured field should be enabled if it was previously disabled.  
  
Definition of Done:  
- The validation logic for the GMI value is implemented and tested.  
- The warning message is displayed correctly when the GMI value is null.  
- The sum assured is calculated and displayed correctly when the GMI value is provided.  
- The sum assured field is enabled if it was previously disabled.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include any direct database queries.

# Premium Frequency Selection for Solution Product

Type: SOLUTION\_PRODUCT

Title: Premium Frequency Selection for Solution Product  
  
Acceptance Criteria:  
1. The premium frequency field should display a list of predefined options: "5", "8", "10", and "No".  
2. The field should be read-only, meaning users cannot insert or update the values directly.  
3. The field should be displayed on the "COVER\_SOLUTION" tab page.  
4. The prompt for the field should be "Prem. Frequency" and should be right-aligned.  
5. The field should be displayed in uppercase.  
  
Definition of Done:  
1. The premium frequency field is visible and displays the correct predefined options.  
2. The field is read-only and cannot be modified by the user.  
3. The field is correctly positioned on the "COVER\_SOLUTION" tab page with the prompt "Prem. Frequency" right-aligned.  
4. The field values are displayed in uppercase.  
  
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 percentage increase in SA from predefined list

Type: SOLUTION\_PRODUCT

Title: User can select percentage increase in SA from predefined list  
  
Acceptance Criteria:  
1. The user should be able to see a list of predefined percentage increase options when interacting with the field.  
2. The list should include the following options: 5%, 8%, and "No" (which corresponds to "SL").  
3. The field should be located on the "COVER\_SOLUTION" tab page.  
4. The field should be visible and editable by the user.  
5. The field should accept numeric input and restrict the input to uppercase characters.  
  
Definition of Done:  
1. The user can successfully select a percentage increase from the predefined list.  
2. The selected value is saved and can be retrieved for future use.  
3. The field is properly aligned and displayed on the "COVER\_SOLUTION" tab page.  
4. The field adheres to the specified formatting and input restrictions.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include specific database queries.

# Input and Update Premium Value for Solution Product

Type: SOLUTION\_PRODUCT

Title: Input and Update Premium Value for Solution Product  
  
Acceptance Criteria:  
1. The premium field should accept numeric values only.  
2. The premium field should allow both insertion and updating of values.  
3. Upon entering a value in the premium field and pressing the next item key, the system should automatically navigate to the frequency field of the correct value section.  
4. The premium field should be displayed on the "Cover Solution" tab page.  
5. The premium field should have a white background, black text, and be displayed in uppercase.  
6. The premium field should have a prompt labeled "Premium" aligned to the right of the field.  
  
Definition of Done:  
1. The premium field is implemented and visible on the "Cover Solution" tab page.  
2. The premium field accepts numeric values and allows for both insertion and updating.  
3. The system navigates to the frequency field of the correct value section upon pressing the next item key after entering a premium value.  
4. The premium field has the specified visual properties (white background, black text, uppercase).  
5. The premium field prompt is correctly labeled and aligned.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Not applicable as the provided XML content does not include specific database queries.

# Manage SPW Flag and SPW Percentage Field

Type: SOLUTION\_PRODUCT

Title: Manage SPW Flag and SPW Percentage Field  
  
Acceptance Criteria:  
1. When the SPW Flag is set to 'Y', the SPW Percentage field should be enabled.  
2. When the SPW Flag is set to 'N', the SPW Percentage field should be disabled.  
  
Definition of Done:  
1. The SPW Flag can be toggled between 'Y' and 'N'.  
2. The SPW Percentage field's enabled/disabled state changes dynamically based on the SPW Flag's value.  
3. The changes are reflected immediately without requiring a page refresh or additional user action.  
  
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.

# Package Code Selection and Navigation

Type: SOLUTION\_PRODUCT

Title: Package Code Selection and Navigation  
  
Acceptance Criteria:  
1. When the user focuses on the package code field, a list of available packages should be displayed.  
2. The list of available packages should be fetched from the database based on the product code.  
3. Upon selecting a package code, the system should clear any existing data in the related fund details sections.  
4. The system should then navigate to the premium field for further data entry.  
  
Definition of Done:  
1. The package code field displays a list of available packages when focused.  
2. The list of available packages is correctly filtered based on the product code.  
3. Upon selection of a package code, related fund details sections are cleared.  
4. The system navigates to the premium field after the package code is selected.  
5. All functionalities are tested and verified to work as expected.  
  
DB queries for Table reference CRUD operations only(With Usage):  
```sql  
SELECT FROM azbj\_package\_master WHERE product\_id = :correct\_value.product\_code;  
```  
This query is used to fetch the list of available packages based on the product code.

# Implement Annuity Frequency Dropdown for Pension Options

Type: SOLUTION\_PRODUCT

Title: Implement Annuity Frequency Dropdown for Pension Options  
  
Acceptance Criteria:  
1. The annuity frequency field should be a dropdown list with the following options:  
 - "5" with a value of "5"  
 - "8" with a value of "8"  
 - "No" with a value of "SL"  
2. The dropdown list should be disabled by default.  
3. The dropdown list should be displayed on the "COVER\_SOLUTION" tab page.  
4. The dropdown list should have a prompt labeled "Annuity Frequency" aligned to the right.  
5. The dropdown list should be editable, allowing users to insert and update values.  
  
Definition of Done:  
- The dropdown list for annuity frequency is implemented and displays the correct options.  
- The dropdown list is disabled by default.  
- The dropdown list is placed on the "COVER\_SOLUTION" tab page with the prompt "Annuity Frequency" aligned to the right.  
- Users can insert and update values in the dropdown list.  
- The feature is tested and verified to meet the acceptance criteria.

# Dropdown List for Frequency Selection

Type: SOLUTION\_PRODUCT

Title: Dropdown List for Frequency Selection  
  
Acceptance Criteria:  
1. The frequency field should be a dropdown list with the following options:  
 - "5"  
 - "8"  
 - "No"  
2. The dropdown list should be displayed on the "COVER\_SOLUTION" tab page.  
3. The field should be editable, allowing users to select a value from the list.  
4. The selected value should be stored as a character data type.  
  
Definition of Done:  
1. The dropdown list for frequency is implemented and displays the correct options.  
2. The dropdown list is positioned correctly on the "COVER\_SOLUTION" tab page.  
3. Users can select and update the frequency value.  
4. The selected value is stored correctly 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 or table references.

# Package Description Selection

Type: SOLUTION\_PRODUCT

Title: Package Description Selection  
  
Acceptance Criteria:  
1. When the user navigates to the package description field, a list of available packages should be displayed.  
2. The list of available packages should be fetched from the `azbj\_package\_master` table where the `product\_id` matches the `product\_code` of the current record.  
3. Upon selecting a package from the list, the system should clear the blocks `AZBJ\_FUND\_DETAILS` and `AZBJ\_SSO\_FUND\_DETAILS`.  
4. The system should then set the `PORTFOLIO\_STRATEGY` field to an empty string and move the cursor to the `PREMIUM` field.  
  
Definition of Done:  
- The package description field displays a list of available packages when accessed.  
- The list of available packages is correctly filtered based on the current product code.  
- The specified blocks are cleared, and the `PORTFOLIO\_STRATEGY` field is reset upon package selection.  
- The cursor moves to the `PREMIUM` field after the package is selected.  
  
DB queries for Table reference CRUD operations only(With Usage):  
```sql  
SELECT   
FROM azbj\_package\_master   
WHERE product\_id = :correct\_value.product\_code;  
```

# Dynamic Payment Method Update Based on Payout Frequency

Type: SOLUTION\_PRODUCT

Title: Dynamic Payment Method Update Based on Payout Frequency  
  
Acceptance Criteria:  
1. When the payout frequency is '4' or '12', or if the payout frequency is '2' and the FSC code is not one of '2000003024', '7000005464', '7000002372', or '7000000702':  
 - The available payment methods should be:  
 - NACH  
 - Auto Debit Instruction (ADI)  
 - EMANDATE  
 - Credit Card - Standing instruction (CCSI)  
 - SSS  
2. For all other payout frequencies:  
 - The available payment methods should be:  
 - Cash/Cheque/DD  
 - SSS  
 - Direct Debit  
 - ECS  
 - Credit Card - Standing instruction (CCSI)  
 - Group Cash  
 - Group Cheque  
 - Group Direct Credit  
 - Bill Junction - ECS  
 - Auto Debit Instruction (ADI)  
 - BG  
 - NACH  
 - EMANDATE  
3. If the payout frequency is '01':  
 - The payment method selection should be disabled.  
4. If the payout frequency is not '01' and the selected payment method is not 'CCSI' or 'EMAND', and the ADI flag is not 'Y':  
 - The payment method selection should be enabled.  
  
Definition of Done:  
- The system correctly updates the list of available payment methods based on the selected payout frequency and specific conditions.  
- The payment method selection is disabled when the payout frequency is '01'.  
- The payment method selection is enabled when the payout frequency is not '01' and the selected payment method is not 'CCSI' or 'EMAND', and the ADI flag is not 'Y'.  
- The functionality is tested and verified to ensure it meets the acceptance criteria.

# Auto-populate Rider Cover Sum Assured

Type: SOLUTION\_PRODUCT

Title: Auto-populate Rider Cover Sum Assured  
  
Acceptance Criteria:  
1. When the user moves to the next item in the form, the system should automatically navigate to the rider details section.  
2. The system should populate the rider cover sum assured field with the value from the solution product sum assured field.  
3. The system should handle any exceptions gracefully without causing errors.  
  
Definition of Done:  
1. The system navigates to the rider details section upon moving to the next item.  
2. The rider cover sum assured field is correctly populated with the value from the solution product sum assured field.  
3. Any exceptions are handled gracefully, ensuring the system remains stable and functional.  
  
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 Premium Term for Solution Product

Type: SOLUTION\_PRODUCT

Title: Manage Premium Term for Solution Product  
  
Acceptance Criteria:  
1. When the user inputs a premium term in the solution product section, the system should automatically navigate to the rider details section.  
2. The premium term entered in the solution product section should be copied to the corresponding field in the rider details section.  
3. The system should handle any exceptions gracefully without causing errors or interruptions to the user.  
  
Definition of Done:  
- The premium term field is available for input and can accept numeric values.  
- Upon entering a premium term, the system navigates to the rider details section and populates the premium term field there.  
- The system handles exceptions without causing errors or interruptions.  
- The functionality is tested and verified to ensure data consistency and smooth navigation.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Not applicable as the provided XML content does not include specific database queries.

# Dynamic Update of Renewal Payment Methods Based on Frequency and Codes

Type: SOLUTION\_PRODUCT

Title: Dynamic Update of Renewal Payment Methods Based on Frequency and Codes  
  
Acceptance Criteria:  
1. When the frequency is '4' or '12', or when the frequency is '2' and the FSC code is not one of '2000003024', '7000005464', '7000002372', or '7000000702':  
 - The system should clear the current list of renewal payment methods.  
 - The system should add the following payment methods to the list:  
 - NACH  
 - Auto Debit Instruction (ADI)  
 - EMANDATE (EMAND)  
 - Credit Card - Standing instruction (CCSI)  
2. For all other cases:  
 - The system should clear the current list of renewal payment methods.  
 - The system should add the following payment methods to the list:  
 - Cash/Cheque/DD (CA)  
 - SSS (S)  
 - Direct Debit (DD)  
 - ECS  
 - Credit Card - Standing instruction (CCSI)  
 - Group Cash (GCSH)  
 - Group Cheque (GCHQ)  
 - Group Direct Credit (GDC)  
 - Bill Junction - ECS (BJECS)  
 - Auto Debit Instruction (ADI)  
 - BG  
 - NACH  
 - EMANDATE (EMAND)  
3. When the frequency is '01':  
 - If the renewal payment field is enabled, it should be disabled.  
4. For all other frequencies:  
 - If the renewal payment field is disabled and the selected renewal payment method is not 'CCSI' or 'EMAND', and the ADI flag is not 'Y':  
 - The renewal payment field should be enabled.  
  
Definition of Done:  
- The system correctly updates the list of renewal payment methods based on the specified conditions.  
- The renewal payment field is enabled or disabled based on the specified conditions.  
- The changes are tested and verified to ensure they meet the acceptance criteria.  
- The functionality is documented and reviewed.  
  
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.

# Input and Update Benefit Term for Solution Product

Type: SOLUTION\_PRODUCT

Title: Input and Update Benefit Term for Solution Product  
  
Acceptance Criteria:  
1. When the user inputs or updates the benefit term in the solution product section, the system should automatically navigate to the rider details section.  
2. The benefit term value from the solution product section should be copied to the corresponding field in the rider details section.  
3. The system should handle any exceptions gracefully without causing errors or interruptions to the user.  
  
Definition of Done:  
- The benefit term field in the solution product section is functional and allows input and updates.  
- Upon input or update, the system navigates to the rider details section and copies the benefit term value.  
- The system handles exceptions without causing errors or interruptions.  
- 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 database queries.

# Calculate and Update Rider Cover Amounts on Ok Button Press

Type: SOLUTION\_PRODUCT

Title: Calculate and Update Rider Cover Amounts on Ok Button Press  
  
Acceptance Criteria:  
1. When the "Ok" button is pressed, the system should navigate to the rider details block and count the number of records.  
2. For each record in the rider details block:  
 - If the rider cover code starts with 'R', calculate the rider cover amount using the formula: `ROUND(TOTAL\_RIDER\_INVEST SUM\_ASSURED / (SUM\_ASSURED + CORRECT\_VALUE.SUM\_ASSURED), 2)`.  
 - Navigate to the corresponding rider details block and update the rider cover amount using the formula: `ROUND(TOTAL\_RIDER\_INVEST CORRECT\_VALUE.SUM\_ASSURED / (SUM\_ASSURED + CORRECT\_VALUE.SUM\_ASSURED), 2)`.  
 - If the total rider investment is zero, delete the record.  
3. Ensure that the system correctly handles navigation between records and blocks, updating or deleting records as necessary.  
4. The process should end by setting the focus back to the initial item in the solution product block.  
  
Definition of Done:  
- The "Ok" button functionality is implemented and tested.  
- The system accurately calculates and updates rider cover amounts based on the specified formulas.  
- Records with zero total rider investment are deleted.  
- The process handles navigation between records and blocks correctly.  
- The focus is set back to the initial item in the solution product block after the process is complete.  
- All acceptance criteria are met, and the functionality is verified through testing.

# Implement Money Back Option Field

Type: SOLUTION\_PRODUCT

Title: Implement Money Back Option Field  
  
Acceptance Criteria:  
1. The "Money Back Option" field should display a list of values: 5, 8, 10, Self, Policyholder, Other.  
2. The field should be positioned at coordinates (745, 39) on the screen.  
3. The field should have a width of 95 and a height of 14.  
4. The field should have a white background and black text.  
5. The field should be labeled "Money Back Option" with the label aligned to the right.  
6. The label should be bold and have a font size of 800.  
7. The field should be editable, allowing both insertions and updates.  
8. The field should automatically convert input to uppercase.  
9. The field should display a tooltip "[Self/Policyholder/Other]" when hovered over.  
10. The field should be part of the "COVER\_SOLUTION" tab page.  
  
Definition of Done:  
1. The "Money Back Option" field is implemented and displays the correct list of values.  
2. The field is correctly positioned and styled as per the acceptance criteria.  
3. The field is editable and converts input to uppercase.  
4. The tooltip is displayed correctly.  
5. The field is part of the "COVER\_SOLUTION" tab page.  
6. 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 are provided in the XML content for CRUD operations.

# Input and Save Reason Comment for Non-Stop Event

Type: BLK\_NONSTP\_REASON

Title: Input and Save Reason Comment for Non-Stop Event  
  
Acceptance Criteria:  
1. The user should be able to input a reason comment in a text field.  
2. The text field should support multiline input and have a maximum length of 2000 characters.  
3. The user should be able to click a "RECHECK" button to review the entered reason comment.  
4. The user should be able to click a "PROCEED" button to save the reason comment and proceed with the necessary actions.  
  
Definition of Done:  
1. The reason comment text field is displayed and allows multiline input.  
2. The "RECHECK" button is functional and allows the user to review the entered reason comment.  
3. The "PROCEED" button is functional and saves the reason comment, allowing the user to proceed with the next steps.  
4. The UI elements are styled according to the specified design (e.g., font, color, size).  
  
DB queries for Table reference CRUD operations only(With Usage):  
- No specific database queries are mentioned in the provided XML content.

# Recheck Non-Stop Reason and Navigate to Quality Check Questions

Type: BLK\_NONSTP\_REASON

Detailed description: As a user, I want to be able to recheck the non-stop reason and navigate to the quality check questions so that I can verify and update the necessary information.  
  
Acceptance criteria:  
1. When the "RECHECK" button is pressed, the current non-stop reason should be cleared.  
2. The current view should be hidden.  
3. The system should navigate to the quality check questions section.  
4. The first record in the quality check questions should be displayed.  
5. The focus should be set to the answer field in the quality check questions.  
6. The recheck flag should be set to 'Y'.  
7. The stop flag should be set to 'Y'.  
  
Definition of Done:  
- The "RECHECK" button functionality is implemented as described.  
- The system correctly navigates to the quality check questions and sets the appropriate flags.  
- The user can verify that the non-stop reason is cleared and the quality check questions are displayed starting from the first record.  
- The focus is correctly set to the answer field in the quality check questions.  
- 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.

# Save Non-Stop Reason and Update Scrutiny Partner Form

Type: BLK\_NONSTP\_REASON

Detailed description: As a user, I want to be able to save the non-stop reason and update the scrutiny partner form based on the provided application number and previous policy details, so that the system can handle simultaneous proposals and display appropriate messages.  
  
Acceptance criteria:  
1. When the "Proceed" button is pressed, the system should hide the current view.  
2. If the non-stop reason is provided:  
 - The system should navigate to the previous policy block and iterate through the records.  
 - Depending on the type (IP or PH), the system should concatenate the previous policy number to the respective text variable.  
 - The system should update the scrutiny partner form with the details from the current application.  
 - If there is a message, it should be displayed to the user.  
 - The system should check for simultaneous proposals and handle them accordingly, updating the partner ID and displaying relevant messages.  
 - If the insured check is passed, the system should call the merge popup for the appropriate partner type.  
3. If the non-stop reason is not provided, the system should prompt the user to enter the reason and focus on the input field.  
  
Definition of Done:  
- The non-stop reason is saved and the scrutiny partner form is updated correctly.  
- Appropriate messages are displayed to the user based on the conditions.  
- The system handles simultaneous proposals and updates the partner ID as needed.  
- The user is prompted to enter the non-stop reason if it is not provided.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Select scrutiny details from `azbj\_phub\_scrutiny\_prop` based on the application number.  
- Update scrutiny partner form using `azbj\_pk\_scrutiny\_partner.update\_scrutiny\_partner\_form` procedure.

# View and Manage Proposal Details for STP Cases

Type: AUTO\_BBU

Title: View and Manage Proposal Details for STP Cases  
  
Acceptance Criteria:  
1. The system should display the following attributes:  
 - TASA  
 - BMI  
 - Manual Underwriting Status (checkbox)  
 - Reasons (dropdown list)  
 - Comments (text area with vertical scrollbar)  
 - Application Number (hidden)  
 - Proposal Number (hidden)  
 - IP\_CP\_ID  
 - PH\_CP\_ID  
 - UL\_TRAD\_TEXT (disabled text area)  
 - BOP  
 - Receipt Amount (hidden)  
 - Premium (hidden)  
 - Reverse SA (hidden)  
 - Sum Assured (hidden)  
 - SA Difference (hidden)  
2. The "Manual Underwriting Status" checkbox should have values "Y" for checked and "N" for unchecked.  
3. The "Reasons" dropdown list should contain predefined values such as "IP", "PH", "SON", "5", "8", "10", "N", "SL", and "Other".  
4. The "Comments" text area should allow up to 1000 characters and support multiline input.  
5. The "OK" button should be available for user interaction to confirm the details.  
  
Definition of Done:  
- The proposal details screen is implemented and displays all the specified attributes.  
- The "Manual Underwriting Status" checkbox functions correctly with the specified values.  
- The "Reasons" dropdown list is populated with the predefined values.  
- The "Comments" text area supports multiline input and has a vertical scrollbar.  
- The "OK" button is functional and allows users to confirm the details.  
- Hidden fields (Application Number, Proposal Number, Receipt Amount, Premium, Reverse SA, Sum Assured, SA Difference) are not visible to the user but are available for backend processing.  
- The screen layout and design are consistent with the specified positions and styles.  
  
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.

# Enable/Disable Reason and Comments Fields Based on Manual Underwriting Checkbox

Type: AUTO\_BBU

Title: Enable/Disable Reason and Comments Fields Based on Manual Underwriting Checkbox  
  
Acceptance Criteria:  
- When the "Manual Underwriting" checkbox is checked (value 'Y'):  
 - The "Reason" field should be enabled.  
 - The "Comments" field should be enabled.  
- When the "Manual Underwriting" checkbox is unchecked (value 'N'):  
 - The "Reason" field should be cleared and disabled.  
 - The "Comments" field should be cleared and disabled.  
  
Definition of Done:  
- The "Reason" and "Comments" fields are correctly enabled or disabled based on the state of the "Manual Underwriting" checkbox.  
- The fields are cleared when the checkbox is unchecked.  
- The functionality is tested and verified to work as expected in the user interface.

# Dropdown List for Reason Field in Proposal Details

Type: AUTO\_BBU

Title: Dropdown List for Reason Field in Proposal Details  
  
Acceptance Criteria:  
1. The reason field should be a dropdown list.  
2. The dropdown list should contain the following options:  
 - SON  
 - 5  
 - 8  
 - 10  
 - SL  
 - AUTO  
3. The dropdown list should be displayed with a gray background and black text.  
4. The dropdown list should be positioned at the specified location within the form.  
5. The dropdown list should have a maximum length of 100 characters.  
  
Definition of Done:  
- The reason field is implemented as a dropdown list.  
- The dropdown list contains all specified options.  
- The dropdown list is styled and positioned as per the requirements.  
- The dropdown list is functional and allows users to select an option.  
- The changes are tested and verified to ensure they meet the acceptance criteria.

# Enable/Disable Reason and Comments Fields Based on Manual Underwriting Checkbox

Type: AUTO\_BBU

Title: Enable/Disable Reason and Comments Fields Based on Manual Underwriting Checkbox  
  
Acceptance Criteria:  
1. When the "Manual Underwriting" checkbox is checked:  
 - The "Reason" field should be enabled.  
 - The "Comments" field should be enabled.  
2. When the "Manual Underwriting" checkbox is unchecked:  
 - The "Reason" field should be cleared and disabled.  
 - The "Comments" field should be cleared and disabled.  
  
Definition of Done:  
- The "Reason" and "Comments" fields are enabled or disabled based on the state of the "Manual Underwriting" checkbox.  
- The "Reason" and "Comments" fields are cleared when the "Manual Underwriting" checkbox is unchecked.  
- The functionality is tested and verified to work as expected.

# Enable/Disable Reason and Comments Fields Based on Checkbox

Type: AUTO\_BBU

Title: Enable/Disable Reason and Comments Fields Based on Checkbox  
  
Acceptance Criteria:  
- When the "Frequency is as per Proposal Form" checkbox is checked, the "Reason" and "Comments" fields should be enabled.  
- When the "Frequency is as per Proposal Form" checkbox is unchecked, the "Reason" and "Comments" fields should be cleared and disabled.  
  
Definition of Done:  
- The "Reason" and "Comments" fields are enabled when the checkbox is checked.  
- The "Reason" and "Comments" fields are cleared and disabled when the checkbox is unchecked.  
- 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.

# Implement OK Button Functionality for AUTO\_BBU Form

Type: AUTO\_BBU

Detailed description: As a user, I want to ensure that when I click the "OK" button on the AUTO\_BBU form, the system performs a series of validations and database operations to log the event, check certain conditions, and update relevant tables, so that the data integrity and business rules are maintained.  
  
Acceptance criteria:  
1. When the "OK" button is clicked, the system should log the event with the application number and transaction ID.  
2. The system should determine the next event number for the contract and log this information.  
3. If the frequency or rider checkboxes are not selected, the system should display an error message prompting the user to check these fields.  
4. If manual underwriting is selected, the system should ensure that a reason and comments are provided. If not, appropriate error messages should be displayed.  
5. The system should insert a new record into the `azbj\_uw\_comments` table with the event number, contract ID, policy reference, and user comments.  
6. The system should insert a new record into the `azbj\_qc\_autobbu\_log` table with the application number, policy reference, and other relevant details.  
7. If the transaction ID is not null, the system should update the `azbj\_phub\_tracker` table to set the proposal status to "PENDING\_FOR\_AUTO\_BBU" and insert a new record into the `azbj\_auto\_issuance\_grp` table.  
8. The system should commit the transaction and exit the form.  
  
Definition of Done:  
- The "OK" button functionality is implemented as per the acceptance criteria.  
- All relevant database operations are performed correctly.  
- Appropriate error messages are displayed for missing or incorrect data.  
- The form exits successfully after the operations are completed.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Insert into `azbj\_uw\_comments`:  
 ```sql  
 INSERT INTO azbj\_uw\_comments (event\_no, contract\_id, policy\_no, ip\_no, move\_code, policy\_status, user\_id, comment\_date, comments, userrole, opinion, flag)  
 VALUES (v\_event\_no, v\_contract\_id, :correct\_value.new\_policy\_ref, 1, 'AZBJ\_NB', 'PROPOSAL', USER, SYSDATE, SUBSTR(:auto\_bbu.reason || '-' || :auto\_bbu.comments, 1, 1000), 'BBU', NULL, 'N');  
 ```  
  
- Insert into `azbj\_qc\_autobbu\_log`:  
 ```sql  
 INSERT INTO azbj\_qc\_autobbu\_log  
 VALUES (:correct\_value.appln\_no, :correct\_value.new\_policy\_ref, :auto\_bbu.ip\_cp\_id, :auto\_bbu.ph\_cp\_id, :auto\_bbu.tasa, :auto\_bbu.bmi, :auto\_bbu.man\_uw, :auto\_bbu.reason, USER, SYSDATE, :auto\_bbu.comments);  
 ```  
  
- Update `azbj\_phub\_tracker`:  
 ```sql  
 UPDATE azbj\_phub\_tracker  
 SET proposal\_status = 'PENDING\_FOR\_AUTO\_BBU', locking\_flag = 'N'  
 WHERE application\_no = :correct\_value.appln\_no;  
 ```  
  
- Insert into `azbj\_auto\_issuance\_grp`:  
 ```sql  
 INSERT INTO azbj\_auto\_issuance\_grp (contract\_id, policy\_ref, user\_id, create\_date, schedule\_flag, application\_no, bbu\_type, product\_id)  
 VALUES (v\_contract\_id, :correct\_value.new\_policy\_ref, USER, SYSDATE, 'N', :correct\_value.appln\_no, 'AUTO', :correct\_value.product\_code);  
 ```

# Input and Manage AML/KYC Details

Type: CTRL

Title: Input and Manage AML/KYC Details  
  
Acceptance Criteria:  
1. The user should be able to input the PAN number, name, status, partner, and document type.  
2. The input fields should be displayed on a specific tab page named "AML\_KYC".  
3. The "Partner" field should be a dropdown list with three predefined options.  
4. The input fields should be visually organized and aligned horizontally.  
5. The fields should have specific dimensions and positions to ensure a consistent layout.  
  
Definition of Done:  
1. The user can successfully input and save details for PAN number, name, status, partner, and document type.  
2. The "Partner" dropdown list should display three options for selection.  
3. The input fields should be displayed on the "AML\_KYC" tab page.  
4. The layout of the input fields should match the specified dimensions and positions.  
5. The changes 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 any database-specific queries or operations.

# Partner Selection List in AML\_KYC Tab

Type: CTRL

Title: Partner Selection List in AML\_KYC Tab  
  
Acceptance Criteria:  
1. The partner selection list should display predefined values.  
2. The list should be accessible within the "AML\_KYC" tab.  
3. The selected partner value should be stored and retrievable for further processing.  
  
Definition of Done:  
1. The partner selection list is implemented and displays the predefined values.  
2. The list is accessible and functional within the "AML\_KYC" tab.  
3. The selected partner value is correctly stored and can be retrieved for further processing.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include any specific database operations.

# View and Manage Partner Details

Type: PARTNER\_DETAILS

Detailed description: As a user, I want to view and manage partner details including their personal information such as Partner ID, Full Name, Father's Name, Spouse's Name, Address, Telephone, and Date of Birth. Additionally, I want to have the ability to validate the information and exit the form when needed.  
  
Acceptance criteria:  
1. The form should display the following fields for each partner:  
 - Partner ID  
 - Full Name  
 - Father's Name  
 - Spouse's Name  
 - Address  
 - Telephone  
 - Date of Birth  
2. The fields should be read-only and not allow any updates or insertions.  
3. The form should include a "Validate" button to validate the partner details.  
4. The form should include an "Exit" button to close the form.  
5. The form should display checkboxes for additional options.  
6. The form should display a parameter and module information.  
7. The form should be able to handle multiple records, displaying up to 5 records at a time.  
  
Definition of Done:  
- The form displays all required partner details fields.  
- The fields are read-only and do not allow updates or insertions.  
- The "Validate" button is functional and validates the partner details.  
- The "Exit" button is functional and closes the form.  
- Checkboxes for additional options are displayed and functional.  
- Parameter and module information are displayed.  
- The form can display up to 5 records at a time.  
  
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 Merging Process

Type: PARTNER\_DETAILS

Title: Validate Merging Process  
  
Acceptance Criteria:  
1. When the "Validate" button is pressed, the system should check if a specific checkbox (identified as 'control.ch\_merge') is checked.  
2. If the checkbox is checked, the system should execute a validation process.  
3. If the validation process results in a count of zero for a specific reason, the system should prompt the user to select a reason by displaying an error message and moving the cursor to the reason selection field.  
  
Definition of Done:  
- The "Validate" button is functional and triggers the validation process.  
- The system correctly checks the status of the checkbox.  
- The validation process is executed when the checkbox is checked.  
- An error message is displayed, and the cursor is moved to the reason selection field if no reason 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 queries.

# Exit Partner Details with Validation and Merging

Type: PARTNER\_DETAILS

Detailed description: As a user, I want to be able to exit the partner details section of the application, ensuring that all necessary conditions are met before exiting. This includes validating the selection of partners and handling any necessary merging of partner details.  
  
Acceptance criteria:  
1. When the exit button is pressed, the system should check if the "Merge" checkbox is selected.  
2. If the "Merge" checkbox is selected:  
 - The system should count the number of child and master partners selected.  
 - If no partners are selected, the system should prompt the user with a confirmation alert asking if they want to exit without selecting a partner.  
 - If one master partner and one or more child partners are selected, the system should validate and merge the partners.  
 - If the user has chosen specific options (e.g., IP or PH), the system should delete corresponding records from the `azbj\_cp\_merging\_reason` table.  
 - The system should commit the changes.  
 - If the selection criteria are not met, the system should prompt the user to choose a unique master partner and other partners to be merged.  
3. If the "Merge" checkbox is not selected:  
 - The system should check if any reasons are selected.  
 - If no reasons are selected, the system should prompt the user to choose a reason.  
4. The system should handle exceptions and ensure the user is returned to the appropriate state if an error occurs.  
  
Definition of Done:  
- The exit functionality is implemented and tested.  
- The system correctly validates partner selections and handles merging as specified.  
- Appropriate alerts and messages are displayed to the user based on their actions.  
- Changes are committed to the database where necessary.  
- The user is returned to the correct state in case of errors.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- DELETE FROM azbj\_cp\_merging\_reason WHERE application\_no = :correct\_value.appln\_no AND PARTNER\_TYPE = 'IP';  
- DELETE FROM azbj\_cp\_merging\_reason WHERE application\_no = :correct\_value.appln\_no AND PARTNER\_TYPE = 'PH';  
  
Note: The above queries are used to delete records from the `azbj\_cp\_merging\_reason` table based on the application number and partner type.

# Retrieve Partner Details and Address Information

Type: PARTNER\_DETAILS

Detailed description: As a user, I want to be able to enter a Partner ID and retrieve detailed information about the partner, including personal details and address information, so that I can verify and validate the partner's data.  
  
Acceptance criteria:  
1. When a Partner ID is entered, the system should search for the partner using the provided application number.  
2. If multiple partners are found, the system should validate that the selected Partner ID matches the master CP ID.  
3. If the Partner ID does not match, an error message should be displayed, and the user should be prompted to correct the ID.  
4. The system should retrieve and display the partner's personal details, including name, marital status, date of birth, nationality, occupation, gender, email, father's name, spouse's name, place of birth, education, and mobile number.  
5. The system should retrieve and display the partner's residential and mailing address details, including door number, building name, plot number, area, address lines, postcode, phone number, and country name.  
6. If the residential and mailing addresses are the same, a flag should be set to indicate this.  
7. The system should log the retrieval action with relevant details for auditing purposes.  
8. The system should update specific parameters with the retrieved address details.  
  
Definition of Done:  
- The user can enter a Partner ID and retrieve detailed information about the partner.  
- The system performs necessary validations and displays appropriate error messages if needed.  
- The partner's personal and address details are accurately retrieved and displayed.  
- The system logs the retrieval action for auditing purposes.  
- The system updates specific parameters with the retrieved address details.  
  
DB queries for Table reference CRUD operations only (With Usage):  
1. Retrieve partner details:  
 ```sql  
 SELECT a.part\_id, a.before\_title, a.first\_name, a.middle\_name, a.surname,   
 CASE WHEN a.marital\_status='U' THEN 'S' ELSE a.marital\_status END,   
 a.date\_of\_birth, a.nationality, a.occupation, a.sex, a.email,   
 c.father\_name, c.spouse\_name, c.place\_of\_birth, c.education,   
 c.mobile\_no, a.add\_id, c.mail\_add\_id  
 INTO :correct\_value.ip\_part\_id, :correct\_value.ip\_title, :correct\_value.ip\_name,   
 :correct\_value.ip\_middle\_name, :correct\_value.ip\_surname,   
 :correct\_value.ip\_marital\_status, :correct\_value.ip\_dob,   
 :correct\_value.ip\_nationality, :correct\_value.ip\_occupation,   
 :correct\_value.ip\_gender, :correct\_value.ip\_email,   
 :correct\_value.ip\_father, :correct\_value.ip\_spouse,   
 :correct\_value.ip\_birthplace, :correct\_value.ip\_education,   
 :correct\_value.ip\_mobile, v\_res\_add\_id, v\_mail\_add\_id  
 FROM cp\_partners a, azbj\_partner\_extn c  
 WHERE a.part\_id = c.part\_id AND a.part\_id = p\_partner\_id;  
 ```  
  
2. Retrieve residential address details:  
 ```sql  
 SELECT substr(address\_line1, 1, instr(address\_line1, ',') - 1),   
 substr(address\_line1, instr(address\_line1, ',') + 1, length(address\_line1)),   
 substr(address\_line2, 1, instr(address\_line2, '~') - 1),   
 substr(address\_line2, instr(address\_line2, '~') + 1, length(address\_line2)),   
 address\_line3, address\_line4, address\_line5, postcode, telephone, country\_code  
 INTO :correct\_value.ip\_door\_no, :correct\_value.ip\_building\_name,   
 :correct\_value.ip\_plot\_no, :correct\_value.ip\_area,   
 :correct\_value.ip\_address\_line3, :correct\_value.ip\_address\_line4,   
 :correct\_value.ip\_address\_line5, :correct\_value.ip\_pin,   
 :correct\_value.ip\_phone, :correct\_value.ip\_country\_name  
 FROM cp\_addresses  
 WHERE add\_id = v\_res\_add\_id;  
 ```  
  
3. Retrieve mailing address details:  
 ```sql  
 SELECT substr(address\_line1, 1, instr(address\_line1, ',') - 1),   
 substr(address\_line1, instr(address\_line1, ',') + 1, length(address\_line1)),   
 substr(address\_line2, 1, instr(address\_line2, '~') - 1),   
 substr(address\_line2, instr(address\_line2, '~') + 1, length(address\_line2)),   
 address\_line3, address\_line4, address\_line5, postcode, telephone, country\_code  
 INTO :correct\_value.ip\_m\_door\_no, :correct\_value.ip\_m\_building\_name,   
 :correct\_value.ip\_m\_plot\_no, :correct\_value.ip\_m\_area,   
 :correct\_value.ip\_m\_address\_line3, :correct\_value.ip\_m\_address\_line4,   
 :correct\_value.ip\_m\_address\_line5, :correct\_value.ip\_pin,   
 :correct\_value.ip\_phone, :correct\_value.ip\_country\_name  
 FROM cp\_addresses  
 WHERE add\_id = v\_mail\_add\_id;  
 ```  
  
4. Log the retrieval action:  
 ```sql  
 INSERT INTO log\_table (log\_id, appln\_no, log\_message)  
 VALUES (NULL, :correct\_value.appln\_no, '1-->PART\_DET\_PART\_ID-KEY-ENTER: ' || :correct\_value.ip\_pin || ':' || :correct\_value.ip\_area);  
 ```  
  
5. Update parameters with address details:  
 ```sql  
 :parameter.V\_POSTCODE := :correct\_value.IP\_PIN;  
 :parameter.v\_state := :correct\_value.IP\_address\_line5;  
 :parameter.v\_district := :correct\_value.IP\_address\_line4;  
 ```

# Display and Interact with Questions List

Type: BLK\_IND\_QUEST

Title: Display and Interact with Questions List  
  
Acceptance Criteria:  
1. The system should display a list of questions with the following details:  
 - Member Name  
 - Question Description  
 - Form Question Number  
 - Member Number  
 - Answer Description  
 - Correct Value  
2. The questions should be displayed in a scrollable list with a maximum of 5 records visible at a time.  
3. The following fields should be hidden from the user:  
 - Question ID  
 - Sub Question  
4. The "Answer Type" field should be visible and enabled for user interaction.  
5. The "Correct Value" field should allow a maximum length of 1 character.  
6. The "Answer Description" field should allow a maximum length of 500 characters.  
7. The "Member Name", "Question Description", "Form Question Number", and "Member Number" fields should be displayed but not editable by the user.  
  
Definition of Done:  
- The user can see a list of questions with the specified details.  
- The user can scroll through the list if there are more than 5 questions.  
- The hidden fields are not visible to the user.  
- The "Answer Type" field is interactive.  
- The "Correct Value" and "Answer Description" fields adhere to their specified length constraints.  
- The non-editable fields are displayed correctly.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Not applicable as the provided XML content does not include specific SQL queries or table references.

# Validate and Navigate Description Field for Individual Questions

Type: BLK\_IND\_QUEST

Detailed description: As a user, I want to ensure that the description field for an individual question is only filled when the correct value is 'Y', so that the data integrity is maintained and incorrect descriptions are not entered.  
  
Acceptance criteria:  
1. If the correct value for the question is not 'Y' and the description field is not empty, an error message should be displayed stating, "Answer to the question is N. Hence cannot enter description."  
2. When navigating through the records:  
 - If the current record is not the last record, the focus should move to the correct value field of the next record.  
 - If the current record is the last record, the focus should move to the save control.  
3. When navigating to the previous record, the focus should move to the correct value field of the previous record.  
  
Definition of Done:  
- The system should validate the description field based on the correct value.  
- Appropriate error messages should be displayed when validation fails.  
- Navigation through records should work as specified, ensuring smooth user experience.  
  
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.

# Validation and Navigation for Correct Value Field

Type: BLK\_IND\_QUEST

Title: Validation and Navigation for Correct Value Field  
  
Acceptance Criteria:  
1. If the "Correct Value" field is empty, the system should display an error message prompting the user to enter a value.  
2. If the "Correct Value" field contains 'Y', the system should navigate to the "Answer Description" field.  
3. If the "Correct Value" field contains 'N', the system should move to the next record.  
4. If the "Correct Value" field contains any value other than 'Y' or 'N', the system should display an error message indicating that the value should be either 'Y' or 'N'.  
5. When navigating downwards, if the "Correct Value" field is empty, the system should display an error message prompting the user to enter a value.  
6. When navigating downwards, if the "Correct Value" field is not empty and it is not the last record, the system should execute the next item trigger.  
7. When navigating downwards, if the "Correct Value" field is not empty and it is the last record, the system should navigate to the "Save" control.  
  
Definition of Done:  
- The system correctly validates the "Correct Value" field based on the specified criteria.  
- Appropriate error messages are displayed when the "Correct Value" field is empty or contains invalid values.  
- The system navigates to the correct fields or records based on the value in the "Correct Value" field.  
- The functionality is tested and verified to ensure it meets the acceptance criteria.

# Manage Policy Member Details

Type: POLICY\_MEMBER

Detailed description: As a user, I want to manage the details of policy members, including their personal information and health metrics, so that I can maintain accurate and up-to-date records for each member.  
  
Acceptance criteria:  
1. The system should allow the user to input and update the following details for each policy member:  
 - Member Name  
 - Gender  
 - Date of Birth (DOB)  
 - Age (calculated based on DOB)  
 - Height (in cm)  
 - Weight (in kg)  
 - Body Mass Index (BMI, calculated based on height and weight)  
 - Age Proof  
 - Age Proof ID  
 - Relation to the policyholder  
 - Sum Assured (SA)  
 - Partner ID  
  
2. The system should provide a button to add a new member and a button to delete an existing member.  
  
3. The system should automatically calculate and display the age and BMI of the member based on the provided DOB, height, and weight.  
  
4. The system should ensure that the "Add Covers" checkbox is available for each member, allowing the user to specify additional coverage options.  
  
5. The system should validate the input fields to ensure data integrity, such as ensuring that the DOB is in the correct format (dd/mm/yyyy) and that numeric fields like height, weight, and SA are within acceptable ranges.  
  
6. The system should navigate through the member records and update the "Add Covers" checkbox for each member when the user moves to the next item.  
  
Definition of Done:  
- The user can successfully add, update, and delete policy member details.  
- The system correctly calculates and displays the age and BMI of each member.  
- The "Add Covers" checkbox functionality works as expected.  
- All input fields are validated to ensure data integrity.  
- The user interface is intuitive and user-friendly, allowing easy navigation through member records.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Not applicable as the provided XML content does not include specific database queries.

# Input and Update Policy Member Name

Type: POLICY\_MEMBER

Title: Input and Update Policy Member Name  
  
Acceptance Criteria:  
1. The name field should accept input up to 100 characters.  
2. The name should be automatically converted to uppercase.  
3. The name field should be editable and visible to the user.  
4. Upon pressing the next item key, the focus should move to the gender field.  
  
Definition of Done:  
1. The name field is implemented and meets the acceptance criteria.  
2. The functionality is tested and verified to work as expected.  
3. The feature is reviewed and approved by the stakeholders.  
4. The code is merged into the main branch and deployed to the production environment.

# Implement Age Proof Dropdown for Policy Member

Type: POLICY\_MEMBER

Title: Implement Age Proof Dropdown for Policy Member  
  
Acceptance Criteria:  
1. The age proof field should be a dropdown list that allows the user to select from predefined values.  
2. The dropdown list should include the following values: "5", "8", and "No".  
3. The age proof field should be visible and enabled for user interaction.  
4. The field should be positioned correctly on the form and should be labeled as "Age Proof".  
5. When the user navigates to the next item, the system should automatically move to the "MEMBER\_AGE\_PROOF\_ID" field.  
  
Definition of Done:  
1. The age proof dropdown list is implemented and displays the correct predefined values.  
2. The field is visible, enabled, and correctly positioned on the form.  
3. The field is labeled as "Age Proof".  
4. Navigation to the next item works as specified, moving to the "MEMBER\_AGE\_PROOF\_ID" field.  
5. All acceptance criteria are met and tested successfully.

# Automatic Age Calculation Based on Date of Birth

Type: POLICY\_MEMBER

Title: Automatic Age Calculation Based on Date of Birth  
  
Acceptance Criteria:  
1. When the user enters a valid date of birth in the DOB field, the system should automatically calculate the member's age.  
2. The age calculation should consider the current date and specific business rules (e.g., 'P' and 1 as parameters).  
3. If the DOB field is left empty, no age calculation should be performed.  
4. The calculated age should be displayed in the appropriate field (e.g., PM\_AGE).  
5. The system should handle any exceptions gracefully without crashing.  
  
Definition of Done:  
- The user can enter a date of birth in the DOB field.  
- Upon entering a valid DOB, the system calculates the member's age and displays it in the PM\_AGE field.  
- The age calculation logic adheres to the specified business rules.  
- The system does not perform any age calculation if the DOB field is empty.  
- The system handles exceptions without crashing or displaying error messages to the user.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include any direct database queries.

# Calculate BMI and Navigate to Next Field

Type: POLICY\_MEMBER

Title: Calculate BMI and Navigate to Next Field  
  
Acceptance Criteria:  
1. The weight input field should accept numeric values up to one decimal place.  
2. The system should calculate the BMI using the formula: BMI = weight / (height/100)^2, where height is in centimeters.  
3. If the height is greater than 0, the BMI should be calculated and stored in the appropriate field.  
4. After entering the weight, the system should automatically navigate to the next field for the member's age proof.  
  
Definition of Done:  
1. The weight input field is functional and accepts numeric values up to one decimal place.  
2. The BMI calculation is correctly implemented and updates the BMI field when the height is greater than 0.  
3. The system successfully navigates to the next field for the member's age proof after the weight is entered.  
4. All acceptance criteria are met and tested successfully.

# Add Coverage Details for Policy Members

Type: POLICY\_MEMBER

Title: Add Coverage Details for Policy Members  
  
Acceptance Criteria:  
1. When the user selects the "Add Covers" checkbox, the system should check if the checkbox is marked as 'Y'.  
2. If the checkbox is marked as 'Y', the system should retrieve the current record of the policy member.  
3. The system should then call a function to assign coverage details to the policy member based on the member's cover code, sum assured, and age.  
  
Definition of Done:  
- The "Add Covers" checkbox is functional and triggers the appropriate logic when selected.  
- The system correctly identifies the current policy member record.  
- The function to assign coverage details is called with the correct parameters.  
- The coverage details are successfully assigned to the policy member.  
  
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 Relationship Type for Policy Member

Type: POLICY\_MEMBER

Title: Select Relationship Type for Policy Member  
  
Acceptance Criteria:  
1. When a relationship type is selected from the list, the system should check if the selected relationship is not null.  
2. If the relationship is not null, the system should determine the cover code for the policy member based on the selected relationship and the current record.  
3. The system should update the policy member's cover code with the determined value.  
4. The system should add or remove spouse details based on the presence of a spouse.  
5. The system should navigate to the "Sum Assured" field after processing the relationship type.  
  
Definition of Done:  
1. The relationship type can be selected from a predefined list.  
2. The system correctly determines and updates the cover code based on the selected relationship.  
3. Spouse details are correctly added or removed as needed.  
4. The system navigates to the "Sum Assured" field after processing the relationship type.  
5. 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 specific CRUD operations that can be directly executed in the database.

# Gender Selection for Policy Member

Type: POLICY\_MEMBER

Title: Gender Selection for Policy Member  
  
Acceptance Criteria:  
1. The gender field should be a dropdown list with the following options:  
 - 5  
 - 8  
 - 10  
 - SL (No)  
2. The gender field should be displayed on the "Family Members" tab.  
3. The gender field should be visible and enabled for both insertion and updates.  
4. The gender field should have a maximum length of 6 characters and should be displayed in uppercase.  
5. Upon selecting a gender and pressing the "Next" key, the focus should move to the "Member Date of Birth" field.  
  
Definition of Done:  
1. The gender dropdown list is implemented and displays the correct options.  
2. The gender field is correctly positioned on the "Family Members" tab and is visible and enabled.  
3. The gender field adheres to the specified display and input constraints (maximum length, uppercase).  
4. The navigation logic is implemented to move the focus to the "Member Date of Birth" field upon pressing the "Next" key after selecting a gender.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include specific database queries.

# Input and Update Policy Member Height

Type: POLICY\_MEMBER

Title: Input and Update Policy Member Height  
  
Acceptance Criteria:  
1. The height field should accept numeric input up to one decimal place (e.g., 999.9).  
2. The height field should be visible and enabled for user interaction.  
3. The height field should have a maximum length of 5 characters.  
4. The height field should be displayed with a gray background and black text.  
5. The height field should be positioned correctly on the form.  
6. The height field should be labeled "Ht(cm)" and the label should be bold and centered above the field.  
7. The height field should allow both insertion and updating of data.  
8. Upon pressing the key to navigate to the next item, the focus should move to the weight input field.  
  
Definition of Done:  
- The height field meets all the specified acceptance criteria.  
- The height field is tested and verified to accept and display data correctly.  
- The navigation from the height field to the weight field works as expected.  
- The user interface is reviewed and approved for correct positioning and styling of the height field and its label.

# 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 current record in the policy member list.  
 - Retrieve the cover code of the member to be deleted.  
 - Delete the current member record.  
 - Navigate to the rider details section and delete any associated rider details that match the cover code of the deleted member.  
 - Return to the policy member list and update the cover codes for the remaining members based on their relationship.  
 - Ensure that the spouse details are updated accordingly.  
  
Definition of Done:  
- The "Delete Member" button is functional and performs the deletion of the member and associated rider details as specified.  
- The cover codes for the remaining members are updated correctly.  
- Spouse details are updated as per the business logic.  
- The user interface reflects the changes immediately after the deletion process is completed.  
- All changes are saved to the database successfully.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- No specific DB queries provided in the XML content that can be directly executed without modification.

# Input and Update Age Proof ID for Policy Member

Type: POLICY\_MEMBER

Title: Input and Update Age Proof ID for Policy Member  
  
Acceptance Criteria:  
1. The age proof ID field should be enabled and allow for text input.  
2. The field should accept a maximum of 20 characters and convert all input to uppercase.  
3. The field should be visible and positioned appropriately on the user interface.  
4. The field should be labeled "Age Proof ID" and the label should be bold and centered above the field.  
5. The field should allow for both insertion and updating of data.  
6. Upon pressing the key to navigate to the next item, the system should automatically move the focus to the "Member Relation" field.  
  
Definition of Done:  
1. The age proof ID field is implemented and meets all acceptance criteria.  
2. The field is tested to ensure it accepts input correctly and navigates to the next item as specified.  
3. The user interface displays the field and its label correctly.  
4. The functionality is verified to ensure data can be inserted and updated in the field.  
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.

# Add New Member to Policy Member List

Type: POLICY\_MEMBER

Detailed description: As a user, I want to add a new member to the policy member list by clicking a button, so that I can easily manage and update the list of members associated with a policy.  
  
Acceptance criteria:  
1. When the "Add Member" button is clicked, the system should navigate to the last record in the policy member list.  
2. If the current member name field is not empty, the system should move to the next record.  
3. The system should create a new record and set the focus to the member name field for the new record.  
4. If the user attempts to navigate to the next item, an alert should prompt the user to confirm if they wish to add more members.  
5. If the user confirms, the system should move to the next record and set the focus to the member relation field.  
6. If the user does not confirm, the system should populate the member questionnaire and set the focus to the correct field in the questionnaire.  
  
Definition of Done:  
- The "Add Member" button functionality is implemented and tested.  
- The system correctly navigates and creates new records as per the acceptance criteria.  
- The alert prompt for adding more members is functional and correctly handles user responses.  
- The member questionnaire is populated and the focus is set correctly based on user input.  
- 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 specific CRUD operations or SQL queries that can be directly executed in the database.

# Capture Age of Policy Member

Type: POLICY\_MEMBER

Title: Capture Age of Policy Member  
  
Acceptance Criteria:  
1. The age field should be a numeric input field.  
2. The field should be labeled "Age" and should be positioned appropriately on the form.  
3. The field should not allow insertion or updates directly by the user.  
4. When the user navigates away from the age field, the system should automatically move the cursor to the "Member Height" field.  
  
Definition of Done:  
1. The age field is implemented as a numeric input field.  
2. The field is labeled "Age" and is positioned correctly on the form.  
3. The field does not allow direct insertion or updates by the user.  
4. The cursor automatically moves to the "Member Height" field when the user navigates away from the age 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.

# Manage and View SIP Control Records

Type: SIP\_CONTROL

Detailed description: As a user, I want to manage and view the details of SIP (Systematic Investment Plan) control records, including field values, proof types, proof descriptions, document dates, derived income, and derived Tasa values, so that I can ensure accurate and comprehensive data management for SIP controls.  
  
Acceptance criteria:  
1. The system should allow users to input and view the following details for SIP control records:  
 - Field Value  
 - Surrogate Proof Type  
 - Proof Description  
 - Document Date  
 - Derived Income (read-only)  
 - Derived Tasa Value (read-only)  
2. The system should display these details in a structured format with appropriate labels and alignment.  
3. The Derived Income and Derived Tasa Value fields should be non-editable by the user.  
4. The system should support a list of predefined values for the Surrogate Proof Type field.  
  
Definition of Done:  
- The user interface for managing SIP control records is implemented and accessible.  
- Users can input and view all required details for SIP control records.  
- Derived Income and Derived Tasa Value fields are displayed as read-only.  
- The Surrogate Proof Type field supports a predefined list of values.  
- The user interface is tested and verified for accuracy and usability.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Not applicable as the provided XML content does not include specific SQL queries or table references.

# Calculate Entry Age and Derive Surrogate Value

Type: SIP\_CONTROL

Title: Calculate Entry Age and Derive Surrogate Value  
  
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 policy date.  
2. If the insured person's entry age is less than 18, the policyholder's entry age should be used; otherwise, the insured person's entry age should be used.  
3. The system should call a function to derive a surrogate value using the calculated age, proof type, and field value.  
4. The system should handle any exceptions gracefully without causing a crash.  
  
Definition of Done:  
1. The entry age calculation logic is implemented and tested.  
2. The surrogate value derivation function is called with the correct parameters.  
3. The system handles exceptions without crashing.  
4. The functionality is verified through unit tests and integration tests.  
  
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 Proof Type Dropdown for SIP Control

Type: SIP\_CONTROL

Title: Implement Proof Type Dropdown for SIP Control  
  
Acceptance Criteria:  
1. The proof type field should display a dropdown list with the following options:  
 - IP  
 - 5  
 - 8  
 - 10  
 - SL  
2. The dropdown list should be positioned at coordinates (30, 75) on the screen.  
3. The dropdown list should have a width of 62 units and a height of 21 units.  
4. The dropdown list should have a white background and black text.  
5. The font used for the dropdown list should be Tahoma, with a font size of 8 and a demilight weight.  
6. The prompt for the dropdown list should read "Surrogate Proof Type" and should be bold and plain in style.  
7. The prompt should be positioned with an offset of -67 for attachment and -24 for alignment.  
  
Definition of Done:  
- The dropdown list for proof type is implemented and displays the correct options.  
- The dropdown list is correctly positioned and styled as per the acceptance criteria.  
- The prompt for the dropdown list is correctly displayed and positioned.  
- The functionality is tested and verified to ensure that users can select a proof type from 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.

# Manage Child Cover Details

Type: BLK\_CHILD\_COVERS

Detailed description: As a user, I want to manage child cover details within a proposal form, so that I can add, view, and delete child cover information efficiently.  
  
Acceptance criteria:  
1. The user should be able to add a new child cover by entering the child's name, date of birth, age, and percentage of sum assured.  
2. The system should automatically calculate and display the child's age based on the date of birth entered.  
3. The user should be able to delete a child cover by selecting the corresponding checkbox and clicking the "Delete" button.  
4. The user should be able to exit the child cover management section by clicking the "Exit" button.  
  
Definition of Done:  
1. The user interface should allow for the input of child cover details including name, date of birth, age, and percentage of sum assured.  
2. The age field should be non-editable and automatically calculated based on the date of birth.  
3. The "Add Child" button should add a new child cover entry to the list.  
4. The "Delete" button should remove the selected child cover entry from the list.  
5. The "Exit" button should close the child cover management section and return the user to the main proposal form.  
6. 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.

# Add Child Record Functionality

Type: BLK\_CHILD\_COVERS

Title: Add Child Record Functionality  
  
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 child-related information.  
  
Definition of Done:  
- The "Add Child" button is functional and performs the described actions.  
- The system successfully navigates to the child covers section.  
- The system moves to the last record in the child covers section.  
- A new record is created in the child covers section, ready for user input.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include specific database queries.

# 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 child cover 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 and repeat the process until all records have been checked.  
6. After processing all records, the system should navigate back to the first record.  
  
Definition of Done:  
- The delete button functionality is implemented and tested.  
- The system correctly navigates through the child cover records and deletes the marked records.  
- The user is able to see the updated list of child cover records after the deletion process.  
- All acceptance criteria are met and verified through testing.

# Calculate Child's Age Based on Date of Birth

Type: BLK\_CHILD\_COVERS

Title: Calculate Child's Age Based on Date of Birth  
  
Acceptance Criteria:  
1. When the user enters a valid date of birth for the child, the system should automatically calculate the child's age.  
2. The age calculation should be based on the current date and should follow the specific business logic provided by the function `azbj\_calc\_entry\_age`.  
3. If the date of birth is not provided or is invalid, the child's age should be set to null.  
  
Definition of Done:  
1. The user can enter the child's date of birth in the specified format (dd/mm/yyyy).  
2. Upon entering a valid date of birth, the child's age is calculated and displayed correctly.  
3. The system handles exceptions gracefully, setting the child's age to null if any errors occur during the calculation.  
  
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 Child Covers Section

Type: BLK\_CHILD\_COVERS

Detailed description: As a user, I want to be able to exit the "Child Covers" section of the application easily, so that I can navigate back to the main control screen without any hassle.  
  
Acceptance criteria:  
1. When the "Exit" button is pressed, the system should refresh the data in the "Child Covers" section.  
2. The "Child Covers" section should be hidden from view.  
3. The window displaying the "Child Covers" section should be closed.  
4. The focus should be moved back to the main control screen.  
  
Definition of Done:  
- The "Exit" button is functional and performs all the actions as specified in the acceptance criteria.  
- The user is able to navigate back to the main control screen seamlessly after pressing the "Exit" button.  
- The "Child Covers" section and its window are properly hidden and closed.  
- The data in the "Child Covers" section is refreshed before it is hidden.  
  
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 Portfolio of Funds

Type: UL\_SOL\_AT\_PORTFOL\_2

Title: Manage Portfolio of Funds  
  
Acceptance Criteria:  
1. The user should be able to view a list of funds with their IDs, names, and apportionment percentages.  
2. The user should be able to add new funds to the portfolio by entering the fund ID, name, and apportionment percentage.  
3. The user should be able to update the apportionment percentage of existing funds.  
4. The user should be able to delete funds from the portfolio.  
5. The total apportionment percentage should be automatically calculated and displayed.  
6. The fund ID should be selected from a predefined list of values (LOV) that fetches data from the database based on specific criteria.  
  
Definition of Done:  
1. The user interface allows for the display, addition, update, and deletion of fund details.  
2. The total apportionment percentage is correctly calculated and displayed.  
3. The fund ID LOV fetches data correctly from the database based on the specified criteria.  
4. All CRUD operations are functional and meet the acceptance criteria.  
5. The feature is tested and verified to work as expected.  
  
DB queries for Table reference CRUD operations only (With Usage):  
```sql  
-- Fetching fund details for LOV  
SELECT fund\_short\_name AS FUND\_ID, 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 = :correct\_value.product\_CODE  
AND :susac.sa\_daterecd >= NVL(start\_date, '01-apr-2005')  
AND :susac.sa\_daterecd <= NVL(end\_date, '01-jan-3000');  
```

# Validate Apportionment Percentage and Navigation

Type: UL\_SOL\_AT\_PORTFOL\_2

Title: Validate Apportionment Percentage and Navigation  
  
Acceptance Criteria:  
1. When the user enters an apportionment percentage, the system should validate that the value is between 0 and 100.  
2. If the entered value is outside this range, the system should display a warning message: "Apportionment percentage should be between 0 and 100" and prevent the user from proceeding.  
3. The system should sum up all apportionment percentages in the current block and ensure that the total equals 100.  
4. If the total apportionment percentage equals 100 and the product code is 71 with a specific product definition, the system should enable certain fields and navigate to the appropriate block and item.  
5. If the total apportionment percentage does not equal 100, the system should navigate to a specific item in the current block.  
  
Definition of Done:  
- The validation logic for apportionment percentage is implemented and tested.  
- The warning message is displayed correctly when the entered value is outside the valid range.  
- The system correctly sums up the apportionment percentages and performs the necessary actions based on the total.  
- Navigation to the appropriate blocks and items is functioning 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):  
- Not applicable as the provided XML content does not include specific CRUD operations that can be directly executed in the database without modification.

# Display Fund Name Based on Proposal Type and Number of Pages

Type: UL\_SOL\_AT\_PORTFOL\_2

Title: Display Fund Name Based on Proposal Type and Number of Pages  
  
Acceptance Criteria:  
1. When the user navigates to the fund name field, the system should load the field from an image based on the current item.  
2. If the proposal type is 'N' and the number of pages is 50, the system should zoom into the image 'PROPOSALFORM.image51' at 50% zoom.  
3. On double-clicking the fund name field, the system should display a list of values (LOV) for fund selection.  
4. When the user presses a key to list values, the system should show the LOV for fund selection.  
5. When the user navigates to the next item, the system should clear the visual attribute of the current item and navigate to the 'APPORTIONMENT' item.  
6. When the user navigates to the fund name field, the system should load the field from an image based on the current item.  
  
Definition of Done:  
- The fund name field should display the correct fund information based on the proposal type and number of pages.  
- The system should handle user interactions such as double-clicking and key presses to display the LOV for fund selection.  
- The system should navigate correctly between items and clear visual attributes as specified.  
- The functionality should be 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 fetch fund information for the LOV:  
```sql  
SELECT fund\_short\_name AS FUND\_ID, 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 = :correct\_value.product\_CODE  
AND :susac.sa\_daterecd >= NVL(start\_date, '01-apr-2005')  
AND :susac.sa\_daterecd <= NVL(end\_date, '01-jan-3000');  
```

# Form Field Navigation and Validation

Type: CORRECT\_VALUE

Detailed description: As a user, I want to ensure that the system correctly handles the navigation and validation of form fields within the "CORRECT\_VALUE" section, so that data entry is accurate and efficient.  
  
Acceptance criteria:  
1. When navigating to the previous item, the system should:  
 - Check if the current block is "CORRECT\_VALUE".  
 - Retrieve the previous item based on a mapping table.  
 - Clear any visual attributes from the current item.  
 - Navigate to the previous item.  
 - If the block is not "CORRECT\_VALUE", simply navigate to the previous item.  
  
2. When navigating to the next item, the system should:  
 - Perform all necessary validations on the current item.  
 - Clear any visual attributes from the current item.  
 - Navigate to the next item.  
 - If the product group flag is set, clear specific fields in the "CORRECT\_VALUE" and "CONTROL" blocks.  
  
3. When a text item is focused, the system should:  
 - Load the corresponding field from an image based on the current item.  
  
4. When a mouse click occurs, the system should:  
 - Navigate to the clicked item if it is not a button.  
  
5. When an error occurs, the system should:  
 - Suppress specific error messages.  
 - Set the system message level to a higher value to prevent further error messages.  
  
Definition of Done:  
- The system correctly navigates between form fields based on the specified logic.  
- All validations are performed as expected.  
- Visual attributes are cleared appropriately.  
- The system handles mouse clicks and errors as specified.  
- The functionality is tested and verified to ensure accuracy and efficiency in data entry.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- No specific DB queries are mentioned that can be executed directly in the database without modification.  
  
Explanation of Oracle Form Logics:  
- The `KEY-PREV-ITEM` trigger handles navigation to the previous item by checking the current block and retrieving the previous item from a mapping table.  
- The `KEY-NEXT-ITEM` trigger performs validations and navigates to the next item, with additional logic for clearing fields based on a product group flag.  
- The `PRE-TEXT-ITEM` trigger loads the corresponding field from an image when a text item is focused.  
- The `WHEN-MOUSE-CLICK` trigger navigates to the clicked item if it is not a button.  
- The `ON-ERROR` trigger suppresses specific error messages and sets the system message level to prevent further error messages.

# Dropdown for Skip Reason Selection

Type: CORRECT\_VALUE

Title: Dropdown for Skip Reason Selection  
  
Acceptance Criteria:  
1. The list of options should include the following values:  
 - IP  
 - SON  
 - 5  
 - 8  
 - SL  
2. The list should be displayed in a dropdown format.  
3. The dropdown should be positioned at the specified location on the user interface.  
4. The dropdown should be styled with the specified font, size, and colors.  
  
Definition of Done:  
- The dropdown list is implemented and displays the correct options.  
- The dropdown is positioned correctly on the user interface.  
- The dropdown is styled according to the specified requirements.  
- The functionality is tested and verified to ensure that the correct value is saved when an option is selected.  
  
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 Select Image Paths

Type: CORRECT\_VALUE

Title: View and Select Image Paths  
  
Acceptance Criteria:  
1. The user should be able to see a list of predefined image paths.  
2. The list should include options such as "5", "8", and "SL".  
3. The user should be able to select an image path from the list.  
4. The selected image path should be displayed in a designated area on the screen.  
  
Definition of Done:  
1. The list of image paths is visible and selectable.  
2. The selected image path is displayed correctly.  
3. The user interface is intuitive and user-friendly.  
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.

# Employee ID Field Validation and Processing

Type: CORRECT\_VALUE

Title: Employee ID Field Validation and Processing  
  
Acceptance Criteria:  
1. When the user double-clicks on the Employee ID field:  
 - If the value of `fsc\_code` is '7100000003', the system should display a message prompting the user to enter the Employee ID manually.  
 - If the value of `fsc\_code` is not '7100000003', the system should display a list of values for the user to select from.  
  
Definition of Done:  
1. The Employee ID field should be enabled and allow for both insertion and updates.  
2. The field should have a maximum length of 30 characters and should be displayed in uppercase.  
3. The field should be visually styled with a white background, black text, and a lowered bevel.  
4. The prompt for the Employee ID field should be displayed to the right of the field, in bold and plain style.  
5. The system should handle the double-click event on the Employee ID field as per the specified conditions.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Not applicable as the provided XML content does not include specific database queries.

# Implement Agent Type Dropdown

Type: CORRECT\_VALUE

Title: Implement Agent Type Dropdown  
  
Acceptance Criteria:  
1. The agent type field should be a dropdown list.  
2. The dropdown list should contain the following options:  
 - 8  
 - 10  
 - SL (No)  
3. The field should be enabled and allow both insertion and updates.  
4. The field should be visually styled with a white background, black text, and bold font.  
5. The prompt for the field should be "Agent Type" and should be right-aligned.  
  
Definition of Done:  
1. The agent type dropdown list is implemented and displays the correct options.  
2. The field is enabled and allows for data entry and updates.  
3. The visual styling of the field matches the specified requirements.  
4. The prompt "Agent Type" is displayed and right-aligned next to the 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.

# Validation for Multiple Policy Count Field

Type: CORRECT\_VALUE

Title: Validation for Multiple Policy Count Field  
  
Acceptance Criteria:  
1. If the "Multiple Policy Count" field is not empty and the "Multiple Policy" checkbox is not checked, display a warning message: "Please check the multiple policy checkbox".  
2. Clear the "Multiple Policy Count" field if the above condition is met.  
3. Move the cursor to the "Multiple Policy" checkbox field after displaying the warning message.  
  
Definition of Done:  
- The "Multiple Policy Count" field should only retain a value if the "Multiple Policy" checkbox is checked.  
- A warning message should be displayed if the "Multiple Policy Count" field is filled without the "Multiple Policy" checkbox being checked.  
- The cursor should automatically move to the "Multiple Policy" checkbox field when the warning message is displayed.  
  
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 and Validate Sub IC Code

Type: CORRECT\_VALUE

Title: Input and Validate Sub IC Code  
  
Acceptance Criteria:  
1. The "Sub IC Code" field should allow users to input a code up to 30 characters in length.  
2. The field should automatically convert any input to uppercase.  
3. The field should be enabled and allow both insert and update operations.  
4. When a user double-clicks on the "Sub IC Code" field, a list of valid values should be displayed for selection.  
5. The list of valid values should include:  
 - "SON"  
 - "5"  
 - "8"  
 - "1"  
 - "SL"  
6. The field should be visually styled with a white background, black text, and a lowered bevel.  
7. The field should be positioned correctly on the form and be part of the "AGENT" tab page.  
  
Definition of Done:  
- The "Sub IC Code" field is implemented and meets all acceptance criteria.  
- The field is tested to ensure it functions correctly, including validation and list display.  
- 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 specific database queries.

# Validate Application Number and Authorize Scrutiny Failure Process

Type: CORRECT\_VALUE

Detailed description: As a user, I want to ensure that the application number is validated and that only authorized users can initiate the scrutiny failure process, so that the process is secure and accurate.  
  
Acceptance criteria:  
1. When the "Scrutiny Failure" button is pressed, the system should check if the application number is not null.  
2. If the application number is null, the system should prompt the user to enter an application number and display an error message stating "Application number cannot be null."  
3. If the user ID starts with 'UU', the system should create a parameter list with the following parameters:  
 - `RES\_TYPE` set to 'QC'  
 - `RES\_appln` set to the application number  
4. The system should then call the scrutiny failure process with the created parameter list.  
5. If the user ID does not start with 'UU', the system should display an error message stating "Not an Authorized ID."  
  
Definition of Done:  
- The application number validation is implemented and prompts the user if the application number is null.  
- The system correctly identifies authorized users based on the user ID prefix.  
- The scrutiny failure process is initiated with the correct parameters for authorized users.  
- Error messages are displayed appropriately for both null application numbers and unauthorized user IDs.  
  
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.

# Display list of agents on double-clicking 'Lead By' field

Type: CORRECT\_VALUE

Detailed description: As a user, I want the system to display a list of agents when I double-click on the "Lead By" field, so that I can select an appropriate agent based on specific criteria.  
  
Acceptance criteria:  
1. When the "Lead By" field is double-clicked, the system should check if the value of `get\_bnkasr` is one of the following: 'T', 'I', 'O', 'DM', 'LG', 'BP' or contains 'X'.  
2. If the condition is met, the system should display a list of agents.  
3. The list of agents should be fetched using the following query:  
 ```sql  
 SELECT reference\_code code,  
 DECODE(c.partner\_type, 'P', c.first\_name || ' ' || c.middle\_name || ' ' || c.surname, c.institution\_name) name  
 FROM dmt\_agents a, azbj\_agents\_ext b, cp\_partners c  
 WHERE a.int\_id = b.int\_id  
 AND a.cust\_part\_unique\_code = c.partner\_ref  
 AND recruited\_by = :CORRECT\_VALUE.FSC\_CODE  
 UNION ALL  
 SELECT 'NOSUBCODE', 'NOSUBCODE' FROM dual;  
 ```  
  
Definition of Done:  
1. The "Lead By" field should be enabled and allow both insertion and updates.  
2. The system should correctly display the list of agents when the "Lead By" field is double-clicked, based on the specified criteria.  
3. The list of agents should be fetched and displayed as per the provided SQL query.  
4. The functionality should be tested and verified to ensure it meets the acceptance criteria.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- The provided SQL query is used to fetch the list of agents based on the criteria specified in the acceptance criteria.

# Broker Model Dropdown Implementation

Type: CORRECT\_VALUE

Title: Broker Model Dropdown Implementation  
  
Acceptance Criteria:  
1. The broker model field should be a dropdown list with the following options:  
 - 5  
 - 8  
 - 10  
 - LA  
 - No  
2. The dropdown list should be displayed on the "AGENT" tab.  
3. The field should be enabled and allow both insertions and updates.  
4. The field should be case-insensitive and convert all input to uppercase.  
5. The field should have a maximum length of 100 characters.  
6. The field should be visually styled with a white background, black text, and a lowered bevel.  
7. The prompt for the field should be "Type of Broker model" and should be right-justified.  
8. The prompt should be bold and plain, with a font size of 800.  
  
Definition of Done:  
- The dropdown list for the broker model is implemented and functional.  
- The field adheres to the specified visual and functional requirements.  
- The field is tested to ensure it allows insertions and updates.  
- The field is tested to ensure it converts all input to uppercase.  
- The prompt is correctly displayed and styled as specified.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- No specific DB queries provided in the XML content.

# Input and Validate Referral Code

Type: CORRECT\_VALUE

Title: Input and Validate Referral Code  
  
Acceptance Criteria:  
1. The referral code field should accept a maximum of 30 characters.  
2. The referral code should be displayed in uppercase.  
3. The field should be enabled and allow both insert and update operations.  
4. When the user double-clicks on the referral code field, a list of values (LOV) should be displayed, allowing the user to select from predefined options.  
5. The LOV should fetch data from the "SYNDICATE" record group, which retrieves branch codes and names based on the type of branch.  
6. The referral code field should be visually distinct with a white background and black text.  
  
Definition of Done:  
- The referral code field is implemented and meets all the acceptance criteria.  
- The LOV functionality is working as expected, displaying the correct branch codes and names.  
- The field is tested for both insert and update operations.  
- The field's appearance matches the specified design requirements.  
  
DB queries for Table reference CRUD operations only (With Usage):  
```sql  
SELECT branch\_code, branch\_name  
FROM azbj\_syn\_bank\_branches  
WHERE br\_type = (  
 CASE   
 WHEN :CORRECT\_VALUE.FSC\_CODE LIKE '522%' THEN 'B'  
 ELSE SUBSTR(:CORRECT\_VALUE.FSC\_CODE, 1, 3)  
 END  
);  
```  
- This query is used to fetch the branch codes and names for the LOV based on the type of branch.

# Auto-populate agent and recruiter information based on FSC/IC code

Type: CORRECT\_VALUE

Detailed description: As a user, I want the system to automatically populate the full name, recruited by, and branch code fields when I enter a valid FSC/IC code, so that I can quickly retrieve and verify agent information.  
  
Acceptance criteria:  
1. When a valid FSC/IC code is entered:  
 - The system should fetch and display the full name, recruited by, and branch code of the agent from the `azbj\_v\_agents` table.  
 - If the recruited by code is present, the system should also fetch and display the full name and branch code of the recruiter from the `azbj\_v\_agents` table.  
 - The system should check if the FSC/IC code matches a specific pattern in the `azbj\_system\_constants` table and set a flag (`v\_web\_aggr`) accordingly.  
 - If no data is found for the recruiter, the system should handle the exception gracefully without causing errors.  
  
Definition of Done:  
- The system correctly populates the agent and recruiter information based on the FSC/IC code.  
- The system sets the `v\_web\_aggr` flag based on the pattern match in the `azbj\_system\_constants` table.  
- The system handles exceptions gracefully when no data is found for the recruiter.  
- The functionality is tested and verified to ensure it works as expected.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Fetch agent information:  
 ```sql  
 SELECT full\_name, recruited\_by, branch\_code  
 INTO :correct\_value.ic\_name, :correct\_value.stm\_code, :correct\_value.fsc\_branch  
 FROM azbj\_v\_agents  
 WHERE reference\_code = :correct\_value.fsc\_code;  
 ```  
  
- Fetch recruiter information:  
 ```sql  
 SELECT full\_name, branch\_code  
 INTO :correct\_value.stm\_name, :correct\_value.stm\_branch  
 FROM azbj\_v\_agents  
 WHERE reference\_code = :correct\_value.stm\_code;  
 ```  
  
- Check pattern match in system constants:  
 ```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 :correct\_value.fsc\_code LIKE char\_value || '%' AND ROWNUM = 1;  
 ```

# Enter and Validate Backdated Policy Date

Type: CORRECT\_VALUE

Title: Enter and Validate Backdated Policy Date  
  
Acceptance Criteria:  
1. The backdated policy date should be entered in the format "dd/mm/yyyy".  
2. The system should validate that the backdated policy date is not greater than the current date (opus date).  
3. If the backdated policy date is not equal to the current date, the system should calculate the insured person's age based on the provided date of birth and the backdated policy date.  
4. If the insured person is not checked, the system should set the policyholder's age to the insured person's age.  
5. The system should navigate to the next item after processing the backdated policy date.  
  
Definition of Done:  
1. The user can successfully enter a backdated policy date in the specified format.  
2. The system performs the necessary validations and calculations as per the acceptance criteria.  
3. The system navigates to the next item after processing the backdated policy date.  
4. All validations and calculations are tested and verified to be working correctly.  
5. The feature is documented and reviewed for accuracy and completeness.

# Nationality Field Implementation

Type: CORRECT\_VALUE

Title: Nationality Field Implementation  
  
Acceptance Criteria:  
1. The nationality field should display a list of predefined options including "IP", "SON", "5", "8", "LA", and "SL".  
2. The default value for the nationality field should be set to "IN".  
3. The nationality field should be case-insensitive and convert all input to uppercase.  
4. The nationality field should be located at the specified position on the form and should be visually aligned with other form elements.  
5. The nationality field should be editable and allow users to update the value.  
6. The nationality field should have a prompt labeled "Citizenship" that is bold and right-aligned.  
  
Definition of Done:  
- The nationality field is implemented and displays the predefined list of options.  
- The default value is set to "IN".  
- The field converts all input to uppercase.  
- The field is positioned correctly on the form and visually aligned.  
- The field is editable and allows updates.  
- The prompt "Citizenship" is bold and right-aligned.  
  
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 Nominee Fields Based on Proposal Type

Type: CORRECT\_VALUE

Title: Manage Nominee Fields Based on Proposal Type  
  
Acceptance Criteria:  
1. When the proposal type is 'K', 'PS', 'H', or 'E':  
 - The nominee-related fields (nominee name, nominee birthplace, nominee date of birth, nominee relation) should be disabled and not allow insert or update operations.  
 - The message "Nomination not allowed for this policy" should be displayed in the nominee name field.  
 - For proposal type 'E', the assign flag should be set to 'Y' and disabled.  
2. When the proposal type is 'I':  
 - The multi-nominee control should be visible and enabled.  
3. For other proposal types:  
 - The nominee-related fields should be enabled and allow insert and update operations.  
 - The assign flag should be enabled if it was previously disabled.  
4. If the proposal type is 'MWP':  
 - The nominee name field should be disabled and not allow insert or update operations.  
5. The system should fetch the proposal type from the database using the application number and handle any exceptions by setting a default value.  
  
Definition of Done:  
- The system correctly manages the visibility and editability of nominee-related fields based on the selected proposal type.  
- All acceptance criteria are met.  
- The functionality is tested and verified to ensure it works as expected.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- The system should execute the following query to fetch the proposal type:  
 ```sql  
 SELECT proposal\_type  
 INTO v\_prop\_type  
 FROM azbj\_phub\_scrutiny\_prop  
 WHERE application\_no = :correct\_value.appln\_no;  
 ```  
 - This query retrieves the proposal type based on the application number and handles exceptions by setting a default value if any error occurs.

# Nationality Selection for Insured Person

Type: CORRECT\_VALUE

Title: Nationality Selection for Insured Person  
  
Acceptance Criteria:  
1. The nationality field should display a list of predefined options for the user to select from.  
2. The list should include the following options: "IP", "SON", "5", "8", "10", "1", "SL".  
3. The field should not be mandatory, allowing users to leave it blank if necessary.  
4. The field should be displayed on the "INSURED\_PERSON" tab page.  
5. The field should have a label "Nationality" aligned to the right of the field.  
6. The field should be editable, allowing users to update the nationality if needed.  
7. The field should be visually distinct with a white background and black text.  
  
Definition of Done:  
1. The nationality field is implemented and displays the predefined list of options.  
2. Users can select and update the nationality from the list.  
3. The field is correctly positioned on the "INSURED\_PERSON" tab page with the label "Nationality" aligned to the right.  
4. The field is not mandatory and can be left blank.  
5. The field's appearance matches the specified design (white background, black text).  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include specific database queries.

# Country of Residence Dropdown for Insured Person

Type: CORRECT\_VALUE

Title: Country of Residence Dropdown for Insured Person  
  
Acceptance Criteria:  
1. The country of residence field should display a dropdown list with predefined values.  
2. The dropdown list should include the following values: "IP", "SON", "5", "8", "10", "1", "SL".  
3. The field should not be mandatory, allowing users to leave it blank if necessary.  
4. The field should be editable, allowing users to change the selected value if needed.  
5. The field should be displayed on the "INSURED\_PERSON" tab page.  
6. The field should have a prompt labeled "Resi. Country" aligned to the right of the field.  
  
Definition of Done:  
1. The country of residence field is implemented and displays a dropdown list with the specified values.  
2. The field is editable and not mandatory.  
3. The field is correctly positioned on the "INSURED\_PERSON" tab page with the prompt "Resi. Country" aligned to the right.  
4. The functionality is tested and verified to ensure it meets the acceptance criteria.  
5. The feature is reviewed and approved by the stakeholders.

# Next Tab Navigation and Validation

Type: CORRECT\_VALUE

Detailed description: As a user, I want to navigate to the next tab in the application and ensure that all necessary validations and checks are performed before proceeding. This includes verifying agent bypass status, product group, and signature decisions.  
  
Acceptance criteria:  
1. When the "Next Tab" button is pressed, the system should:  
 - Check if the agent has a bypass flag set to 'Y' in the `azbj\_sign\_bypass\_agents` table.  
 - Verify if the product code belongs to a specific group using the `azbj\_pk\_nb\_general.group\_product` function.  
 - If the agent does not have a bypass flag and the product code is not in the specified group, further checks should be performed:  
 - If the agent code does not start with '3', ensure that the signature decision is not null. If it is null, prompt the user to match the IC signature and take a signature decision.  
 - Enable or disable the 'INSURED\_PERSON' tab based on the checks.  
 - Navigate to the 'IP\_TITLE' field in the 'CORRECT\_VALUE' block.  
  
Definition of Done:  
- The "Next Tab" button performs all the necessary checks and validations as described.  
- Appropriate messages are displayed to the user if any validation fails.  
- The system navigates to the 'IP\_TITLE' field after all checks are completed.  
- The 'INSURED\_PERSON' tab is enabled or disabled based on the validation results.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Query to check agent bypass status:  
 ```sql  
 SELECT COUNT(1)  
 INTO v\_agnet\_sign\_bypass  
 FROM azbj\_sign\_bypass\_agents  
 WHERE agent\_code = :correct\_value.fsc\_code AND by\_pass\_flag = 'Y';  
 ```  
  
- Function call to check product group:  
 ```sql  
 azbj\_pk\_nb\_general.group\_product(:correct\_value.product\_code)  
 ```

# Title Dropdown for Insured Person

Type: CORRECT\_VALUE

Detailed description: As a user, I want to be able to select a title from a predefined list when entering information about an insured person, so that the data entry process is streamlined and consistent.  
  
Acceptance criteria:  
1. The title field should be a dropdown list containing the following options:  
 - IP  
 - SON  
 - 5  
 - 8  
 - SL  
2. The dropdown list should be displayed on the "Insured Person" tab.  
3. The title field should be case-insensitive and automatically convert any input to uppercase.  
4. The title field should allow both insertion and updating of values.  
5. The title field should be visually styled with a white background, black text, and a specific font and size.  
6. The prompt for the title field should be "Title" and should be right-justified.  
7. The prompt should be bold and displayed for all records.  
  
Definition of Done:  
- The title dropdown list is implemented and contains the specified options.  
- The title field is functional on the "Insured Person" tab.  
- The title field automatically converts input to uppercase.  
- The title field allows insertion and updating of values.  
- The title field and its prompt are styled as specified.  
- The prompt for the title field is correctly displayed and justified.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include specific database queries.

# Validate Age Proof ID for Pan Card

Type: CORRECT\_VALUE

Title: Validate Age Proof ID for Pan Card  
  
Acceptance Criteria:  
1. When the age proof type is 'Pan Card' and the age proof ID is not null:  
 - The system should navigate to the PAN validation control.  
 - The system should execute the PAN validation trigger.  
 - The system should return to the age proof field after validation.  
  
Definition of Done:  
1. The system correctly navigates to the PAN validation control when the age proof type is 'Pan Card' and the age proof ID is not null.  
2. The PAN validation trigger is executed successfully.  
3. The system returns to the age proof field after the validation process is complete.  
4. All validations and navigations are tested and verified to work as expected.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- No direct database queries are provided in the XML content for CRUD operations.

# Gender Selection Dropdown

Type: CORRECT\_VALUE

Title: Gender Selection Dropdown  
  
Acceptance Criteria:  
1. The gender field should be a dropdown list.  
2. The dropdown list should contain the following values:  
 - "M" for Male  
 - "F" for Female  
 - "O" for Other  
3. The default value of the gender field should be "M".  
4. The gender field should be case-insensitive and accept only uppercase values.  
5. The gender field should be linked to the database column "sex".  
  
Definition of Done:  
1. The gender dropdown list is implemented and displays the correct values.  
2. The default value is set to "M".  
3. The gender field is case-insensitive and only accepts uppercase values.  
4. The gender information is correctly saved to and retrieved from the database column "sex".  
5. The feature is tested and verified to work as expected.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- No specific DB queries provided in the XML content.

# Select/Deselect All Riders Using Checkbox

Type: CORRECT\_VALUE

Title: Select/Deselect All Riders Using Checkbox  
  
Acceptance Criteria:  
1. When the "All Riders" checkbox is checked, all riders with a cover code starting with 'R' should be selected.  
2. When the "All Riders" checkbox is unchecked, all riders with a cover code starting with 'R' should be deselected.  
3. The system should navigate to the rider details section and iterate through all rider records to apply the selection or deselection.  
4. The total number of riders should be updated and stored in a global variable.  
5. The system should navigate to the delete item in the rider details section after processing the checkbox change.  
  
Definition of Done:  
- The "All Riders" checkbox functionality is implemented and tested.  
- The checkbox correctly selects or deselects all applicable riders.  
- The total number of riders is accurately updated.  
- The system navigates to the appropriate item after processing the checkbox change.  
- 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.

# Auto-populate Pin Code, State, and District for India

Type: CORRECT\_VALUE

Title: Auto-populate Pin Code, State, and District for India  
  
Acceptance Criteria:  
1. When the country name is set to 'India' and the pin code field is empty, the system should automatically query the database to retrieve distinct pin codes, states, and districts.  
2. The retrieved data should be displayed in a list of values (LOV) for the user to select from.  
3. The pin code field should be validated against the list of values to ensure that only valid pin codes are entered.  
  
Definition of Done:  
1. The system correctly queries the database for distinct pin codes, states, and districts when the country name is 'India' and the pin code field is empty.  
2. The list of values (LOV) is displayed to the user, allowing them to select a valid pin code.  
3. The pin code field is validated against the list of values, ensuring that only valid pin codes are accepted.  
4. The functionality is tested and verified to work as expected.  
  
DB queries for Table reference CRUD operations only (With Usage):  
```sql  
SELECT DISTINCT pincode, state, district   
FROM azbj\_state\_district\_pincode;  
```  
This query is used to populate the list of values (LOV) with distinct pin codes, states, and districts when the country name is 'India' and the pin code field is empty.

# Dynamic Management of Marital Status Field

Type: CORRECT\_VALUE

Detailed description: As a user, I want the system to manage the "Marital Status" field in a way that dynamically enables or disables related fields based on the selected marital status, ensuring that the data entry process is intuitive and accurate.  
  
Acceptance criteria:  
1. When the "Marital Status" is set to 'M' (Married):  
 - The fields for "Father" and "Spouse" should be enabled.  
 - The system should navigate to the "Spouse Date of Birth" field if it is not already enabled.  
2. When the "Marital Status" is set to 'W' (Widowed):  
 - The "Spouse" field should be enabled.  
3. When the "Marital Status" is set to any other value:  
 - The system should apply a default visual attribute to the current item.  
4. The system should validate the combination of "Title", "Gender", and "Marital Status" to ensure they are appropriate:  
 - If the title is 'MRS', gender is 'F', and marital status is 'S' (Single), an error message should be displayed.  
 - If the title is 'MISS', gender is 'F', and marital status is 'M' (Married), an error message should be displayed.  
  
Definition of Done:  
- The "Marital Status" field dynamically enables or disables related fields based on the selected value.  
- Appropriate error messages are displayed for invalid combinations of "Title", "Gender", and "Marital Status".  
- The system navigates to the correct fields as specified.  
- The visual attributes are applied correctly based on the conditions.  
- All changes are tested and verified to ensure they meet the acceptance criteria.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- No specific DB queries are provided in the XML content for CRUD operations.

# Input and Validate Third Line of Address

Type: CORRECT\_VALUE

Title: Input and Validate Third Line of Address  
  
Acceptance Criteria:  
1. When the user double-clicks on the input field for the third line of the address, a dropdown list should appear, allowing the user to select a value.  
2. The input field should validate the selected value against the predefined list to ensure consistency.  
3. The input field should allow the user to insert and update values.  
4. The input field should display a prompt "Place" aligned to the right of the field.  
  
Definition of Done:  
1. The input field for the third line of the address is implemented and visible on the form.  
2. The dropdown list appears upon double-clicking the input field.  
3. The input field validates the selected value against the predefined list.  
4. The input field allows for insertion and updating of values.  
5. The prompt "Place" is displayed and aligned correctly.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Not applicable as the provided XML content does not include specific database queries.

# State Selection for Address Information

Type: CORRECT\_VALUE

Title: State Selection for Address Information  
  
Acceptance Criteria:  
1. When the user is entering the address information and the country is set to "India", a list of states should be available for selection.  
2. The list of states should be fetched from the database and displayed in alphabetical order.  
3. The list should include both the state name and the state code.  
  
Definition of Done:  
1. The state selection functionality is implemented and available only when the country is "India".  
2. The list of states is correctly fetched from the database and displayed in alphabetical order.  
3. The user can select a state from the list, and the selected state is correctly saved in the address information.  
4. The functionality is tested and verified to work as expected.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Fetching the list of states:  
 ```sql  
 SELECT state\_name, scode FROM azbj\_states ORDER BY state\_name;  
 ```  
  
- Fetching the list of states for a specific LOV:  
 ```sql  
 SELECT state\_name FROM azbj\_states ORDER BY state\_name;  
 ```

# Implement Age Proof Field

Type: CORRECT\_VALUE

Title: Implement Age Proof Field  
  
Acceptance Criteria:  
1. The age proof field should be a list item allowing the user to select from predefined values.  
2. The field should be positioned appropriately on the form and should be labeled "IP Age Proof".  
3. The field should allow both insertion and updating of values.  
4. The field should navigate to the next item "IP\_DOOR\_NO" upon completion and should navigate from the previous item "IP\_BIRTHPLACE".  
5. The field should be displayed on the "AML\_KYC" tab page.  
6. The list of values for the age proof should include options such as "IP", "5", "8", "10", "N", "1", and "SL".  
  
Definition of Done:  
- The age proof field is implemented as a list item in the form.  
- The field is labeled "IP Age Proof" and is positioned correctly.  
- The field allows for insertion and updating of values.  
- The field navigates correctly to the next and previous items.  
- The field is displayed on the "AML\_KYC" tab page.  
- The list of values for the age proof is populated with the specified options.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include specific database queries.

# Display List of Values for City/District Field Based on Country

Type: CORRECT\_VALUE

Title: Display List of Values for City/District Field Based on Country  
  
Acceptance Criteria:  
1. When the user navigates to the "City/District" field and the country is set to "India", a list of values should be displayed.  
2. The list of values should be relevant to cities and districts in India.  
3. The "City/District" field should be editable and allow the user to select a value from the list.  
  
Definition of Done:  
1. The functionality to display a list of values for the "City/District" field when the country is "India" is implemented.  
2. The list of values is verified to contain relevant cities and districts in India.  
3. The "City/District" field is tested to ensure it is editable and allows selection from the list.  
4. The feature is tested and verified to work correctly in the user interface.  
5. Documentation is updated to reflect the new functionality.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- No specific DB queries provided in the XML content.

# Manage Residence Proof Information for Insured Persons

Type: CORRECT\_VALUE

Title: Manage Residence Proof Information for Insured Persons  
  
Acceptance Criteria:  
1. The residence proof information should be displayed in a list format.  
2. The list should include predefined values such as "IP" and "SON".  
3. The list should allow for the selection of multiple values.  
4. The list should be editable, allowing users to update the residence proof information as needed.  
5. The list should be visible only when required, based on specific conditions or user actions.  
  
Definition of Done:  
1. The residence proof information is displayed in a list format within the proposal form.  
2. The list includes predefined values such as "IP" and "SON".  
3. Users can select multiple values from the list.  
4. Users can update the residence proof information.  
5. The list visibility is controlled based on specific conditions or user actions.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include specific database queries.

# Auto-populate Address Fields for India

Type: CORRECT\_VALUE

Title: Auto-populate Address Fields for India  
  
Acceptance Criteria:  
1. When the user double-clicks on the "Landmark/Area" field and the country is set to India:  
 - The system should populate the postal code, state, and district fields based on the values in the corresponding address fields.  
 - If the country is not India, these fields should be cleared.  
 - The "Landmark/Area" field should be validated against a predefined list of values.  
  
2. When the user navigates to the "Landmark/Area" field and the country is set to India:  
 - The system should populate the postal code, state, and district fields based on the values in the corresponding address fields.  
 - The "Landmark/Area" field should be validated against a predefined list of values.  
  
Definition of Done:  
- The system correctly populates and clears the postal code, state, and district fields based on the country value.  
- The "Landmark/Area" field is validated against a predefined list of values.  
- The functionality is tested and verified to work as expected.  
  
DB queries for Table reference CRUD operations only (With Usage):  
```sql  
SELECT DISTINCT Area, PINCODE   
FROM (  
 SELECT DISTINCT Area, PINCODE  
 FROM azbj\_state\_district\_pincode   
 WHERE UPPER(STATE) = UPPER(:parameter.v\_state)   
 AND UPPER(DISTRICT) = UPPER(:parameter.v\_district)  
 AND pincode = NVL(:parameter.V\_POSTCODE, pincode)  
)  
ORDER BY pincode, Area;  
```  
  
This query is used to fetch distinct areas and pincodes based on the state and district parameters, and optionally the postal code, to validate the "Landmark/Area" field.

# Add Area/Pincode Functionality

Type: CORRECT\_VALUE

Title: Add Area/Pincode Functionality  
  
Acceptance Criteria:  
1. When the "ADD AREA/PINCODE" button is pressed, the system should open a new form or window to allow the user to input the area or pincode details.  
2. The new form or window should not hide the current form.  
3. The new form or window should replace any existing data in the input fields.  
4. The new form or window should not perform any query operations when opened.  
  
Definition of Done:  
1. The "ADD AREA/PINCODE" button is visible and enabled on the form.  
2. Pressing the button successfully opens a new form or window for area or pincode input.  
3. The new form or window adheres to the specified acceptance criteria.  
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):  
- Not applicable as the provided XML content does not include any specific database queries.

# User selects preferred communication language

Type: CORRECT\_VALUE

Detailed description: As a user, I want to be able to select the preferred communication language 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 see a dropdown list of available languages when selecting the preferred communication language.  
2. The list of available languages should be fetched from the database and ordered by language ID.  
3. The selected language should be stored in the system and should be editable.  
4. The system should validate that the selected language is from the predefined list.  
5. Upon selecting a language, the system should automatically navigate to the next item in the form.  
  
Definition of Done:  
1. The dropdown list of languages is displayed correctly.  
2. The list of languages is fetched from the database and ordered by language ID.  
3. The selected language is stored and can be updated.  
4. Validation ensures that only languages from the predefined list can be selected.  
5. The form navigates to the next item after a language is selected.  
  
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 from the database, ordered by language ID.

# Input and Update City/District Information

Type: CORRECT\_VALUE

Title: Input and Update City/District Information  
  
Acceptance Criteria:  
1. The field should allow the user to input and update the City/District information.  
2. The input should be automatically converted to uppercase.  
3. The field should have a maximum length of 40 characters.  
4. The field should be visually distinct with a white background and black text.  
5. The field should be positioned correctly on the form as per the design specifications.  
6. The field should display a prompt labeled "City/District" aligned to the right of the field.  
7. The field should be part of the "INSURED\_PERSON" tab page.  
8. When the field is focused, it should trigger a list of values for selection.  
  
Definition of Done:  
- The field for City/District information is implemented and functional.  
- The field adheres to the specified visual and functional requirements.  
- The field is tested to ensure it converts input to uppercase and restricts input to 40 characters.  
- The prompt "City/District" is correctly displayed and aligned.  
- The list of values is triggered correctly when the field is focused.  
- All acceptance criteria are met and verified through testing.

# Manage Area (Free Text) Field Based on IP Area Status

Type: CORRECT\_VALUE

Detailed description: As a user, I want the system to manage the "Area (Free Text)" field based on the status of the "IP Area" field and retrieve the latest request status for the application number, so that I can ensure the correct handling of the area information.  
  
Acceptance criteria:  
1. When the "IP Area" field has the value "NOT FOUND":  
 - The "Area (Free Text)" field should be enabled for both insertion and updating.  
 - The system should retrieve the latest request status for the application number from the `azbj\_area\_pincode\_request` table where the address type is 'IP' and the area free text is not null.  
 - The retrieved status should be displayed in the format: "The Area Free Text for [area\_free\_text] Status is [status]", where the status is decoded as 'Approved' for 'A', 'Pending' for 'P', and 'Rejected' for 'R'.  
 - If any error occurs during the retrieval, the status should be set to null.  
  
2. When the "IP Area" field does not have the value "NOT FOUND":  
 - The "Area (Free Text)" field should be disabled for both insertion and updating.  
  
Definition of Done:  
- The "Area (Free Text)" field behavior is correctly managed based on the "IP Area" field value.  
- The latest request status is accurately retrieved and displayed when applicable.  
- The system handles errors gracefully by setting the status to null if retrieval fails.  
- The functionality is tested and verified to work as expected.  
  
DB queries for Table reference CRUD operations only (With Usage):  
```sql  
SELECT 'The Area Free Text for "' || area\_free\_text || '" Status is ' || DECODE(request\_status, 'A', 'Approved', 'P', 'Pending', 'R', 'Rejected')  
INTO v\_free\_text  
FROM azbj\_area\_pincode\_request  
WHERE application\_no = :correct\_value.appln\_no  
 AND address\_type = 'IP'  
 AND area\_free\_text IS NOT NULL  
 AND REQUEST\_DATE = (SELECT MAX(REQUEST\_DATE)   
 FROM azbj\_area\_pincode\_request  
 WHERE application\_no = :correct\_value.appln\_no  
 AND address\_type = 'IP'  
 AND area\_free\_text IS NOT NULL);  
```

# Specify Delivery Place for Policy Dispatch

Type: CORRECT\_VALUE

Title: Specify Delivery Place for Policy Dispatch  
  
Acceptance Criteria:  
1. The input field for specifying the delivery place should accept up to 4 characters.  
2. The input should be automatically converted to uppercase.  
3. The field should have a tooltip and hint text that says, "Please specify the place where policy to be dispatched."  
4. The field should be labeled as "Del.My pol.at" and the label should be right-justified.  
5. The field should be editable and allow both insertion and updates.  
6. Upon pressing the key to move to the next item, the focus should shift to the next tab item.  
  
Definition of Done:  
1. The input field for the delivery place is implemented and visible on the form.  
2. The field accepts up to 4 characters and converts input to uppercase.  
3. Tooltip and hint text are displayed correctly.  
4. The label "Del.My pol.at" is right-justified and displayed correctly.  
5. The field allows insertion and updates.  
6. The focus shifts to the next tab item when the key to move to the next item is pressed.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Not applicable as the provided XML content does not include any specific database queries.

# Residence Proof Type Selection for Insured Person

Type: CORRECT\_VALUE

Detailed description: As a user, I want to be able to select a residence proof type from a predefined list when entering information about an insured person, so that I can accurately record the type of residence proof provided.  
  
Acceptance criteria:  
1. The residence proof type should be selectable from a dropdown list.  
2. The dropdown list should include the following options:  
 - IP  
 - SON  
 - 5  
 - 8  
 - SL  
3. The dropdown list should be associated with the insured person section.  
4. The selected residence proof type should be saved and retrievable for future reference.  
  
Definition of Done:  
1. The dropdown list for residence proof type is implemented and visible in the insured person section.  
2. The dropdown list contains the specified options.  
3. Users can select an option from the dropdown list.  
4. The selected option is saved correctly and can be retrieved when needed.  
5. 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 database queries.

# Dropdown Selection for Address Line 3

Type: CORRECT\_VALUE

Title: Dropdown Selection for Address Line 3  
  
Acceptance Criteria:  
1. When the user double-clicks on the address line 3 field, a dropdown list of places should appear.  
2. The dropdown list should be populated with unique places based on the district entered in address line 4.  
3. If the district entered does not match any records, the dropdown should include an option labeled 'NOT FOUND'.  
4. The user should be prompted to select a place from the dropdown list, even if the spelling is different but the pronunciation is the same.  
5. The address line 3 field should be validated against the list of places to ensure a valid selection.  
  
Definition of Done:  
1. The dropdown list for address line 3 is functional and displays unique places based on the district entered in address line 4.  
2. The 'NOT FOUND' option is included in the dropdown if no matching records are found.  
3. The user receives a prompt to select a place from the dropdown list.  
4. The address line 3 field is validated to ensure a valid selection from the dropdown list.  
5. All acceptance criteria are met and tested.  
  
DB queries for Table reference CRUD operations only (With Usage):  
```sql  
SELECT UNIQUE state, district, place  
FROM azbj\_urban\_rural\_data   
WHERE UPPER(district) = UPPER(:CORRECT\_VALUE.IP\_M\_ADDRESS\_LINE4)  
UNION  
SELECT null, null, 'NOT FOUND'  
FROM dual  
ORDER BY place;  
```

# Validate and Populate Pin Code for Insured Person

Type: CORRECT\_VALUE

Title: Validate and Populate Pin Code for Insured Person  
  
Acceptance Criteria:  
1. When the user navigates to the pin code input field, if the country name is 'INDIA', the system should automatically populate a list of valid pin codes, states, and districts from the database.  
2. The pin code input field should validate the entered pin code against the populated list to ensure it is valid.  
3. On double-clicking the pin code input field, the system should again populate the list of valid pin codes, states, and districts from the database.  
  
Definition of Done:  
1. The pin code input field is functional and allows users to input a pin code.  
2. The system correctly populates and validates the pin code against the list of valid pin codes, states, and districts when the user navigates to or double-clicks the pin code input field.  
3. The feature is tested and verified to work as expected.  
  
SQL Query for Reference:  
- SELECT DISTINCT pincode, state, district FROM azbj\_state\_district\_pincode (Used to populate the list of valid pin codes, states, and districts).

# Consent for Communication via Mail/SMS

Type: CORRECT\_VALUE

Title: Consent for Communication via Mail/SMS  
  
Acceptance Criteria:  
1. The consent option should be presented as a checkbox labeled "Consent for sending comm. on Mail/SMS".  
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 is checked, it should store the value 'Y'.  
5. When the checkbox is unchecked, it should store the value 'N'.  
6. The checkbox should be located on the "INSURED\_PERSON" tab.  
7. The tooltip for the checkbox should read "Please specify the place where policy to be dispatched."  
8. The hint for the checkbox should read "Please specify the place where policy to be dispatched."  
9. The checkbox should allow both insertion and updates.  
10. Upon pressing the "Next" key, the focus should move to the next item in the form.  
  
Definition of Done:  
- The checkbox for consent is implemented and visible on the "INSURED\_PERSON" tab.  
- The checkbox correctly stores 'Y' when checked and 'N' when unchecked.  
- The default state of the checkbox is unchecked.  
- The tooltip and hint are correctly displayed.  
- The checkbox allows insertion and updates.  
- The focus moves to the next item when the "Next" key is pressed.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include specific database queries.

# Navigation based on Marital Status for Father's Name Input

Type: CORRECT\_VALUE

Title: Navigation based on Marital Status for Father's Name Input  
  
Acceptance Criteria:  
1. When the user inputs the father's name and the marital status is 'S' (Single), the system should automatically navigate to the field for entering the Personal Identification Number (PIN).  
2. When the user inputs the father's name and the marital status is not 'S' (Single), the system should automatically navigate to the field for entering the spouse's name.  
  
Definition of Done:  
- The system correctly navigates to the PIN field if the marital status is 'S'.  
- The system correctly navigates to the spouse's name field if the marital status is not 'S'.  
- The navigation logic is tested and verified to ensure it works as expected in all scenarios.  
  
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.

# Add Area/Pincode Button Functionality

Type: CORRECT\_VALUE

Title: Add Area/Pincode Button Functionality  
  
Acceptance Criteria:  
1. When the "ADD AREA/PINCODE" button is pressed, a new form should be called to allow the user to input the area or pincode.  
2. The new form should open without hiding the current form.  
3. The new form should replace the current form's content.  
4. The new form should not perform any query operations upon opening.  
  
Definition of Done:  
- The "ADD AREA/PINCODE" button is visible and enabled on the insured person's tab.  
- Pressing the button successfully opens a new form for area or pincode input.  
- The current form remains visible and is replaced by the new form's content.  
- No query operations are performed when the new form is opened.  
  
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 Populate Area/Landmark Information

Type: CORRECT\_VALUE

Title: Validate and Populate Area/Landmark Information  
  
Acceptance Criteria:  
1. When the user double-clicks on the area/landmark field, if the country is 'INDIA', the system should:  
 - Set the postcode, state, and district parameters based on the corresponding fields.  
 - Display a list of valid area values for selection.  
 - Enable validation from the list for the area field.  
  
2. When the user navigates to the area/landmark field, the system should:  
 - Check if the country is 'INDIA'. If true:  
 - Set the postcode, state, and district parameters based on the corresponding fields.  
 - Populate a list of valid area values from the database where the state and district match the provided values.  
 - Enable validation from the list for the area field.  
 - If the country is not 'INDIA':  
 - Clear the list of valid area values.  
 - Disable validation from the list for the area field.  
  
Definition of Done:  
- The area/landmark field should correctly display and validate values based on the country, state, and district information.  
- The system should dynamically populate the list of valid area values from the database when the country is 'INDIA'.  
- The validation logic should be correctly applied based on the country information.  
  
DB queries for Table reference CRUD operations only(With Usage):  
```sql  
SELECT DISTINCT Area, PINCODE   
FROM azbj\_state\_district\_pincode   
WHERE UPPER(STATE) = UPPER(:parameter.v\_state)   
AND UPPER(DISTRICT) = UPPER(:parameter.v\_district)   
AND PINCODE = NVL(:parameter.V\_POSTCODE, PINCODE)   
ORDER BY PINCODE, Area;  
```  
  
This query is used to populate the list of valid area values when the country is 'INDIA'.

# Implement Education Level Dropdown

Type: CORRECT\_VALUE

Title: Implement Education Level Dropdown  
  
Acceptance Criteria:  
1. The education level field should be displayed as a dropdown list.  
2. The dropdown list should contain the following options:  
 - IP  
 - 5  
 - 8  
 - SL  
 - EDUCATION  
3. The field should be editable, allowing users to select an option from the list.  
4. The field should be positioned correctly on the form, with the label "Education" aligned to the right of the field.  
5. The field should be displayed on the "IP\_INFO" tab page.  
6. The field should have a white background and black text.  
7. The field should be 95 units wide and 15 units high.  
8. The label "Education" should be bold and in "MS Sans Serif" font.  
  
Definition of Done:  
- The education level field is implemented as a dropdown list with the specified options.  
- The field is editable and positioned correctly on the form.  
- The field and its label meet the specified design and formatting requirements.  
- The field is displayed on the correct tab page.  
- The implementation is tested and verified to ensure it meets all acceptance criteria.

# Dynamic Control of 'Area (Free Text)' Field Based on 'IP\_M\_AREA' Status

Type: CORRECT\_VALUE

Detailed description: As a user, I want the system to manage the "Area (Free Text)" field based on the status of the "IP\_M\_AREA" field and the latest request status from the "azbj\_area\_pincode\_request" table, so that the field's insert and update permissions are dynamically controlled and relevant status messages are displayed.  
  
Acceptance criteria:  
1. When the "IP\_M\_AREA" field has the value 'NOT FOUND':  
 - The "Area (Free Text)" field should be enabled for both insert and update operations.  
 - The system should fetch the latest request status for the corresponding application number and address type 'IP\_M' from the "azbj\_area\_pincode\_request" table.  
 - If a matching record is found, a message should be displayed indicating the status of the "Area Free Text".  
 - If no matching record is found, no message should be displayed.  
  
2. When the "IP\_M\_AREA" field does not have the value 'NOT FOUND':  
 - The "Area (Free Text)" field should be disabled for both insert and update operations.  
  
Definition of Done:  
- The "Area (Free Text)" field's insert and update permissions are correctly controlled based on the value of the "IP\_M\_AREA" field.  
- The system correctly fetches and displays the latest request status message when applicable.  
- 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):  
```sql  
SELECT 'The Area Free Text for "' || area\_free\_text || '" Status is ' || DECODE(request\_status, 'A', 'Approved', 'P', 'Pending', 'R', 'Rejected')  
INTO v\_free\_text  
FROM azbj\_area\_pincode\_request  
WHERE application\_no = :correct\_value.appln\_no  
 AND address\_type = 'IP\_M'  
 AND area\_free\_text IS NOT NULL  
 AND REQUEST\_DATE = (  
 SELECT MAX(REQUEST\_DATE)  
 FROM azbj\_area\_pincode\_request  
 WHERE application\_no = :correct\_value.appln\_no  
 AND address\_type = 'IP\_M'  
 AND area\_free\_text IS NOT NULL  
 );  
```

# Manage PAN Information

Type: CORRECT\_VALUE

Title: Manage PAN Information  
  
Acceptance Criteria:  
1. The PAN field should accept a maximum of 10 characters.  
2. The PAN field should automatically convert input to uppercase.  
3. The PAN field should allow for both insertion and updates.  
4. Upon entering the PAN, the system should navigate to the IP Type field.  
  
Definition of Done:  
1. The PAN field is implemented with a maximum length of 10 characters.  
2. The PAN field converts all input to uppercase.  
3. The PAN field allows users to insert new PANs and update existing ones.  
4. The system successfully navigates to the IP Type field after the PAN is entered.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include specific database queries.

# Profession Selection from Predefined List

Type: CORRECT\_VALUE

Title: Profession Selection from Predefined List  
  
Acceptance Criteria:  
1. The profession field should allow the user to select a value from a predefined list of professions.  
2. The list of professions should be fetched dynamically based on the industry code provided.  
3. The list should only include professions that are currently active (i.e., where the end date is null).  
4. The list should be sorted alphabetically by the profession description.  
5. When the user double-clicks on the profession field, the list of values should be displayed.  
  
Definition of Done:  
- The profession field is implemented and allows selection from a predefined list.  
- The list of professions is dynamically fetched based on the industry code.  
- Only active professions are included in the list.  
- The list is sorted alphabetically by profession description.  
- Double-clicking the profession field displays the list of values.  
  
DB queries for Table reference CRUD operations only(With Usage):  
```sql  
SELECT a.occ\_code, a.occ\_desc   
FROM azbj\_occupation a, azbj\_ind\_occ\_link b   
WHERE a.occ\_code = b.occ\_code   
AND b.ind\_code = TO\_NUMBER(:correct\_value.IP\_INDUSTRY)  
AND end\_date IS NULL  
ORDER BY 2;  
```

# Implement IT Assessee Status Dropdown

Type: CORRECT\_VALUE

Title: Implement IT Assessee Status Dropdown  
  
Acceptance Criteria:  
1. The user should be able to select "YES" or "NO" from a dropdown list for the IT Assessee status.  
2. The dropdown list should display "YES" with a value of "Y" and "NO" with a value of "N".  
3. The selected value should be saved and updated in the database when the record is saved or updated.  
4. The IT Assessee status field should be visible only when required and should be hidden otherwise.  
  
Definition of Done:  
1. The dropdown list for IT Assessee status is implemented and displays the correct options.  
2. The selected value is correctly saved and updated in the database.  
3. The visibility of the IT Assessee status field is controlled based on the specified conditions.  
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 table references.

# Implement Dispatch Mode Field

Type: CORRECT\_VALUE

Title: Implement Dispatch Mode Field  
  
Acceptance Criteria:  
1. The dispatch mode field should display a list of predefined options including "SON", "5", "8", "10", "1", "SL", and "Other".  
2. The dispatch mode field should be case-insensitive and accept only uppercase input.  
3. The dispatch mode field should be visible on the "INSURED\_PERSON" tab page.  
4. The dispatch mode field should allow both insertion and updating of values.  
5. The dispatch mode field should have a maximum length of 10 characters.  
6. The dispatch mode field should be initialized with the value "Courier".  
  
Definition of Done:  
1. The dispatch mode field is implemented and displays the predefined list of options.  
2. The dispatch mode field accepts only uppercase input and has a maximum length of 10 characters.  
3. The dispatch mode field is visible on the "INSURED\_PERSON" tab page.  
4. The dispatch mode field allows insertion and updating of values.  
5. The dispatch mode field is initialized with the value "Courier".  
6. All acceptance criteria are met and verified through testing.

# Validate and Display Industry Description

Type: CORRECT\_VALUE

Title: Validate and Display Industry Description  
  
Acceptance Criteria:  
1. When the user double-clicks on the industry description field, a list of valid industry values should be displayed.  
2. Upon validating the industry description field:  
 - The system should count the number of relevant records in the `azbj\_strip\_questionnaire\_det` table where `appln\_no` matches the current application number, `question\_id` is 85, and `sub\_question` is one of 52, 53, 54, 55, or 56.  
 - If the `fsc\_code` is '3000000007' or the in-house flag is 'Y' or the web aggregate flag is 'Y', and the count of relevant records is greater than 0:  
 - If the industry description is 'CRPF' or 'DEFENCE', the additional question field `CRPF\_QUES\_IP` should be made visible and enabled.  
 - If the industry description is not 'CRPF' or 'DEFENCE', the additional question field `CRPF\_QUES\_IP` should be hidden and disabled.  
 - If the conditions are not met, the additional question field `CRPF\_QUES\_IP` should be hidden and disabled.  
  
Definition of Done:  
- The industry description field should display a list of valid values upon double-click.  
- The validation logic should correctly count relevant records and display or hide the additional question field based on the specified conditions.  
- The system should log debug information at various points in the validation process for troubleshooting purposes.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- `SELECT COUNT(1) INTO v\_cnt FROM azbj\_strip\_questionnaire\_det WHERE appln\_no = :correct\_value.appln\_no AND question\_id = 85 AND sub\_question IN (52, 53, 54, 55, 56);`  
- `SELECT ind\_code, ind\_desc FROM azbj\_industry WHERE end\_date IS NULL ORDER BY 2;`

# Input and Update Income Proof Information

Type: CORRECT\_VALUE

Title: Input and Update Income Proof Information  
  
Acceptance Criteria:  
1. The input field for income proof should be clearly labeled as "Income Proof".  
2. The input field should accept only uppercase characters.  
3. The input field should be a list item, allowing the user to select from predefined options.  
4. The predefined options for the income proof list should include:  
 - "IP" with a value of "IP"  
 - "5" with a value of "5"  
 - "8" with a value of "8"  
 - "No" with a value of "SL"  
5. The input field should be editable, allowing users to insert and update information.  
6. The input field should be positioned at coordinates (245, 42) on the screen.  
7. The input field should have a width of 95 and a height of 15.  
8. The input field should be displayed on the "IP\_INFO" tab page.  
9. The input field should have a white background and black foreground color.  
10. The prompt for the input field should be right-justified and displayed with a bold font style.  
  
Definition of Done:  
- The input field for income proof is implemented as per the acceptance criteria.  
- The input field is tested to ensure it accepts only uppercase characters.  
- The predefined options for the list item are correctly displayed and selectable.  
- The input field is editable, allowing users to insert and update information.  
- The input field is correctly positioned and displayed on the "IP\_INFO" tab page.  
- The prompt for the input field is right-justified and displayed with a bold font style.  
- The feature is reviewed and approved by the stakeholders.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include specific database queries.

# User Selection for Politically Exposed Person (PEP) Status

Type: CORRECT\_VALUE

Title: User Selection for Politically Exposed Person (PEP) Status  
  
Acceptance Criteria:  
1. The field should allow the user to select from a list of predefined values.  
2. The list should include the following options:  
 - "IP" with a value of "IP"  
 - "5" with a value of "5"  
 - "8" with a value of "8"  
 - "LA" with a value of "1"  
 - "No" with a value of "SL"  
3. The field should be displayed on the "IP\_INFO" tab page.  
4. The field should be labeled as "Pol Exp Person".  
5. The field should only accept uppercase characters.  
6. The field should be editable and allow both insert and update operations.  
  
Definition of Done:  
- The field is implemented and displayed on the "IP\_INFO" tab page.  
- The field allows selection from the predefined list of values.  
- The field label "Pol Exp Person" is visible.  
- The field only accepts uppercase characters.  
- The field supports insert and update operations.  
- The functionality is tested and verified to meet the acceptance criteria.

# Manage PEP Details Input and Navigation

Type: CORRECT\_VALUE

Detailed description: As a user, I want to input and manage PEP (Politically Exposed Person) details within the system, ensuring that the data is stored accurately and can be navigated efficiently based on specific conditions.  
  
Acceptance criteria:  
1. When the user inputs PEP details, the system should allow the entry of up to 2000 characters.  
2. The input field for PEP details should be displayed with a gray background and black text.  
3. The PEP details field should be positioned correctly on the user interface and should be labeled as "PEP Details".  
4. The system should ensure that the PEP details are stored in uppercase.  
5. The user should be able to insert and update the PEP details.  
6. The system should navigate to the appropriate section based on the value of `pk\_vars.v\_grp\_product`:  
 - If `pk\_vars.v\_grp\_product` is 'N', the system should navigate to the 'AZBJ\_FAMILY\_DETAILS' section and display the first record.  
 - Otherwise, the system should navigate to the 'AZBJ\_UL\_CHECKER' section and set the next item to 'AZBJ\_UL\_CHECKER.CHK\_INSURED'.  
  
Definition of Done:  
- The PEP details input functionality is implemented and tested.  
- The navigation logic based on `pk\_vars.v\_grp\_product` is correctly implemented and tested.  
- The user interface displays the PEP details field as specified.  
- The system allows insertion and updating of PEP details.  
- 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 CRUD operations or SQL queries.

# Dynamic Update of Income Proof Types and Employer Names Based on Occupation and Product Code

Type: CORRECT\_VALUE

Detailed description: As a user, I want the system to dynamically update the list of income proof types and employer names based on the selected occupation and product code, so that I can ensure the correct information is displayed and editable according to the business rules.  
  
Acceptance criteria:  
1. When the occupation is changed, the system should clear the existing lists of income proof types and populate them with new values based on the selected occupation.  
2. The system should add 'Not Received' and 'No income proof for Cat 1 & Cat 2 available' as default options in the income proof lists.  
3. If the product code is either 297 or 315 and the occupation is 'SG' or 'SN':  
 - The employer name field should be made visible and enabled.  
 - The system should clear the existing list of employer names and populate it with values from the 'HEALTH\_CARE' system constants.  
4. If the product code is either 297 or 315 and the occupation is not 'SG' or 'SN':  
 - The employer name field should be hidden and disabled.  
 - The employer's name field should be enabled and cleared.  
  
Definition of Done:  
- The system dynamically updates the income proof types and employer names based on the selected occupation and product code.  
- The employer name field's visibility and enabled state are correctly toggled based on the occupation and product code.  
- The lists are populated with the correct values from the database.  
- The changes are tested and verified to ensure they meet the acceptance criteria.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- The system should execute the following query to fetch distinct income proof types and descriptions:  
 ```sql  
 SELECT DISTINCT a.proof\_type, b.SYS\_DESC proof\_desc  
 FROM AZBJ\_FEMALE\_UW\_CATEGORY a, azbj\_system\_constants b  
 WHERE a.sys\_type = 'FEMALE\_UW\_CATEGORY'  
 AND b.SYS\_TYPE = 'FR\_REQ'  
 AND a.INCOME\_PROOF\_CODE = b.sys\_code  
 AND a.OCCUPATION\_CODE = :CORRECT\_VALUE.IP\_OCCUPATION  
 UNION ALL  
 SELECT 'NR', 'Not Received' FROM dual  
 UNION ALL  
 SELECT 'NA', 'No income proof for Cat 1 & Cat 2 available' FROM dual;  
 ```  
  
- The system should execute the following query to fetch employer names for the 'HEALTH\_CARE' system constants:  
 ```sql  
 SELECT CHAR\_VALUE  
 FROM azbj\_system\_constants  
 WHERE sys\_type = 'HEALTH\_CARE'  
 AND sys\_code = 'TOP\_CORPORATES'  
 ORDER BY CHAR\_VALUE;  
 ```

# Dynamic Field Enable/Disable Based on Partner Type

Type: CORRECT\_VALUE

Title: Dynamic Field Enable/Disable Based on Partner Type  
  
Acceptance Criteria:  
1. When the partner type is set to 'I':  
 - The fields for name, middle name, date of birth, title, gender, birthplace, age proof, age proof ID, marital status, father's name, height, weight, and weight change should be disabled.  
 - The values for name, middle name, and date of birth should be cleared.  
2. When the partner type is set to any value other than 'I':  
 - The fields for title, name, middle name, date of birth, gender, nationality, birthplace, age proof, age proof ID, marital status, father's name, height, weight, and weight change should be enabled.  
3. The system should handle the transition between fields correctly, ensuring that the next item logic is applied based on the partner type.  
4. The system should set the rural/urban flag appropriately when the partner type changes.  
  
Definition of Done:  
- The form dynamically enables or disables fields based on the selected partner type.  
- The form clears specific fields when the partner type is 'I'.  
- The form correctly handles the transition to the next item based on the partner type.  
- The rural/urban flag is set correctly when the partner type changes.  
- All changes are tested and verified to ensure they meet the acceptance criteria.

# Next Tab Button Functionality

Type: CORRECT\_VALUE

Title: Next Tab Button Functionality  
  
Acceptance Criteria:  
1. When the "Next Tab" button is pressed:  
 - If the Aadhaar field is empty, the age proof type is either 'AC' or 'ACS', and the age proof ID is not empty, then the Aadhaar field should be auto-filled with the age proof ID.  
 - The system should query the `azbj\_state\_district\_pincode` table to retrieve the area based on the state, district, pincode, and area fields.  
 - If the area is not found and the area field is not 'NOT FOUND' or 'NOTFOUND', and the property type is not 'N', the system should navigate to the area field for correction.  
 - If the area is found or the area field is 'NOT FOUND' or 'NOTFOUND', the system should query the `azbj\_urban\_rural\_data` table to retrieve the place based on the address line 3 and district fields.  
 - If the address line 3 field is empty and not 'NOT FOUND' or 'NOTFOUND', the system should navigate to the address line 3 field for correction.  
 - If the address line 3 field is valid, the system should enable the 'IP\_INFO' tab and navigate to the education field.  
  
2. The system should log specific information for debugging purposes, including the application number, date of birth, and age.  
  
3. If the age is 5 or more, the system should clear the occupation list and populate it with values from the `inf\_dnm\_poplists` table where the poplist code is 'EMPLOYMENT\_STATUS' and the internal value is not 'I'.  
  
Definition of Done:  
- The "Next Tab" button functionality is implemented as per the acceptance criteria.  
- The system correctly auto-fills, validates, and navigates based on the specified conditions.  
- The necessary database queries are executed, and the results are handled appropriately.  
- Debugging information is logged as specified.  
- The occupation list is updated correctly based on the age condition.  
- The functionality is tested and verified to work as expected.  
  
DB queries for Table reference CRUD operations only(With Usage):  
```sql  
-- Query to retrieve area  
SELECT DISTINCT Area  
INTO v\_area  
FROM azbj\_state\_district\_pincode  
WHERE UPPER(STATE) = UPPER(:CORRECT\_VALUE.IP\_ADDRESS\_LINE5)  
 AND UPPER(DISTRICT) = UPPER(:CORRECT\_VALUE.IP\_ADDRESS\_LINE4)  
 AND pincode = NVL(:CORRECT\_VALUE.IP\_PIN, pincode)  
 AND UPPER(area) = UPPER(:CORRECT\_VALUE.IP\_AREA);  
  
-- Query to retrieve place  
SELECT UNIQUE place  
INTO v\_place  
FROM azbj\_urban\_rural\_data  
WHERE UPPER(place) = UPPER(:CORRECT\_VALUE.IP\_ADDRESS\_LINE3)  
 AND UPPER(district) = UPPER(:CORRECT\_VALUE.IP\_ADDRESS\_LINE4);  
  
-- Query to populate occupation list  
SELECT INTERNAL\_VALUE, SCREEN\_VALUE  
FROM inf\_dnm\_poplists  
WHERE poplist\_code = 'EMPLOYMENT\_STATUS'  
 AND internal\_value <> 'I';  
```

# Implement Title Dropdown for Policyholder

Type: CORRECT\_VALUE

Title: Implement Title Dropdown for Policyholder  
  
Acceptance Criteria:  
1. The title field should display a dropdown list with predefined options.  
2. The dropdown list should include the following options: "PH", "5", "8", and "SL".  
3. The title field should be positioned correctly on the form and should be visually aligned with other fields.  
4. The title field should be editable, allowing users to select an option from the dropdown list.  
5. The title field should be restricted to uppercase input.  
  
Definition of Done:  
1. The title field is implemented and displays a dropdown list with the specified options.  
2. The title field is editable and restricted to uppercase input.  
3. The title field is correctly positioned and aligned on the form.  
4. The form is tested to ensure that the title field works as expected and meets the acceptance criteria.  
5. The changes are reviewed and approved by the stakeholders.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include specific database queries.

# Gender Selection for Policyholder

Type: CORRECT\_VALUE

Title: Gender Selection for Policyholder  
  
Acceptance Criteria:  
1. The gender field should be a dropdown list with the following options:  
 - "PH"  
 - "5"  
 - "8"  
 - "10"  
 - "LA"  
 - "No"  
 - "Other"  
2. The default value of the gender field should be set to "M".  
3. The gender field should only accept uppercase letters.  
4. The gender field should be editable and allow updates.  
5. The gender field should be linked to the database column named "sex".  
  
Definition of Done:  
1. The gender dropdown list is implemented and displays the correct options.  
2. The default value of the gender field is set to "M".  
3. The gender field only accepts uppercase letters.  
4. The gender field is editable and allows updates.  
5. The gender field is correctly linked to the database column "sex".  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Not applicable as the provided XML content does not include specific SQL queries or CRUD operations.

# Manage Nominee Relationship Details

Type: CORRECT\_VALUE

Title: Manage Nominee Relationship Details  
  
Acceptance Criteria:  
1. The relationship details should be displayed in a list format.  
2. The list should include predefined values such as "IP", "8", "NO", "LA", and "SL".  
3. The relationship field should be restricted to uppercase input.  
4. The field should allow insertion and updates.  
5. The field should be displayed on the "NOMINEE\_HEALTH\_DTLS" tab page.  
6. The field should be visually styled with specific font and color settings.  
  
Definition of Done:  
1. The relationship details can be successfully added, viewed, and updated.  
2. The list of predefined values is correctly displayed and selectable.  
3. The field enforces uppercase input.  
4. The field is accessible on the specified tab page.  
5. The visual styling matches the specified requirements.  
  
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.

# Navigation to Next Tab on Button Press

Type: CORRECT\_VALUE

Title: Navigation to Next Tab on Button Press  
  
Acceptance Criteria:  
1. When the "Next Tab" button is pressed:  
 - If the insured checkbox is unchecked, enabled, and the spouse is not present:  
 - Navigate to the insured checkbox item and execute the associated trigger.  
 - If the product code is 315:  
 - Refresh member covers.  
 - Enable or disable the "FAMILY\_MEMBERS" tab.  
 - If the spouse is present, set the partner type to 'P' and navigate to the partner type item.  
 - Otherwise, navigate to the next tab for policy members.  
 - If the product code is not 315:  
 - If the age is 18 or older and the occupation is not 'T':  
 - Disable the "MLQ" tab if it is enabled.  
 - If the insured checkbox is unchecked:  
 - Disable the "POLICY\_HOLDER" and "PH\_INFO" tabs if they are enabled.  
 - Enable or disable the "NOMINEE\_HEALTH\_DETAILS" tab.  
 - If the age is 18 or older, navigate to the nominee name item.  
 - Otherwise, navigate to the questionnaire item.  
 - If the insured checkbox is checked:  
 - Enable the "POLICY\_HOLDER" and "PH\_INFO" tabs if they are disabled.  
 - Navigate to the policy holder title item.  
 - If the insured checkbox is checked and the product code is not 315:  
 - Enable the "POLICY\_HOLDER" and "PH\_INFO" tabs if they are disabled.  
 - Navigate to the policy holder title item.  
 - If the insured checkbox is unchecked and the product code is not 315:  
 - Disable the "POLICY\_HOLDER" and "PH\_INFO" tabs if they are enabled.  
 - Enable or disable the "MLQ" tab.  
 - Navigate to the father's income item.  
 - If the group product is 'Y', enable or disable the "NOMINEE\_HEALTH\_DETAILS" tab and navigate to the nominee name item.  
  
Definition of Done:  
- The "Next Tab" button should navigate to the appropriate tab or item based on the specified conditions.  
- All conditions and navigations should be tested and verified to ensure they work as expected.  
- The functionality should be independent of any specific technology or Oracle Forms terminology.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- No specific DB queries are mentioned in the provided XML content.

# Select and Update Age Proof Type

Type: CORRECT\_VALUE

Title: Select and Update Age Proof Type  
  
Acceptance Criteria:  
1. The age proof type should be selectable from a predefined list of values.  
2. The list of values should include options such as "PH", "5", "8", "10", "LA", and "No".  
3. The selected age proof type should be saved and updated in the system.  
4. The age proof type should be displayed in uppercase letters.  
5. The age proof type should be displayed on the "AML\_KYC" tab page.  
  
Definition of Done:  
1. The user can select an age proof type from the predefined list.  
2. The selected age proof type is saved and can be updated.  
3. The age proof type is displayed in uppercase on the "AML\_KYC" tab page.  
4. The functionality is tested and verified to work as expected.  
5. 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 SQL queries or table references.

# Nationality Dropdown for Policy Holder

Type: CORRECT\_VALUE

Title: Nationality Dropdown for Policy Holder  
  
Acceptance Criteria:  
1. The nationality field should be a dropdown list containing the following options:  
 - IP  
 - PH  
 - 5  
 - 8  
 - LA  
 - SL  
2. The default value for the nationality field should be "IN".  
3. The nationality field should be displayed on the "Policy Holder" tab.  
4. The field should be editable, allowing users to select a different nationality from the list.  
5. The field should be case-insensitive and automatically convert any input to uppercase.  
6. The field should have a prompt labeled "Citizenship" aligned to the right of the field.  
7. The field should be visible and accessible to users.  
  
Definition of Done:  
- The nationality field is implemented as a dropdown list with the specified options.  
- The default value is set to "IN".  
- The field is placed on the "Policy Holder" tab and is editable.  
- The field converts any input to uppercase.  
- The prompt "Citizenship" is correctly aligned to the right of the field.  
- The field is visible and accessible to users.  
  
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.

# Country of Residence Dropdown for Policy Holder

Type: CORRECT\_VALUE

Title: Country of Residence Dropdown for Policy Holder  
  
Acceptance Criteria:  
1. The country of residence field should display a dropdown list with predefined values.  
2. The dropdown list should include the following options:  
 - PH  
 - 5  
 - 8  
 - 10  
 - LA  
 - SL  
3. The field should not be mandatory, allowing users to leave it blank if necessary.  
4. The field should be editable, allowing users to select or change the value as needed.  
5. The field should be displayed on the "POLICY\_HOLDER" tab page.  
6. The field should be visually styled with a white background and black text.  
  
Definition of Done:  
- The country of residence field is implemented and displays a dropdown list with the specified options.  
- The field is editable and not mandatory.  
- The field is correctly positioned and styled on the "POLICY\_HOLDER" tab page.  
- The functionality is tested and verified to ensure it meets the acceptance criteria.

# Validate Age Proof ID and Trigger Necessary Validation Processes

Type: CORRECT\_VALUE

Title: Validate Age Proof ID and Trigger Necessary Validation Processes  
  
Acceptance Criteria:  
1. When the user enters an age proof ID and the age proof type is "Pan Card":  
 - The system should navigate to the PAN validation section.  
 - The PAN validation process should be executed.  
 - The system should then navigate to the PAN issuance date section.  
  
2. When the user enters an age proof ID and the age proof type is either "AC" or "ACS" with a length of 12 characters:  
 - The system should format the date of birth.  
 - The system should prepare the Aadhaar details for validation.  
 - The Aadhaar validation process should be executed.  
 - The age proof ID should be updated with the validated Aadhaar details.  
 - Debug information should be logged for the validation process.  
  
Definition of Done:  
- The age proof ID validation logic is implemented and tested.  
- The system correctly navigates and executes the validation processes based on the age proof type.  
- The age proof ID is updated with validated details when applicable.  
- Debug information is logged for the validation process.  
- The feature is reviewed and approved by stakeholders.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Not applicable as the provided XML content does not include any direct database queries for CRUD operations.

# Input and Update Insurance Amount

Type: CORRECT\_VALUE

Title: Input and Update Insurance Amount  
  
Acceptance Criteria:  
1. The insurance amount field should allow the user to input a numeric value.  
2. The field should be editable, allowing both insertion and updates.  
3. The field should be displayed with a prompt labeled "Amount of Ins." aligned to the right.  
4. Upon entering the insurance amount and pressing the next item key, the focus should automatically move to the next field, which is the policyholder's father's name.  
  
Definition of Done:  
1. The insurance amount field is implemented and visible on the user interface.  
2. The field accepts numeric input and allows for both new entries and updates.  
3. The prompt "Amount of Ins." is correctly displayed and aligned.  
4. The focus moves to the next field (policyholder's father's name) upon pressing the next item key after entering the insurance amount.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- No specific DB queries provided in the XML content.

# Nationality Selection for Policyholder

Type: CORRECT\_VALUE

Title: Nationality Selection for Policyholder  
  
Acceptance Criteria:  
1. The nationality field should display a list of predefined options for the user to select from.  
2. The list should include the following options: "PH", "5", "8", "10", "LA", and "SL".  
3. The field should be labeled "Nationality" and should be positioned appropriately on the form.  
4. The field should not be mandatory, allowing users to leave it blank if necessary.  
5. The field should be editable, allowing users to update the nationality if needed.  
6. The field should be displayed on the "POLICY\_HOLDER" tab page.  
7. The field should have a white background and black text for readability.  
8. The field should be initialized with no value selected by default.  
  
Definition of Done:  
- The nationality field is implemented and displays the predefined list of options.  
- The field is labeled "Nationality" and is positioned correctly on the form.  
- The field is not mandatory and can be left blank.  
- The field is editable and can be updated by the user.  
- The field is displayed on the "POLICY\_HOLDER" tab page.  
- The field has a white background and black text.  
- The field is initialized with no value selected by default.  
- All acceptance criteria are met and tested successfully.

# City/District Input Field for Policy Holder

Type: CORRECT\_VALUE

Title: City/District Input Field for Policy Holder  
  
Acceptance Criteria:  
1. The input field for the city or district should allow a maximum of 40 characters.  
2. The input should be automatically converted to uppercase.  
3. The field should be displayed on the "Policy Holder" tab page.  
4. The field should be labeled "City/District" and the label should be right-justified.  
5. The field should be editable, allowing both insert and update operations.  
6. The field should be visually distinct with a lowered bevel and a white background.  
7. The field should trigger a list of values when it gains focus.  
  
Definition of Done:  
1. The city or district input field is implemented and meets all acceptance criteria.  
2. The field is tested to ensure it correctly handles input, updates, and displays data in uppercase.  
3. The field is visually verified to match the specified design.  
4. The list of values is triggered correctly when the field gains focus.  
5. All changes are reviewed and approved.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Not applicable as the provided XML content does not include specific database queries.

# Auto-populate Pin Code List for India

Type: CORRECT\_VALUE

Title: Auto-populate Pin Code List for India  
  
Acceptance Criteria:  
1. When the country name is 'INDIA' and the pin code field is empty, the system should execute a query to fetch distinct pin codes, states, and districts from the `azbj\_state\_district\_pincode` table.  
2. The fetched data should be displayed in a list of values (LOV) for the pin code field.  
3. The pin code field should be set to validate from the list of values.  
4. The list of values should be displayed when the pin code field is double-clicked.  
  
Definition of Done:  
1. The system correctly identifies when the country name is 'INDIA' and the pin code field is empty.  
2. The query to fetch pin codes, states, and districts executes without errors.  
3. The list of values is populated with the correct data and displayed to the user.  
4. The pin code field validates against the list of values.  
5. The list of values is accessible via double-click on the pin code field.  
  
DB queries for Table reference CRUD operations only (With Usage):  
```sql  
SELECT DISTINCT pincode, state, district   
FROM azbj\_state\_district\_pincode;  
```

# Display State List for India

Type: CORRECT\_VALUE

Title: Display State List for India  
  
Acceptance Criteria:  
1. When the country name is 'India', the system should display a list of states for selection.  
2. The list of states should be fetched from the database and ordered by state name.  
3. The state list should be displayed in a user-friendly manner, allowing the user to select the desired state.  
  
Definition of Done:  
1. The functionality is implemented and tested to ensure that the state list is displayed correctly when the country name is 'India'.  
2. The state list is fetched from the database and ordered by state name.  
3. The user interface is updated to display the state list in a user-friendly manner.  
4. The feature is reviewed and approved by the stakeholders.  
  
DB queries for Table reference CRUD operations only (With Usage):  
```sql  
SELECT state\_name, scode   
FROM azbj\_states   
ORDER BY state\_name;  
```  
- This query is used to fetch the list of states ordered by state name.

# Select Relationship of Proposer to Life Assured

Type: CORRECT\_VALUE

Title: Select Relationship of Proposer to Life Assured  
  
Acceptance Criteria:  
1. The list of relationships should include the following options:  
 - PH (Policy Holder)  
 - 5  
 - 8  
 - 10  
 - LA (Life Assured)  
 - SL (No)  
2. When the user selects an option from the list, the system should allow the user to proceed to the next item, which is the "SAME\_AS\_LA" field.  
  
Definition of Done:  
1. The list of relationships is displayed correctly with the specified options.  
2. The user can select an option from the list.  
3. Upon selection, the system navigates to the "SAME\_AS\_LA" field.  
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.

# Select Preferred Communication Language

Type: CORRECT\_VALUE

Title: Select Preferred Communication Language  
  
Acceptance Criteria:  
1. The user should be able to select the preferred communication language from a dropdown list.  
2. The dropdown list should display languages fetched from the database.  
3. If the preferred language is not provided in the proposal, the user should be prompted to select the language of the permanent address state.  
4. The selected language should be saved and updated in the system.  
  
Definition of Done:  
1. The dropdown list for selecting the preferred communication language is implemented and functional.  
2. The list of languages is fetched from the database and displayed correctly.  
3. The system prompts the user to select the language of the permanent address state if the preferred language is not provided in the proposal.  
4. The selected language is saved and can be updated as needed.  
  
DB queries for Table reference CRUD operations only (With Usage):  
```sql  
SELECT LANGUAGE\_NAME, LANGUAGE\_ID  
FROM azbj\_comm\_language  
ORDER BY language\_id;  
```  
Usage: This query fetches the list of available languages to populate the dropdown list for selecting the preferred communication language.

# Add Area/Pincode Functionality

Type: CORRECT\_VALUE

Title: Add Area/Pincode Functionality  
  
Acceptance Criteria:  
1. When the "ADD AREA/PINCODE" button is pressed, the system should open a form to input area or pincode details.  
2. The form should be named "AZBJ\_AREA\_PINCODE\_FREE\_TXT".  
3. The form should not hide the current form.  
4. The form should replace the current form's data.  
5. The form should not query only.  
  
Definition of Done:  
- The "ADD AREA/PINCODE" button is visible and enabled in the policy holder section.  
- Pressing the button successfully opens the "AZBJ\_AREA\_PINCODE\_FREE\_TXT" form.  
- The form behaves as specified in 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 Proof of Residence for Policy Holders

Type: CORRECT\_VALUE

Title: Manage Proof of Residence for Policy Holders  
  
Acceptance Criteria:  
1. The system should allow users to insert and update the proof of residence information for policy holders.  
2. The proof of residence field should be a list item with predefined values.  
3. The list should include options such as "PH" and "No".  
4. The proof of residence field should be associated with the policy holder's tab.  
5. The field should be hidden from view by default.  
  
Definition of Done:  
1. The proof of residence field can be successfully inserted and updated.  
2. The predefined list values are correctly displayed and selectable.  
3. The field is correctly associated with the policy holder's tab and is hidden by default.  
4. All changes are saved and reflected in the system without errors.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include specific SQL queries or table references.

# Input and Validate Address Line 3 for Policy Holder

Type: CORRECT\_VALUE

Title: Input and Validate Address Line 3 for Policy Holder  
  
Acceptance Criteria:  
1. The address line 3 field should allow a maximum of 100 characters.  
2. The input should be restricted to uppercase characters.  
3. The field should have a dropdown list of places that the user can select from.  
4. When the user double-clicks on the address line 3 field, the dropdown list should be displayed.  
5. The dropdown list should be populated with unique places based on the district entered in the address line 4 field.  
6. If no matching places are found, the dropdown list should include an option labeled "NOT FOUND".  
  
Definition of Done:  
1. The address line 3 field is implemented with the specified character restrictions and validation.  
2. The dropdown list is functional and displays the correct options based on the district input.  
3. The user can successfully select a place from the dropdown list.  
4. The system correctly handles cases where no matching places are found by displaying the "NOT FOUND" option.  
5. All acceptance criteria are met and tested.  
  
DB queries for Table reference CRUD operations only (With Usage):  
```sql  
SELECT UNIQUE state, district, place  
FROM azbj\_urban\_rural\_data   
WHERE UPPER(district) = UPPER(:CORRECT\_VALUE.PH\_ADDRESS\_LINE4)  
UNION  
SELECT null, null, 'NOT FOUND'  
FROM dual  
ORDER BY place;  
```  
  
This query is used to populate the dropdown list with unique places based on the district entered in the address line 4 field. If no matching places are found, it includes an option labeled "NOT FOUND".

# Auto-populate Area and Postal Code based on Country, State, and District

Type: CORRECT\_VALUE

Detailed description: As a user, I want the system to automatically populate the area and postal code fields based on the country name, state, and district information provided, so that I can ensure accurate and consistent data entry for addresses.  
  
Acceptance criteria:  
1. When the user double-clicks on the area field, if the country name is 'INDIA', the system should:  
 - Set the postal code parameter to the value of the postal code field.  
 - Set the state parameter to the value of the state field.  
 - Set the district parameter to the value of the district field.  
 - Display a list of values for the area field.  
  
2. When the user navigates to the area field, if the country name is 'INDIA', the system should:  
 - Set the postal code parameter to the value of the postal code field.  
 - Set the state parameter to the value of the state field.  
 - Set the district parameter to the value of the district field.  
 - Execute a query to fetch distinct areas and postal codes from the database where the state and district match the provided values, and the postal code matches the provided value or is null.  
 - Populate the area field with the list of values fetched from the query.  
  
Definition of Done:  
- The area and postal code fields are automatically populated based on the country name, state, and district information.  
- The list of values for the area field is displayed correctly when the user double-clicks or navigates to the field.  
- The system performs the necessary database queries to fetch the relevant data.  
- The functionality is tested and verified to ensure accurate and consistent data entry.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- The query to fetch distinct areas and postal codes:  
 ```sql  
 SELECT DISTINCT Area, PINCODE   
 FROM azbj\_state\_district\_pincode   
 WHERE UPPER(STATE) = UPPER(:parameter.v\_state)   
 AND UPPER(DISTRICT) = UPPER(:parameter.v\_district)   
 AND PINCODE = NVL(:parameter.V\_POSTCODE, PINCODE)   
 ORDER BY PINCODE, Area;  
 ```

# Input and Update City/District Information

Type: CORRECT\_VALUE

Title: Input and Update City/District Information  
  
Acceptance Criteria:  
1. The "City/District" field should accept input up to 40 characters and automatically convert all input to uppercase.  
2. The field should be editable, allowing both insertion and updates.  
3. When the field gains focus, a list of values should be displayed to assist the user in selecting the appropriate city or district.  
4. The field should be visually styled with a white background and black text, and it should be positioned at the specified coordinates on the form.  
5. The prompt for the field should be "City/District" and should be aligned to the right of the input field.  
  
Definition of Done:  
- The "City/District" field is implemented and meets all the acceptance criteria.  
- The field is tested to ensure it accepts and displays data correctly, including the automatic conversion to uppercase.  
- The list of values is triggered correctly when the field gains focus.  
- The visual styling and positioning of the field are verified to match the specifications.  
- The prompt is correctly displayed and aligned as specified.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Not applicable as the provided XML content does not include specific database queries.

# Dynamic Control of 'Area (Free Text)' Field Based on 'PH\_AREA' Status

Type: CORRECT\_VALUE

Detailed description: As a user, I want the system to manage the "Area (Free Text)" field based on the status of the "PH\_AREA" field and the latest request status from the "azbj\_area\_pincode\_request" table, so that the field's insert and update permissions are dynamically controlled and relevant status messages are displayed.  
  
Acceptance criteria:  
1. If the "PH\_AREA" field has the value 'NOT FOUND':  
 - The "Area (Free Text)" field should allow insert and update operations.  
 - The system should fetch the latest request status for the corresponding application number and address type 'PH' from the "azbj\_area\_pincode\_request" table.  
 - If a matching record is found, a message should be displayed indicating the status of the "Area Free Text".  
 - If no matching record is found, no message should be displayed.  
2. If the "PH\_AREA" field does not have the value 'NOT FOUND':  
 - The "Area (Free Text)" field should not allow insert and update operations.  
  
Definition of Done:  
- The "Area (Free Text)" field's permissions are correctly set based on the "PH\_AREA" field's value.  
- The system correctly fetches and displays the latest request status message when applicable.  
- The functionality is tested and verified to work as expected in various scenarios.  
  
DB queries for Table reference CRUD operations only(With Usage):  
```sql  
-- Fetch the latest request status for the given application number and address type 'PH'  
SELECT 'The Area Free Text for "' || area\_free\_text || '" Status is ' || DECODE(request\_status, 'A', 'Approved', 'P', 'Pending', 'R', 'Rejected')  
INTO v\_free\_text  
FROM azbj\_area\_pincode\_request  
WHERE application\_no = :correct\_value.appln\_no  
 AND address\_type = 'PH'  
 AND area\_free\_text IS NOT NULL  
 AND REQUEST\_DATE = (  
 SELECT MAX(REQUEST\_DATE)  
 FROM azbj\_area\_pincode\_request  
 WHERE application\_no = :correct\_value.appln\_no  
 AND address\_type = 'PH'  
 AND area\_free\_text IS NOT NULL  
 );  
```

# Auto-populate and validate pin code field based on country name

Type: CORRECT\_VALUE

Title: Auto-populate and validate pin code field based on country name  
  
Acceptance Criteria:  
1. When the country name is 'INDIA' and the pin code field is empty, the system should execute a query to fetch distinct pin codes, states, and districts from the `azbj\_state\_district\_pincode` table.  
2. The fetched data should be displayed in a list of values (LOV) for the pin code field.  
3. The pin code field should be validated against the list of values to ensure that only valid pin codes are selected.  
  
Definition of Done:  
- The system correctly identifies when the country name is 'INDIA' and the pin code field is empty.  
- The query to fetch distinct pin codes, states, and districts is executed successfully.  
- The list of values (LOV) is populated with the fetched data and displayed for the pin code field.  
- The pin code field is validated against the list of values, ensuring only valid pin codes can be selected.  
  
DB queries for Table reference CRUD operations only (With Usage):  
```sql  
SELECT DISTINCT pincode, state, district   
FROM azbj\_state\_district\_pincode;  
```  
Usage: This query is used to populate the list of values (LOV) for the pin code field when the country name is 'INDIA' and the pin code field is empty.

# Navigate to Next Tab on Button Press

Type: CORRECT\_VALUE

Detailed description: As a user, I want to navigate to the next tab in the application when certain conditions are met, so that I can proceed with the data entry process efficiently.  
  
Acceptance criteria:  
1. When the "Next Tab" button is pressed, the system should check if the nominee's name and birthplace fields are not null.  
2. If the fields are not null, the system should store the nominee's name, birthplace, date of birth, and relationship in temporary variables.  
3. The system should then enable or disable the "COVER\_DETAILS" tab based on certain conditions.  
4. The system should query the `AZBJ\_NACH\_REGISTRATION\_DET` table to retrieve the `source` field where the `appln\_no` matches the current application number.  
5. If the `source` field matches specific values ('ENACH\_OPEN\_LINK', 'INSTAB\_ENACH', 'ENACH LINK', 'OPEN\_LINK\_NEW\_ENACH'), the system should set the `RENEWAL\_PAYMENT` field to 'NACH' and make the `ENACH` field visible with the value 'ENACH Case'.  
6. If the `source` field does not match the specified values, the system should hide the `ENACH` field.  
7. The system should then navigate to the `PACKAGE\_CODE` field.  
  
Definition of Done:  
- The "Next Tab" button functionality is implemented and tested.  
- The system correctly checks the nominee's name and birthplace fields.  
- The system stores the nominee's details in temporary variables.  
- The "COVER\_DETAILS" tab is enabled or disabled based on the conditions.  
- The system queries the `AZBJ\_NACH\_REGISTRATION\_DET` table and handles the `source` field appropriately.  
- The `RENEWAL\_PAYMENT` and `ENACH` fields are updated and displayed correctly based on the `source` field.  
- The system navigates to the `PACKAGE\_CODE` field after processing.  
- All acceptance criteria are met, and the functionality is verified through testing.  
  
DB queries for Table reference CRUD operations only(With Usage):  
```sql  
-- Query to retrieve the source field from AZBJ\_NACH\_REGISTRATION\_DET table  
SELECT source  
INTO v\_enach\_source  
FROM AZBJ\_NACH\_REGISTRATION\_DET  
WHERE appln\_no = TO\_CHAR(:correct\_value.appln\_no);  
```

# Add Area/Pincode Functionality

Type: CORRECT\_VALUE

Title: Add Area/Pincode Functionality  
  
Acceptance Criteria:  
1. When the "ADD AREA/PINCODE" button is pressed, a new form should be called to allow the user to input area or pincode details.  
2. The new form should open without hiding the current form.  
3. The new form should replace any existing form in the same session.  
4. The new form should not perform any query operations upon opening.  
  
Definition of Done:  
1. The "ADD AREA/PINCODE" button is visible and enabled on the POLICY\_HOLDER tab.  
2. Pressing the button successfully opens a new form for area or pincode input.  
3. The current form remains visible and is not hidden.  
4. The new form replaces any existing form in the session.  
5. The new form does not perform any query operations upon opening.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include any direct database queries.

# Auto-populate and dynamically update 'Landmark/Area' field based on country, state, and district

Type: CORRECT\_VALUE

Title: Auto-populate and dynamically update 'Landmark/Area' field based on country, state, and district  
  
Acceptance Criteria:  
1. When the user double-clicks on the 'Landmark/Area' field and the country is set to India:  
 - The postal code, state, and district fields should be populated with the corresponding values.  
 - The list of available areas should be displayed.  
2. When the 'Landmark/Area' field is selected:  
 - If the country is set to India, the list of available areas should be dynamically updated based on the state and district information.  
 - If the country is not set to India, the list of available areas should be cleared.  
  
Definition of Done:  
- The system correctly populates the postal code, state, and district fields when the 'Landmark/Area' field is double-clicked, provided the country is set to India.  
- The list of available areas is dynamically updated based on the state and district information when the 'Landmark/Area' field is selected.  
- The list of available areas is cleared if the country is not set to India.  
- All changes are tested and verified to ensure they meet the acceptance criteria.  
  
DB queries for Table reference CRUD operations only (With Usage):  
```sql  
SELECT DISTINCT Area, PINCODE   
FROM azbj\_state\_district\_pincode   
WHERE UPPER(STATE) = UPPER(:parameter.v\_state)   
AND UPPER(DISTRICT) = UPPER(:parameter.v\_district)   
AND PINCODE = NVL(:parameter.V\_POSTCODE, PINCODE)   
ORDER BY PINCODE, Area;  
```  
  
This query is used to fetch the distinct areas and pincodes based on the state and district information provided.

# Manage Residence Proof Information for Policy Holders

Type: CORRECT\_VALUE

Detailed description: As a user, I want to manage the residence proof information for policy holders within the system, so that I can ensure accurate and up-to-date records for each policy holder.  
  
Acceptance criteria:  
1. The residence proof information should be presented as a list item within the policy holder's section.  
2. The list should include predefined values such as "PH" and "SL" to standardize the input.  
3. The residence proof field should be editable, allowing users to insert and update information as needed.  
4. The field should be hidden from view by default but can be made visible when required.  
5. The field should be associated with a specific tab page named "POLICY\_HOLDER" for easy navigation.  
  
Definition of Done:  
- The residence proof information can be successfully added, updated, and saved within the policy holder's section.  
- The predefined values are correctly displayed in the list.  
- The field's visibility can be toggled as needed.  
- The field is correctly associated with the "POLICY\_HOLDER" tab page.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Not applicable as the provided XML content does not include specific database queries.

# Dropdown Selection and Validation for Address Line 3

Type: CORRECT\_VALUE

Title: Dropdown Selection and Validation for Address Line 3  
  
Acceptance Criteria:  
1. When the user double-clicks on the address line 3 field, a dropdown list of places should appear.  
2. The dropdown list should be populated with unique state, district, and place values from the database.  
3. If the district entered in address line 4 does not match any record, the dropdown should display 'NOT FOUND'.  
4. The user should receive a warning message to select the place from the dropdown even if the spelling is different but the pronunciation is the same.  
5. The address line 3 field should be validated against the list of places in the dropdown.  
  
Definition of Done:  
1. The dropdown list for address line 3 is functional and displays the correct data.  
2. The warning message is displayed when the user focuses on the address line 3 field.  
3. The address line 3 field is validated against the dropdown list.  
4. The feature is tested and verified to work as expected.  
  
DB queries for Table reference CRUD operations only (With Usage):  
```sql  
SELECT UNIQUE state, district, place  
FROM azbj\_urban\_rural\_data  
WHERE UPPER(district) = UPPER(:district)  
UNION  
SELECT null, null, 'NOT FOUND'  
FROM dual  
ORDER BY place;  
```  
- This query is used to populate the dropdown list with unique state, district, and place values based on the district entered in address line 4. If no matching district is found, 'NOT FOUND' is displayed.

# Conditional Editability and Status Message for Area (Free Text) Field

Type: CORRECT\_VALUE

Detailed description: As a user, I want the "Area (Free Text)" field to be conditionally editable based on the value of the "PH\_M\_AREA" field and to display a status message if certain conditions are met.  
  
Acceptance criteria:  
1. If the "PH\_M\_AREA" field has the value "NOT FOUND":  
 - The "Area (Free Text)" field should be editable (both insert and update allowed).  
 - A message should be displayed with the status of the area free text if a corresponding record exists in the `azbj\_area\_pincode\_request` table.  
 - The message should be in the format: "The Area Free Text for [area\_free\_text] Status is [status]", where the status can be "Approved", "Pending", or "Rejected".  
 - If no corresponding record is found, no message should be displayed.  
2. If the "PH\_M\_AREA" field does not have the value "NOT FOUND":  
 - The "Area (Free Text)" field should not be editable (both insert and update not allowed).  
  
Definition of Done:  
- The "Area (Free Text)" field's editability is correctly toggled based on the value of the "PH\_M\_AREA" field.  
- The status message is correctly displayed when applicable.  
- The functionality is tested and verified to work as expected.  
  
DB queries for Table reference CRUD operations only (With Usage):  
```sql  
SELECT 'The Area Free Text for "' || area\_free\_text || '" Status is ' || DECODE(request\_status, 'A', 'Approved', 'P', 'Pending', 'R', 'Rejected')  
INTO v\_free\_text  
FROM azbj\_area\_pincode\_request  
WHERE application\_no = :correct\_value.appln\_no  
 AND address\_type = 'PH\_M'  
 AND area\_free\_text IS NOT NULL  
 AND REQUEST\_DATE = (SELECT MAX(REQUEST\_DATE)   
 FROM azbj\_area\_pincode\_request  
 WHERE application\_no = :correct\_value.appln\_no  
 AND address\_type = 'PH\_M'  
 AND area\_free\_text IS NOT NULL);  
```

# Education Field with Predefined List

Type: CORRECT\_VALUE

Title: Education Field with Predefined List  
  
Acceptance Criteria:  
1. The "Education" field should display a list of predefined education levels.  
2. The list should include the following options:  
 - PH  
 - 5  
 - 8  
 - SL  
 - EDUCATION  
3. The field should be editable, allowing users to select an option from the list.  
4. The field should be displayed on the "PH\_INFO" tab.  
5. The field should have a white background and black text.  
6. The prompt for the field should be "Education" and should be right-justified.  
7. The prompt should be bold and in "MS Sans Serif" font.  
  
Definition of Done:  
- The "Education" field is implemented and displays the predefined list of education levels.  
- Users can select an option from the list and the selection is saved.  
- The field and its prompt are displayed correctly on the "PH\_INFO" tab with the specified styles and properties.  
- The functionality is tested and verified to meet the acceptance criteria.

# Manage Marital Status of Policyholder

Type: CORRECT\_VALUE

Title: Manage Marital Status of Policyholder  
  
Acceptance Criteria:  
1. When the marital status is set to 'M' (Married), the fields for the spouse and father should be enabled.  
2. When the marital status is set to 'W' (Widowed), the field for the spouse should be enabled.  
3. If the title is 'MRS' and the gender is 'F' (Female) and the marital status is 'S' (Single), an error message should be displayed prompting the user to select an appropriate title or marital status.  
4. If the title is 'MISS' and the gender is 'F' (Female) and the marital status is 'M' (Married), an error message should be displayed prompting the user to select an appropriate title or marital status.  
5. The system should load the appropriate visual attributes for the current item based on the marital status.  
  
Definition of Done:  
- The system correctly enables or disables fields based on the selected marital status.  
- Error messages are displayed when there is a mismatch between the title, gender, and marital status.  
- The visual attributes for the current item are loaded correctly based on the marital status.  
- All acceptance criteria are met and tested.  
- 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 direct database CRUD operations that can be executed without modification.

# Handle Father's Name Input Based on Marital Status

Type: CORRECT\_VALUE

Title: Handle Father's Name Input Based on Marital Status  
  
Acceptance Criteria:  
1. When the user navigates to the father's name field, if the policy holder's marital status is 'W' (Widowed) or 'D' (Divorced), the system should enable the spouse field.  
2. If the policy holder's marital status is not 'W' or 'D', the system should not enable the spouse field.  
3. When the user attempts to move to the next field from the father's name field:  
 - If the spouse field is enabled, the system should navigate to the spouse field.  
 - If the spouse field is not enabled but the relationship field is enabled, the system should navigate to the relationship field.  
 - If neither the spouse nor the relationship field is enabled, the system should navigate to the "Same as LA" field.  
  
Definition of Done:  
- The father's name field should be displayed with the correct label and formatting.  
- The system should correctly enable or disable the spouse field based on the policy holder's marital status.  
- The navigation logic should work as specified, ensuring a smooth user experience when moving between fields.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Not applicable as the provided XML content does not include any direct database queries.

# Profession Description Field with Predefined List

Type: CORRECT\_VALUE

Title: Profession Description Field with Predefined List  
  
Acceptance Criteria:  
1. The profession description field should allow users to select a value from a predefined list of professions.  
2. The list of professions should be fetched dynamically based on the industry code entered in the industry field.  
3. The list should only include professions that are currently active (i.e., where the end date is null).  
4. The list should be sorted alphabetically by the profession description.  
5. When a user double-clicks on the profession description field, the list of professions should be displayed.  
  
Definition of Done:  
1. The profession description field is implemented and allows selection from a predefined list.  
2. The list of professions is dynamically fetched based on the industry code.  
3. Only active professions are included in the list.  
4. The list is sorted alphabetically by profession description.  
5. Double-clicking the profession description field displays the list of professions.  
  
DB queries for Table reference CRUD operations only(With Usage):  
```sql  
SELECT a.occ\_code, a.occ\_desc   
FROM azbj\_occupation a, azbj\_ind\_occ\_link b   
WHERE a.occ\_code = b.occ\_code   
AND b.ind\_code = TO\_NUMBER(:correct\_value.PH\_INDUSTRY)  
AND end\_date IS NULL  
ORDER BY a.occ\_desc;  
```

# Income Proof Selection

Type: CORRECT\_VALUE

Title: Income Proof Selection  
  
Acceptance Criteria:  
1. The income proof field should display a list of predefined options for the user to select from.  
2. The list should include the following options:  
 - PH  
 - 5  
 - 8  
 - SL  
3. The field should be editable, allowing users to insert and update their selection.  
4. The field should be displayed on the "PH\_INFO" tab page.  
5. The field should have a prompt labeled "Income Proof" aligned to the right of the field.  
  
Definition of Done:  
- The income proof field is implemented and displays the predefined list of options.  
- Users can successfully select and update their choice of income proof.  
- The field is correctly positioned on the "PH\_INFO" tab page with the appropriate prompt.  
- The functionality is tested and verified to meet the acceptance criteria.

# User can select assessee type from predefined list

Type: CORRECT\_VALUE

Title: User can select assessee type from predefined list  
  
Acceptance Criteria:  
1. The list of assessee types should include the following options:  
 - "PH"  
 - "5"  
 - "8"  
 - "SL"  
2. The list should be displayed in uppercase.  
3. The list should be visible on the "PH\_INFO" tab page.  
4. The list should be editable, allowing users to insert and update values.  
  
Definition of Done:  
- The list of assessee types is implemented and displays the correct options.  
- The list is visible on the "PH\_INFO" tab page.  
- Users can insert and update values in the list.  
- The list is displayed in uppercase.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Not applicable as the provided XML content does not include specific database queries.

# Manage Industry Description Validation and Conditional Display

Type: CORRECT\_VALUE

Title: Manage Industry Description Validation and Conditional Display  
  
Acceptance Criteria:  
1. When the user double-clicks on the industry description field, a list of valid industry values should be displayed.  
2. Upon validating the industry description field:  
 - The system should count the number of records in the `azbj\_strip\_questionnaire\_det` table where `appln\_no` matches the current application number, `question\_id` is 85, and `sub\_question` is one of 52, 53, 54, 55, or 56.  
 - If the `fsc\_code` is '3000000007' or the `v\_inhouse\_flag` or `v\_web\_aggr` flags are 'Y', and the count of records is greater than 0:  
 - If the industry description is 'CRPF' or 'DEFENCE', the `CRPF\_QUES\_PH` field should be made visible and enabled.  
 - Otherwise, the `CRPF\_QUES\_PH` field should be hidden and disabled.  
 - If the above conditions are not met, the `CRPF\_QUES\_PH` field should be hidden and disabled.  
  
Definition of Done:  
- The industry description field should display a list of valid values upon double-click.  
- The system should correctly validate the industry description field based on the specified criteria.  
- The `CRPF\_QUES\_PH` field should be conditionally displayed and enabled/disabled based on the validation logic.  
- All changes should be logged for debugging purposes.  
  
DB queries for Table reference CRUD operations only(With Usage):  
```sql  
-- Query to count records in azbj\_strip\_questionnaire\_det table  
SELECT COUNT(1)  
INTO v\_cnt  
FROM azbj\_strip\_questionnaire\_det  
WHERE appln\_no = :correct\_value.appln\_no  
 AND question\_id = 85  
 AND sub\_question IN (52, 53, 54, 55, 56);  
```

# Dynamic Management of Employer Name Field Based on Occupation and Product Code

Type: CORRECT\_VALUE

Detailed description: As a user, I want the system to dynamically manage the visibility and enabled state of the "Employer Name" field based on the selected occupation and product code, so that the form adapts to specific business rules and displays relevant information accordingly.  
  
Acceptance criteria:  
1. When the occupation is either 'SG' or 'SN' and the product code is one of 297, 315, or 345:  
 - The "Employer Name" field should become visible and enabled.  
 - The "Employer Name" field should be populated with a list of values retrieved from the database where the system type is 'HEALTH\_CARE' and the system code is 'TOP\_CORPORATES'.  
2. When the occupation is not 'SG' or 'SN' or the product code is not one of 297, 315, or 345:  
 - The "Employer Name" field should be hidden and disabled.  
 - The "Employer's Name" field should be enabled.  
  
Definition of Done:  
- The form dynamically adjusts the visibility and enabled state of the "Employer Name" field based on the specified conditions.  
- The "Employer Name" field is correctly populated with values from the database when the conditions are met.  
- The changes are tested and verified to ensure they meet the acceptance criteria.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Retrieve list of values for "Employer Name":  
 ```sql  
 SELECT CHAR\_VALUE  
 FROM azbj\_system\_constants  
 WHERE sys\_type = 'HEALTH\_CARE'  
 AND sys\_code = 'TOP\_CORPORATES'  
 ORDER BY CHAR\_VALUE;  
 ```

# Next Tab Button Functionality

Type: CORRECT\_VALUE

Title: Next Tab Button Functionality  
  
Acceptance Criteria:  
1. When the "Next Tab" button is pressed:  
 - If the Aadhaar field is empty, the Age Proof field is either 'AC' or 'ACS', and the Age Proof ID field is not empty, then the Aadhaar field should be auto-filled with the value from the Age Proof ID field.  
 - The system should check if the area exists in the `azbj\_state\_district\_pincode` table based on the state, district, and pincode parameters. If the area is not found and the area field is not 'NOT FOUND' or 'NOTFOUND', the system should navigate to the area field for correction.  
 - If the area is found, the system should check if the place exists in the `azbj\_urban\_rural\_data` table based on the address line 3 field. If the place is not found and the address line 3 field is not 'NOT FOUND' or 'NOTFOUND', the system should navigate to the address line 3 field for correction.  
 - If both the area and place are found, the system should enable or disable the 'PH\_INFO' tab and navigate to the education field.  
 - The system should also enable or disable the nominee section based on specific conditions.  
  
Definition of Done:  
- The "Next Tab" button functionality is implemented as per the acceptance criteria.  
- The system correctly auto-fills the Aadhaar field when conditions are met.  
- The system navigates to the correct fields for correction when the area or place is not found.  
- The 'PH\_INFO' tab and nominee section are enabled or disabled based on the specified conditions.  
- All changes are tested and verified to ensure they work as expected.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Query to check area existence:  
 ```sql  
 SELECT DISTINCT Area  
 INTO v\_area  
 FROM azbj\_state\_district\_pincode  
 WHERE UPPER(STATE) = UPPER(:parameter.v\_state)  
 AND UPPER(DISTRICT) = UPPER(:parameter.v\_district)  
 AND pincode = NVL(:parameter.V\_POSTCODE, pincode)  
 AND UPPER(area) = UPPER(:CORRECT\_VALUE.PH\_AREA);  
 ```  
  
- Query to check place existence:  
 ```sql  
 SELECT UNIQUE place  
 INTO v\_place  
 FROM azbj\_urban\_rural\_data  
 WHERE UPPER(place) = UPPER(:CORRECT\_VALUE.PH\_ADDRESS\_LINE3);  
 ```

# Automatic Navigation Based on Product Code

Type: CORRECT\_VALUE

Title: Automatic Navigation Based on Product Code  
  
Acceptance Criteria:  
1. When the product code is 277, the system should navigate to the "MONEY\_BOSTER" item.  
2. When the product code is not 277, the system should navigate to the "BI" item.  
  
Definition of Done:  
1. The system correctly identifies the product code.  
2. The system navigates to the appropriate item based on the product code.  
3. The navigation logic 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.

# Proposal Deposit Field Implementation

Type: CORRECT\_VALUE

Title: Proposal Deposit Field Implementation  
  
Acceptance Criteria:  
1. The proposal deposit field should allow the user to select from predefined values (5, 8, SL) or enter a custom value.  
2. The field should be visible and accessible on the "COVER\_DETAILS" tab of the proposal form.  
3. The field should be editable, allowing both insertion of new values and updates to existing values.  
4. The field should be displayed with a prompt labeled "Prop.Deposit" aligned to the right of the field.  
5. The field should be styled with a white background, black text, and use the "MS Sans Serif" font.  
  
Definition of Done:  
- The proposal deposit field is implemented and functional as per the acceptance criteria.  
- The field is tested to ensure it allows selection from predefined values and custom input.  
- The field is visible and correctly styled on the "COVER\_DETAILS" tab.  
- The field's prompt is correctly aligned and displayed.  
- All changes are reviewed and approved.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include specific database queries.

# Validate Nominee Relationship Based on Gender

Type: CORRECT\_VALUE

Title: Validate Nominee Relationship Based on Gender  
  
Acceptance Criteria:  
1. If the gender of the insured person is male and the nominee relationship is set to "Husband", an error message should be displayed stating, "Gender of LA is Male, NOMINEE RELATION TO LA cannot be equal to HUSBAND".  
2. If the gender of the insured person is female and the nominee relationship is set to "Wife", an error message should be displayed stating, "Gender of LA is Female, NOMINEE RELATION TO LA cannot be equal to WIFE".  
3. After validation, the system should navigate to the next item in the sequence, which is determined by the application logic.  
  
Definition of Done:  
- The validation logic for nominee relationships based on the gender of the insured person is implemented.  
- Error messages are displayed correctly when invalid relationships are entered.  
- The system navigates to the appropriate next item after validation.  
- The functionality is tested and verified to ensure it works as expected.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- The following query is used to fetch the list of nominee details:  
 ```sql  
 SELECT CHAR\_VALUE FROM azbj\_system\_constants WHERE sys\_type='NOM\_DET';  
 ```  
- This query is used to populate the list of values for nominee relationships.

# User can select relationship of nominee from predefined list

Type: CORRECT\_VALUE

Title: User can select relationship of nominee from predefined list  
  
Acceptance Criteria:  
1. The list of values for the relationship of the nominee should include options such as "IP", "5", "8", "10", "N", "1", and "SL".  
2. The list should be displayed in a dropdown format.  
3. The selected value should be stored in the database column `RELATIONSHIP\_NOMINEE`.  
4. The field should allow both insertion and updating of values.  
5. The field should be case-restricted to uppercase.  
6. The field should be positioned appropriately on the form and should have a prompt labeled "Relation to Nominee" aligned to the right.  
7. The prompt should be bold and in the "MS Sans Serif" font.  
8. The field should be part of the "NOMINEE\_HEALTH\_DTLS" tab page.  
9. The field should trigger navigation to the next item in the form when the user presses the key to move to the next item.  
  
Definition of Done:  
- The user can select a relationship from the predefined list.  
- The selected relationship is saved in the `RELATIONSHIP\_NOMINEE` column.  
- The field and its prompt are displayed correctly on the form.  
- The field allows insertion and updating of values.  
- The field triggers navigation to the next item 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.

# Navigation based on Penny Drop status

Type: CORRECT\_VALUE

Title: Navigation based on Penny Drop status  
  
Acceptance Criteria:  
1. When the "Next Tab" button is pressed, the system should check the status of the "Penny Drop" operation.  
2. If the "Penny Drop" operation is successful, the system should navigate to the "Proposal Sign Date" field.  
3. If the "Penny Drop" operation is not successful, the system should navigate to the "Bank Account Number" field.  
  
Definition of Done:  
- The "Next Tab" button should be functional and trigger the appropriate navigation based on the "Penny Drop" status.  
- The system should correctly identify the "Penny Drop" status and navigate to the correct field without errors.  
- The user should be able to continue data entry seamlessly after pressing the "Next Tab" button.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include any direct database queries.

# Ensure IP Relation Field is Not Blank

Type: CORRECT\_VALUE

Title: Ensure IP Relation Field is Not Blank  
  
Acceptance Criteria:  
1. If the "IP Relation" field is left blank, the system should display an error message stating "Premium Paid by Relation field should not be blank".  
2. The user should be prompted to fill in the "IP Relation" field before moving to the next field.  
3. The system should automatically navigate to the "Renewal Payment" field after the "IP Relation" field is filled.  
  
Definition of Done:  
- The error message is displayed when the "IP Relation" field is blank.  
- The user is unable to proceed to the next field until the "IP Relation" field is filled.  
- The system successfully navigates to the "Renewal Payment" field after the "IP Relation" field is completed.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Not applicable as the provided XML content does not include any specific database queries.

# Annuity Frequency Dropdown

Type: CORRECT\_VALUE

Title: Annuity Frequency Dropdown  
  
Acceptance Criteria:  
1. The user should be able to see a dropdown list labeled "Annuity Frequency" on the "COVER\_DETAILS" tab.  
2. The dropdown list should contain the following options:  
 - 5  
 - 8  
 - SL (No)  
3. The dropdown list should be disabled by default but should allow insertion and updates when enabled.  
4. The dropdown list should be positioned correctly on the screen as per the design specifications.  
5. The dropdown list should have a white background and black text.  
6. The label "Annuity Frequency" should be right-justified and bold.  
  
Definition of Done:  
- The dropdown list for selecting annuity frequency is implemented and visible on the "COVER\_DETAILS" tab.  
- The dropdown list contains the specified options (5, 8, SL).  
- The dropdown list is disabled by default but can be enabled for insertion and updates.  
- The dropdown list and its label are styled and positioned as per the design specifications.  
- The feature has been reviewed and approved by the stakeholders.

# Premium Finance Selection and Validation

Type: CORRECT\_VALUE

Title: Premium Finance Selection and Validation  
  
Acceptance Criteria:  
1. The premium finance field should display a list of options including "IP" and "PH".  
2. When the user selects an option from the list, the system should validate the selection based on the following logic:  
 - If the selected option is "JL" (Joint Life) and the insured person (IP) and policyholder (PH) are the same, an error message should be displayed: "For Joint Life IP and PH cannot be same".  
  
Definition of Done:  
1. The premium finance field is implemented and displays the correct list of options.  
2. The validation logic is correctly implemented and triggers the appropriate error message when conditions are met.  
3. The feature is tested and verified to ensure it works 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 specific database queries.

# Handle Package Code Selection and Validation

Type: CORRECT\_VALUE

Detailed description: As a user, I want the system to handle the selection and validation of package codes, ensuring that specific actions are taken based on the selected package code and associated product code.  
  
Acceptance criteria:  
1. When the user selects a package code, the system should display a list of available package codes.  
2. If the selected package code is 'LONGLIFE\_VARIANT\_2' and the insured check is '0', the system should navigate to the rider details section, iterate through the records, and delete any record with the rider cover code 'R048A01'.  
3. If the product code is 321 and the package code starts with 'SMART\_CHILD\_', the system should make the child covers control visible and refresh the child covers.  
4. If the product code is 321 and the package code starts with 'SMART\_INCREASE\_', the system should make the percentage increase control visible and enabled.  
5. If the product code is 343 and the package code contains 'STEP\_UP\_', the system should set the deferral period to 0 and the income period to 20.  
6. If the product code is 343 and the package code is 'INCOME\_SINGLE\_LIFE', the system should calculate the income period based on the benefit term, premium term, and deferral period.  
7. If the product code is 343 and the package code contains 'WEALTH\_CREATION\_', the system should set the deferral period to 0.  
8. The system should clear the fund details and SSO fund details sections and reset the portfolio strategy to an empty value.  
9. The system should navigate to the premium item after processing the package code.  
  
Definition of Done:  
- The system correctly displays the list of available package codes.  
- The system performs the specified actions based on the selected package code and product code.  
- The system navigates to the appropriate sections and updates the necessary fields as described.  
- The system clears and resets the specified sections and fields.  
- All acceptance criteria are met without any errors.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- The system should use the following query to fetch available package codes based on the product code:  
 ```sql  
 SELECT FROM azbj\_package\_master WHERE product\_id = :correct\_value.product\_code;  
 ```

# Implement Cover Type Dropdown Field

Type: CORRECT\_VALUE

Title: Implement Cover Type Dropdown Field  
  
Acceptance Criteria:  
1. The cover type field should display a dropdown list with the following options:  
 - "5"  
 - "8"  
 - "SL"  
2. The cover type field should be positioned at the top left of the form.  
3. The cover type field should have a maximum length of 30 characters and should only accept uppercase letters.  
4. The cover type field should be editable, allowing users to insert and update values.  
5. The cover type field should be visually styled with a white background, black text, and a specific font and size.  
6. The cover type field should be part of a tabbed section labeled "COVERS\_GROUP" and should be aligned to the right with a bold prompt.  
  
Definition of Done:  
- The cover type field is implemented and displays the correct options in a dropdown list.  
- The field adheres to the specified visual and functional requirements.  
- The field is tested to ensure it accepts only uppercase letters and has a maximum length of 30 characters.  
- The field is positioned correctly within the form and is part of the "COVERS\_GROUP" tab.  
- The field allows for insertion and updating of values as specified.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include specific database queries.

# Input and Update Premium Value

Type: CORRECT\_VALUE

Title: Input and Update Premium Value  
  
Acceptance Criteria:  
1. The premium value should be a numeric input.  
2. The input field for the premium value should be positioned at the specified coordinates on the user interface.  
3. The premium value should be displayed in uppercase.  
4. The input field should allow insertion and updates.  
5. Upon entering the premium value, the system should automatically navigate to the next item, which is the frequency field.  
  
Definition of Done:  
1. The premium input field is implemented and visible on the user interface.  
2. The premium value can be inserted and updated.  
3. The system navigates to the frequency field after the premium value is entered.  
4. The premium value is displayed in uppercase.  
5. The functionality is tested and verified to meet the acceptance criteria.

# Dynamic Payment Method Update Based on Frequency

Type: CORRECT\_VALUE

Detailed description: As a user, I want the system to dynamically update the available payment methods based on the selected frequency and other conditions, so that I can choose the appropriate payment method for my needs.  
  
Acceptance criteria:  
1. When the frequency is '4' or '12', or the frequency is '2' and the FSC code is not one of '2000003024', '7000005464', '7000002372', or '7000000702':  
 - The available payment methods should be:  
 - NACH  
 - Auto Debit Instruction (ADI)  
 - EMANDATE  
 - Credit Card - Standing instruction (CCSI)  
 - SSS  
2. For all other frequencies:  
 - The available payment methods should be:  
 - Cash/Cheque/DD  
 - SSS  
 - Direct Debit  
 - ECS  
 - Credit Card - Standing instruction (CCSI)  
 - Group Cash  
 - Group Cheque  
 - Group Direct Credit  
 - Bill Junction - ECS  
 - Auto Debit Instruction (ADI)  
 - BG  
 - NACH  
 - EMANDATE  
3. If the frequency is '01':  
 - The payment method selection should be disabled.  
4. If the frequency is not '01' and the selected payment method is not 'CCSI' or 'EMAND', and the ADI flag is not 'Y':  
 - The payment method selection should be enabled.  
  
Definition of Done:  
- The system dynamically updates the payment methods based on the selected frequency and other conditions.  
- The payment method selection is disabled or enabled based on the specified conditions.  
- The changes are tested and verified to ensure they meet the acceptance criteria.  
- The user interface reflects the updated payment methods correctly.  
  
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 Joint Life Selection

Type: CORRECT\_VALUE

Title: Validate Joint Life Selection  
  
Acceptance Criteria:  
1. When the user selects "JL" (Joint Life) from the "Joint Life" dropdown list, the system should check if the insured person (IP) and policyholder (PH) are the same.  
2. If the insured person (IP) and policyholder (PH) are the same, the system should display an error message: "For Joint Life IP and PH cannot be same."  
  
Definition of Done:  
1. The "Joint Life" field should be a dropdown list with the options "IP" and "PH".  
2. The system should validate the selection when "JL" is chosen.  
3. An error message should be displayed if the insured person (IP) and policyholder (PH) are the same when "Joint Life" is selected.  
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 specific database queries.

# Implement Dropdown for Loan Type Selection

Type: CORRECT\_VALUE

Detailed description: As a user, I want to select the type of loan from a predefined list so that I can accurately categorize the loan type for further processing.  
  
Acceptance criteria:  
1. The user should be able to see a dropdown list labeled "Type of Loan" on the form.  
2. The dropdown list should contain the following options:  
 - 5  
 - 8  
 - SSS  
 - No  
3. The dropdown list should be positioned appropriately on the form and should be easily accessible.  
4. The user should be able to select an option from the dropdown list.  
5. The selected option should be saved and updated in the system.  
  
Definition of Done:  
- The dropdown list for "Type of Loan" is implemented and visible on the form.  
- The dropdown list contains all the specified options.  
- The user can select an option from the dropdown list.  
- The selected option is saved and updated in the system without any errors.  
- The form layout is user-friendly and meets the specified design requirements.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include specific database queries.

# Enable and Navigate Table Field Mapping on Repayment Period Input

Type: CORRECT\_VALUE

Title: Enable and Navigate Table Field Mapping on Repayment Period Input  
  
Acceptance Criteria:  
1. When the user inputs the repayment period, the system should:  
 - Set the IC\_CODE field in the table field mapping to the value of the FSC\_CODE field in the CORRECT\_VALUE block.  
 - Enable the following fields in the table field mapping:  
 - LEAD\_SUB\_CODE  
 - SUB\_NAME  
 - REFERAL\_ID  
 - REFERAL\_NAME  
 - EMPLOYEE\_ID  
 - Automatically navigate to the LEAD\_SUB\_CODE field in the table field mapping.  
  
Definition of Done:  
- The repayment period input functionality is implemented and tested.  
- The specified fields in the table field mapping are enabled upon input of the repayment period.  
- The system navigates to the LEAD\_SUB\_CODE field after enabling the fields.  
- 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 CRUD operations or SQL queries.

# Validate and Update Portfolio Strategy

Type: CORRECT\_VALUE

Detailed description: As a user, I want the system to validate and update the portfolio strategy based on specific conditions and product definitions, so that the correct portfolio strategy is applied and appropriate fund details are managed accordingly.  
  
Acceptance criteria:  
1. When the portfolio strategy is 'AT' and the frequency is '12', the system should display an error message indicating that the Auto Transfer Portfolio is not applicable for Monthly Mode.  
2. If the product definition is 'MONEY\_SECURE\_PLAN', the portfolio strategy should be set to 'IS'.  
3. If the product definition is 'PRINCIPAL\_GAIN', the portfolio strategy should be set to 'WL'.  
4. If the portfolio strategy is not null and not 'SO', the system should:  
 - Navigate to the 'AZBJ\_SSO\_FUND\_DETAILS' block.  
 - Clear the block.  
 - Disable the 'CORRECT\_FUNDLIST', 'CORRECT\_APPORT', and 'CORRECT\_FUND\_ID' fields if they are enabled.  
5. If the portfolio strategy is 'SO', the system should:  
 - Enable the 'CORRECT\_FUNDLIST' and 'CORRECT\_APPORT' fields if they are disabled.  
6. If the product code is in (307, 316, 331, 351, 353) and the portfolio strategy is in ('TB', 'AT'), the system should:  
 - Navigate to the 'UL\_AT\_PORTFOL' block.  
 - Clear the block without validation.  
 - Navigate to the 'UL\_AT\_PORTFOL\_2' block.  
 - Clear the block without validation.  
 - Navigate to the 'AZBJ\_FUND\_DETAILS' block.  
 - Enable the 'APPORTIONMENT' field for all records.  
7. If the product code is in (307, 316, 331) and the portfolio strategy is in ('TB', 'WL', 'CP'), the 'CMD\_FUND\_VAL' field should be disabled; otherwise, it should be enabled.  
8. If the product code 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 SSO Option is applicable only for Single premium.  
9. The system should call the 'AZBJ\_WOL\_FUND\_POPULATE' procedure to populate fund details based on the portfolio strategy.  
  
Definition of Done:  
- The system correctly validates and updates the portfolio strategy based on the specified conditions.  
- Appropriate error messages are displayed when conditions are not met.  
- Fund details are managed correctly based on the portfolio strategy.  
- All acceptance criteria are met and tested.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- The provided XML content does not include any direct database queries that can be executed without modification. Therefore, no DB queries are included in this user story.

# Input and Update Witness Date

Type: CORRECT\_VALUE

Title: Input and Update Witness Date  
  
Acceptance Criteria:  
1. The witness date field should accept dates in the format DD/MM/YYYY.  
2. The field should allow both insertion and updating of dates.  
3. Upon entering a date, the system should automatically navigate to the consent information section.  
  
Definition of Done:  
1. The witness date field is displayed on the form.  
2. The field accepts and displays dates in the specified format.  
3. Users can insert and update the date.  
4. After entering a date, the system navigates to the consent information section.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include specific database queries.

# Handle Renewal Payment Mode Changes

Type: CORRECT\_VALUE

Detailed description: As a user, I want the system to validate and handle the renewal payment mode changes to ensure that the correct questions are populated and appropriate messages are displayed based on the selected payment mode.  
  
Acceptance criteria:  
1. When the renewal payment mode is changed to or from 'ECS' or 'ADI', the system should:  
 - Navigate to the 'QC Questions' section.  
 - Clear the current block.  
 - Display a warning message: "Renewal paymode has been changed, Please populate QC questions again."  
  
2. The system should check specific conditions related to the renewal payment mode:  
 - If the renewal payment mode is not 'DD', 'ECS', 'ADI', or 'CCSI' and the frequency is not '01', the system should display an error message: "Please Select renewal pay mode as RDDI/ECS/ADI/CCSI for SCB channel."  
  
3. The system should perform the following database checks:  
 - Check if there are any system constants with type 'SCB' and code 'SCB\_PAYMODE' that match the FSC code and are not expired.  
 - Check if there are any application bypass details for the given application number with a bypass flag set to 'Y'.  
 - Check if there are any card cases for the given application number with a bank reference code containing 'SCBCC'.  
  
4. If there are card cases, the system should:  
 - Set the 'cards\_cases' control to 'Cards telesales case'.  
 - Make the 'cards\_cases' control visible.  
  
5. The system should update the renewal mode parameter with the selected renewal payment mode and populate the payment mode details.  
  
Definition of Done:  
- The system correctly handles the renewal payment mode changes and displays appropriate messages.  
- The system performs the necessary database checks and updates the UI elements accordingly.  
- The renewal mode parameter is updated, and payment mode details are populated.  
- All acceptance criteria are met, and the functionality is tested and verified.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Check system constants:  
 ```sql  
 SELECT COUNT(ROWNUM)  
 INTO v\_scb\_cnt  
 FROM azbj\_system\_constants  
 WHERE sys\_type = 'SCB' AND sys\_code = 'SCB\_PAYMODE'  
 AND :correct\_value.fsc\_code LIKE char\_value || '%'  
 AND end\_date IS NULL;  
 ```  
  
- Check application bypass details:  
 ```sql  
 SELECT COUNT(ROWNUM)  
 INTO v\_appln\_cnt  
 FROM azbj\_application\_bypass\_det  
 WHERE application\_no = :correct\_value.appln\_no  
 AND bypass\_flag = 'Y';  
 ```  
  
- Check card cases:  
 ```sql  
 SELECT COUNT(ROWNUM)  
 INTO v\_card\_cases\_cnt  
 FROM azbj\_phub\_scrutiny\_prop  
 WHERE application\_no = :correct\_value.appln\_no  
 AND UPPER(bank\_refcode\_bpc) LIKE '%SCBCC%';  
 ```

# PH Signature Button Functionality

Type: CORRECT\_VALUE

Title: PH Signature Button Functionality  
  
Acceptance Criteria:  
1. When the "PH Signature" button is pressed, the system should display the PH signature.  
2. The count of PH signatures should be incremented by 1 each time the button is pressed.  
  
Definition of Done:  
- The "PH Signature" button is functional and increments the PH signature count.  
- The PH signature is displayed upon pressing the button.  
- 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.

# Implement Annuity Type Dropdown

Type: CORRECT\_VALUE

Title: Implement Annuity Type Dropdown  
  
Acceptance Criteria:  
1. The annuity type field should be a dropdown list with the following options:  
 - "5"  
 - "8"  
 - "SL"  
2. The dropdown list should be disabled by default.  
3. The field should be located at the specified position on the form.  
4. The field should have a white background and black text.  
5. The field should be labeled "Annuity Type" with the label aligned to the right.  
6. The label should be bold and in the "MS Sans Serif" font.  
  
Definition of Done:  
- The annuity type dropdown list is implemented with the specified options.  
- The dropdown list is disabled by default.  
- The field and its label are positioned and styled as specified.  
- The functionality 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.

# Handle Monthly Renewal Frequency

Type: CORRECT\_VALUE

Title: Handle Monthly Renewal Frequency  
  
Acceptance Criteria:  
1. When the renewal frequency is set to monthly, the system should navigate to the questions section and execute the necessary logic to populate the questions.  
2. The system should handle any exceptions that occur during this process and ensure that the renewal frequency is correctly set back to monthly if an error occurs.  
  
Definition of Done:  
- The renewal frequency can be set to monthly.  
- The system navigates to the questions section and executes the logic to populate the questions.  
- Any exceptions are handled gracefully, and the renewal frequency is correctly set back to monthly if an error occurs.  
- The feature is tested and verified to work as expected.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- The provided XML content includes a procedure `populate\_questions` that queries and manipulates data from the `azbj\_bbu\_questions` and `azbj\_fcf\_questionnaire` tables. The procedure uses a cursor to fetch and process questions based on certain conditions and updates the `azbj\_questionnaire` table accordingly.

# Fetch and Display Bank Details Based on IFSC Code

Type: CORRECT\_VALUE

Title: Fetch and Display Bank Details Based on IFSC Code  
  
Acceptance Criteria:  
1. When the user enters an IFSC code and moves to the next field, the system should automatically fetch and display the bank name, bank 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 to select the correct IFSC code from a list of values (LOV).  
3. Once the bank details are fetched, the fields for bank name and bank branch should be set to read-only to prevent further modifications.  
4. The system should navigate to the next field after fetching and displaying the bank details.  
  
Definition of Done:  
- The functionality to fetch and display bank details based on the entered IFSC code is implemented and tested.  
- The system prompts the user with an LOV if the IFSC code is not found.  
- The bank name and bank branch fields are set to read-only after fetching the details.  
- The system successfully navigates to the next field after displaying the bank details.  
- All acceptance criteria are met and verified through testing.  
  
DB queries for Table reference CRUD operations only (With Usage):  
```sql  
SELECT BANK\_NAME, BANK\_BRANCH, BANK\_MICR  
 INTO :CORRECT\_VALUE.BANK\_NAME,  
 :CORRECT\_VALUE.BANK\_BRANCH,  
 :CORRECT\_VALUE.MICR\_CODE  
 FROM azbj\_bank\_ifsc\_detail  
 WHERE BANK\_IFSC = :CORRECT\_VALUE.IFSC\_CODE;  
```

# Navigation based on Nominee's Age

Type: CORRECT\_VALUE

Title: Navigation based on Nominee's Age  
  
Acceptance Criteria:  
1. When the "Next Tab" button is pressed, the system should check the age of the nominee.  
2. If the nominee's age is greater than 18, the system should navigate to the "Nominee Name" field.  
3. If the nominee's age is 18 or less, the system should navigate to the "Questionnaire" section.  
4. The system should enable or disable nominee-related fields based on the nominee's age.  
  
Definition of Done:  
- The "Next Tab" button functionality is implemented and tested.  
- The system correctly navigates to the appropriate fields based on the nominee's age.  
- Nominee-related fields are enabled or disabled as per the business logic.  
- 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 LA Study Standard

Type: CORRECT\_VALUE

Detailed description: As a user, I want to select a standard from a predefined list of options in the "LA Study Standard" field so that I can ensure the correct value is recorded for the study.  
  
Acceptance criteria:  
1. The "LA Study Standard" field should display a dropdown list with the following options:  
 - 5  
 - 8  
 - 10  
 - LA  
 - No  
2. Upon selecting an option, the system should navigate to the next item in the form, specifically the "ML Relation" field in the "Insurance" section.  
  
Definition of Done:  
- The dropdown list in the "LA Study Standard" field displays all the specified options.  
- Selecting an option from the dropdown list successfully navigates the user to the "ML Relation" field in the "Insurance" section.  
- 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 specific database queries.

# Navigation Logic for 'Next Tab' Button

Type: CORRECT\_VALUE

Detailed description: As a user, I want to navigate through different sections of the application based on specific conditions, so that I can efficiently manage and update the relevant information.  
  
Acceptance criteria:  
1. When the "Next Tab" button is pressed:  
 - If the `v\_unitlink\_flag` is either 'Y' or 'P':  
 - The system should enable the 'UNIT\_LINK' tab.  
 - If the 'PORTFOLIO\_STRATEGY' field is enabled, the system should navigate to this field and trigger the 'WHEN-LIST-CHANGED' event.  
 - If the `v\_unitlink\_flag` is neither 'Y' nor 'P':  
 - The system should enable the 'BANK\_DETAILS' tab.  
 - If the `penny\_drop\_successful` status is 'Penny Drop Successful', the system should navigate to the 'PROPOSAL\_SIGN\_DATE' field.  
 - Otherwise, the system should navigate to the 'BANK\_ACCOUNT\_NO' field.  
  
Definition of Done:  
- The navigation logic is implemented and tested.  
- The system correctly enables the appropriate tabs based on the `v\_unitlink\_flag`.  
- The system navigates to the correct fields based on the conditions specified.  
- 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 CRUD operations.

# Product ID Selection and Validation

Type: CORRECT\_VALUE

Title: Product ID Selection and Validation  
  
Acceptance Criteria:  
1. When a product ID is selected, the system should:  
 - Check if the product ID exists in the product master table.  
 - If the product ID exists, populate the package code field with the corresponding package code from the product master table.  
 - Navigate to the premium field and then to the portfolio strategy field, clearing the portfolio strategy field.  
 - If the product definition is 'LIFELONG\_ASSURE' or 'LIFE\_SECURE', calculate the benefit term as 100 minus the entry age.  
 - If the product definition is 'LONGLIFE\_GOAL', calculate the benefit term as 99 minus the entry age.  
 - Check if the product ID exists in the solution configuration master table with a product type of 'M'. If it does and the solution name is less than 3, enable the solution name field; otherwise, disable it.  
 - Set the unit link flag and portfolio strategy based on the product code.  
 - If the unit link flag is 'Y' or 'P', enable the unit link tab.  
2. When a new item instance is created:  
 - If the unit link flag is not 'Y' or 'P' and the premium field is not null, disable the insert and update properties of the IP area free text field.  
  
Definition of Done:  
- The system correctly performs all the specified validations and calculations when a product ID is selected.  
- The system correctly enables or disables fields based on the specified conditions.  
- The system correctly handles new item instances as specified.  
- All functionalities are tested and verified to work as expected.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Check if the product ID exists in the product master table:  
 ```sql  
 SELECT COUNT() INTO v\_cnt FROM azbj\_package\_master WHERE product\_id = :correct\_value.product\_code;  
 ```  
  
- Populate the package code field:  
 ```sql  
 SELECT package\_code INTO :CORRECT\_VALUE.package\_code FROM azbj\_package\_master WHERE product\_id = :correct\_value.product\_code;  
 ```  
  
- Check if the product ID exists in the solution configuration master table:  
 ```sql  
 SELECT COUNT(product\_id) INTO v\_sol\_cnt FROM AZBJ\_SOLUTION\_CONFIG\_MST WHERE product\_id = :correct\_value.product\_code AND PRODUCT\_TYPE = 'M';  
 ```  
  
- Fetch product details for LOV:  
 ```sql  
 SELECT FROM (  
 SELECT product\_ext\_reference prod\_desc, a.product\_id, UPPER(description) description  
 FROM cfg\_v\_oc\_products\_api a, azbj\_product\_launch\_date b  
 WHERE a.product\_id >= 3  
 AND a.product\_id NOT IN (5, 7)  
 AND a.product\_id = b.product\_id  
 AND (b.closure\_date IS NULL OR MONTHS\_BETWEEN(SYSDATE, closure\_date) < 3)  
 AND a.product\_id NOT IN (  
 SELECT TO\_NUMBER(SUBSTR(char\_value, 2, LENGTH(char\_value) - 2))  
 FROM azbj\_system\_constants  
 WHERE UPPER(sys\_type) = 'CASHIER' AND UPPER(sys\_code) = 'PRODUCT\_BLOCKED' AND date\_value < SYSDATE  
 )  
 AND a.product\_id IN (  
 SELECT PRODUCT\_ID  
 FROM AZBJ\_PRODUCT\_COVERS  
 WHERE (CASE  
 WHEN NVL(IL\_FLAG, 'N') IN ('Y', 'P') AND NVL(GROUP\_FLAG, 'N') <> 'Y' THEN 'UL'  
 WHEN NVL(IL\_FLAG, 'N') NOT IN ('Y', 'P') AND NVL(GROUP\_FLAG, 'N') <> 'Y' THEN 'TR'  
 WHEN NVL(GROUP\_FLAG, 'N') = 'Y' THEN 'GRP'  
 END) = (  
 SELECT (CASE  
 WHEN NVL(IL\_FLAG, 'N') IN ('Y', 'P') AND NVL(GROUP\_FLAG, 'N') <> 'Y' THEN 'UL'  
 WHEN NVL(IL\_FLAG, 'N') NOT IN ('Y', 'P') AND NVL(GROUP\_FLAG, 'N') <> 'Y' THEN 'TR'  
 WHEN NVL(GROUP\_FLAG, 'N') = 'Y' THEN 'GRP'  
 END)  
 FROM AZBJ\_PRODUCT\_COVERS  
 WHERE PRODUCT\_ID = :correct\_value.product\_code  
 )  
 )  
 ORDER BY a.product\_id  
 )  
 ```

# Navigation to Next Tab Based on Conditions

Type: CORRECT\_VALUE

Detailed description: As a user, I want to navigate to the next tab based on specific conditions related to product codes and user attributes, so that I can access the relevant information or sections efficiently.  
  
Acceptance criteria:  
1. When the "Next Tab" button is pressed:  
 - If the product code is 315 and the package code is either 'FAMILY\_HEALTH\_CARE' or 'FAMILY\_HEALTH\_CARE\_PLUS':  
 - Refresh the policy member information.  
 - Enable the 'FAMILY\_MEMBERS' tab.  
 - Navigate to the 'POL\_MEMB\_NEXT\_TAB' item.  
 - Otherwise:  
 - Enable or disable the nominee details based on specific conditions.  
 - If the user's age is greater than 18 and their occupation is not 'T':  
 - Enable the 'NOMINEE\_HEALTH\_DTLS' tab.  
 - Navigate to the 'CORRECT\_VALUE' item in the 'AZBJ\_QUESTIONNAIRE' section.  
 - Otherwise:  
 - Enable the 'MLQ' tab.  
 - Navigate to the 'MLQ\_FATHER\_INCOME' item.  
  
Definition of Done:  
- The "Next Tab" button should correctly navigate to the appropriate tab based on the specified conditions.  
- The relevant tabs should be enabled or disabled as per the conditions.  
- The navigation and tab enabling/disabling should be tested and verified for accuracy.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Not applicable as the provided XML content does not include any direct database queries.

# Money Back Option Selection

Type: CORRECT\_VALUE

Title: Money Back Option Selection  
  
Acceptance Criteria:  
1. The money back option should be presented as a list with the following values:  
 - 5  
 - 8  
 - 10  
 - Self  
 - Other  
2. The list should be displayed on the "Cover Details" tab.  
3. The selected value should be stored and retrievable for future reference.  
4. The list should be case-insensitive and accept only uppercase inputs.  
5. The list should have a tooltip and hint text displaying "[Self/Policyholder/Other]".  
  
Definition of Done:  
1. The money back option list is implemented and displayed correctly on the "Cover Details" tab.  
2. Users can select a value from the list, and the selected value is stored in the database.  
3. The list values are displayed in uppercase.  
4. Tooltip and hint text are correctly displayed as "[Self/Policyholder/Other]".  
5. 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 CRUD operations mentioned in the provided XML content.

# Implement Next Tab Navigation and Conditional Logic

Type: CORRECT\_VALUE

Title: Implement Next Tab Navigation and Conditional Logic  
  
Acceptance Criteria:  
1. When the "Next Tab" button is pressed, the system should check if the product code is 315. If it is, the system should refresh the policy member details.  
2. The system should enable or disable the nominee details tab based on the age and occupation of the insured person (IP). If the IP's age is greater than 18 and the occupation is not 'T', the nominee health details tab should be enabled, and the focus should move to the questionnaire section. Otherwise, the MLQ tab should be enabled, and the focus should move to the father's income section.  
  
Definition of Done:  
- The "Next Tab" button functionality is implemented and tested.  
- The system correctly refreshes policy member details when the product code is 315.  
- The nominee details tab is enabled or disabled based on the specified conditions.  
- The focus moves to the appropriate section based on the conditions.  
- All acceptance criteria are met and verified through testing.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Not applicable as the provided XML content does not include specific CRUD operations or SQL queries.

# Manage Signature Confidence Details

Type: CORRECT\_VALUE

Title: Manage Signature Confidence Details  
  
Acceptance Criteria:  
1. When the "Sign Confid" 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 name and total confidence from the database for the given application number.  
4. The retrieved data should be displayed in the "Signature Confidence" section, with each record shown in a new row.  
5. The system should navigate to the first record after populating the data.  
  
Definition of Done:  
- The "Sign Confid" button functionality is implemented and tested.  
- The system correctly navigates to the "Signature Confidence" section upon button press.  
- Existing data in the "Signature Confidence" section is cleared before new data is populated.  
- The system retrieves and displays the correct signature name and total confidence for the given application number.  
- The system navigates to the first record after data population.  
- All acceptance criteria are met and verified through testing.  
  
DB queries for Table reference CRUD operations only(With Usage):  
```sql  
SELECT SIGNATURENAME, TotalConf  
FROM balic.ng\_asv\_results  
WHERE APPLICATIONNO = :CORRECT\_VALUE.APPLN\_NO;  
```  
- This query retrieves the signature name and total confidence for a given application number from the `balic.ng\_asv\_results` table.

# Navigate to Previous Item Based on Conditions

Type: CORRECT\_VALUE

Title: Navigate to Previous Item Based on Conditions  
  
Acceptance Criteria:  
1. When the current block is 'CORRECT' and the current item matches a specific list name and block name, the system should fetch the previous item from the `azbj\_new\_bbu\_field\_map\_gr` table based on the proposal type and navigate to that item.  
2. If the conditions are not met, the system should navigate to an item prefixed with 'L\_' followed by the current item name.  
  
Definition of Done:  
- The system correctly navigates to the previous item based on the specified conditions.  
- The navigation logic is tested and verified to ensure it works as expected.  
- The feature is documented and reviewed for accuracy.  
  
DB queries for Table reference CRUD operations only(With Usage):  
```sql  
SELECT key\_prev\_item  
INTO v\_key\_prev  
FROM azbj\_new\_bbu\_field\_map\_gr  
WHERE LIST\_NAME = :system.current\_item  
AND LIST\_BLOCK\_NAME = :system.current\_block  
AND proposal\_type = :proposalform.proposal\_type;  
```

# Navigation to Previous Item Based on Conditions

Type: CORRECT\_VALUE

Title: Navigation to Previous Item Based on Conditions  
  
Acceptance Criteria:  
1. If the current block is 'CORRECT', the system should:  
 - Retrieve the previous item name from the `azbj\_new\_bbu\_field\_map\_gr` table where `LIST\_NAME` matches the current item, `LIST\_BLOCK\_NAME` matches the current block, and `proposal\_type` matches the proposal type.  
 - Navigate to the retrieved item.  
2. If the current block is not 'CORRECT', the system should:  
 - Navigate to the item prefixed with 'L\_' followed by the current item name.  
  
Definition of Done:  
- The system correctly navigates to the previous item based on the specified conditions.  
- The navigation logic is tested and verified to work as expected.  
- The feature is documented and reviewed.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Retrieve the previous item name:  
 ```sql  
 SELECT key\_prev\_item  
 INTO v\_key\_prev  
 FROM azbj\_new\_bbu\_field\_map\_gr  
 WHERE LIST\_NAME = :system.current\_item  
 AND LIST\_BLOCK\_NAME = :system.current\_block  
 AND proposal\_type = :proposalform.proposal\_type;  
 ```

# Mobile Number Validation

Type: CORRECT\_VALUE

Title: Mobile Number Validation  
  
Acceptance Criteria:  
1. The mobile number field should only accept numeric values.  
2. The mobile number must be exactly 10 digits long.  
3. If the mobile number is not numeric or not 10 digits long, an error message should be displayed to the user indicating the issue.  
4. The field should not allow updates or insertions directly by the user.  
  
Definition of Done:  
1. The mobile number field is implemented and integrated into the application.  
2. Validation logic is in place to ensure the mobile number is numeric and exactly 10 digits long.  
3. Error messages are displayed correctly when the validation fails.  
4. The field is set to read-only to prevent direct user modifications.  
5. All acceptance criteria are met and tested successfully.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- No direct database queries are provided in the XML content for CRUD operations. The logic primarily involves validation and navigation within the application.

# Navigation to Previous Item Based on Conditions

Type: CORRECT\_VALUE

Detailed description: As a user, I want the system to navigate to the previous item in the form based on specific conditions, so that I can efficiently move through the form fields.  
  
Acceptance criteria:  
1. When the current block is 'CORRECT', the system should:  
 - Retrieve the previous item name from the database table `azbj\_new\_bbu\_field\_map\_gr` where `LIST\_NAME` matches the current item, `LIST\_BLOCK\_NAME` matches the current block, and `proposal\_type` matches the proposal type.  
 - Navigate to the retrieved item.  
2. If the current block is not 'CORRECT', the system should:  
 - Navigate to the item prefixed with 'L\_' followed by the current item name.  
  
Definition of Done:  
- The system correctly navigates to the previous item based on the specified conditions.  
- The navigation logic is tested and verified to work as expected.  
- The feature is documented and reviewed.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Retrieve previous item name:  
 ```sql  
 SELECT key\_prev\_item  
 INTO v\_key\_prev  
 FROM azbj\_new\_bbu\_field\_map\_gr  
 WHERE LIST\_NAME = :system.current\_item  
 AND LIST\_BLOCK\_NAME = :system.current\_block  
 AND proposal\_type = :proposalform.proposal\_type;  
 ```

# Navigation to Previous Item Based on Conditions

Type: CORRECT\_VALUE

Title: Navigation to Previous Item Based on Conditions  
  
Acceptance Criteria:  
1. When the current block is 'CORRECT', the system should:  
 - Retrieve the previous item name from the `azbj\_new\_bbu\_field\_map\_gr` table where `LIST\_NAME` matches the current item, `LIST\_BLOCK\_NAME` matches the current block, and `proposal\_type` matches the proposal type.  
 - Navigate to the retrieved item.  
2. If the current block is not 'CORRECT', the system should:  
 - Navigate to the item prefixed with 'L\_' followed by the current item name.  
  
Definition of Done:  
- The system correctly navigates to the previous item based on the specified conditions.  
- The navigation logic is tested and verified to work as expected.  
- The feature is documented and reviewed.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Retrieve the previous item name:  
 ```sql  
 SELECT key\_prev\_item  
 INTO v\_key\_prev  
 FROM azbj\_new\_bbu\_field\_map\_gr  
 WHERE LIST\_NAME = :system.current\_item  
 AND LIST\_BLOCK\_NAME = :system.current\_block  
 AND proposal\_type = :proposalform.proposal\_type;  
 ```

# Gender Field Selection and Navigation

Type: CORRECT\_VALUE

Title: Gender Field Selection and Navigation  
  
Acceptance Criteria:  
1. The gender field should display a list of predefined values (e.g., "M" for Male, "F" for Female, "O" for Other).  
2. The field should be read-only and not allow direct input or modification by the user.  
3. When navigating through the form, the system should automatically move to the next or previous field based on the current field and block.  
4. The system should load relevant data from an image file based on the current field and block.  
5. The system should update the visual attributes of the field based on the current block and item.  
  
Definition of Done:  
1. The gender field displays a list of predefined values.  
2. The field is read-only and cannot be modified directly by the user.  
3. Navigation through the form works as expected, moving to the correct next or previous field.  
4. Relevant data is loaded from an image file based on the current field and block.  
5. The visual attributes of the field are updated correctly based on the current block and item.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- No direct database queries are provided in the XML content for CRUD operations.

# Validate and Calculate GMI Value for Product Code 299

Type: CORRECT\_VALUE

Title: Validate and Calculate GMI Value for Product Code 299  
  
Acceptance Criteria:  
1. When the product code is 299 and the GMI value is null, the system should display a warning message prompting the user to enter a valid GMI value.  
2. If a valid GMI value is entered, the system should calculate the sum assured by multiplying the GMI value by 144.  
3. The sum assured field should be enabled if it was previously disabled.  
  
Definition of Done:  
- The system correctly displays a warning message when the GMI value is null and the product code is 299.  
- The sum assured is accurately calculated and displayed based on the GMI value.  
- The sum assured field is enabled and editable if it was previously disabled.  
  
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 GSTN Number Field

Type: CORRECT\_VALUE

Detailed description: As a user, I want to ensure that the GSTN Number entered in the system is valid and adheres to specific format rules, so that data integrity is maintained and errors are minimized.  
  
Acceptance criteria:  
1. The GSTN Number field should accept a maximum of 15 characters.  
2. The GSTN Number must be in uppercase.  
3. The system should display an error message if the GSTN Number is not exactly 15 characters long.  
4. The system should display an error message if the GSTN Number contains any special characters.  
5. The error messages should be clear and indicate the specific issue with the GSTN Number.  
  
Definition of Done:  
1. The GSTN Number field is implemented with a maximum length of 15 characters.  
2. The GSTN Number is automatically converted to uppercase upon entry.  
3. Validation logic is implemented to check the length of the GSTN Number and ensure it is exactly 15 characters.  
4. Validation logic is implemented to check for the presence of special characters in the GSTN Number.  
5. Appropriate error messages are displayed when validation fails.  
6. The feature is tested and verified to ensure it meets the acceptance criteria.  
7. The feature is reviewed and approved by stakeholders.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Not applicable as the provided XML content does not include specific CRUD operations or table references.

# Validate GSTN Number Input

Type: CORRECT\_VALUE

Title: Validate GSTN Number Input  
  
Acceptance Criteria:  
1. The GSTN number must be exactly 15 characters long.  
2. The GSTN number must not contain any special characters.  
3. If the GSTN number is not valid, an error message should be displayed indicating the specific issue:  
 - "Please enter a valid GSTN Number" if the length is not 15 characters.  
 - "Special characters are not allowed for GSTN Number" if special characters are present.  
  
Definition of Done:  
1. The "GSTN No" field accepts input and validates it according to the specified criteria.  
2. Appropriate error messages are displayed when the input does not meet the criteria.  
3. The system prevents the user from proceeding to the next field until a valid GSTN number 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.

# Phone Number Field Validation and Navigation

Type: CORRECT\_VALUE

Detailed description: As a user, I want to enter my phone number in a specific format so that the system can validate and store it correctly.  
  
Acceptance criteria:  
1. The phone number field should only accept characters in uppercase.  
2. The phone number should be entered in the format: [STD CODE][NUMBER].  
3. The field should display a tooltip and hint with the message: "Please Enter Phone Number in format: [STD CODE][NUMBER]".  
4. The phone number field should not allow insertion or updates directly by the user.  
5. When navigating to the previous item, if the current block is 'CORRECT', the system should fetch the previous item from a mapping table and navigate to it. Otherwise, it should navigate to an item prefixed with 'L\_'.  
6. When navigating to the next item, the system should clear any visual attributes of the current item and navigate to the email field.  
7. Upon entering the phone number field, the system should load the field data from an image based on the current item.  
  
Definition of Done:  
1. The phone number field is implemented with the specified format and restrictions.  
2. Tooltips and hints are displayed correctly.  
3. Navigation between items works as described.  
4. The field data is loaded from an image when the field is accessed.  
5. All acceptance criteria are met and tested.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- The following query is used to fetch the previous item when navigating to the previous field:  
 ```sql  
 SELECT key\_prev\_item  
 INTO v\_key\_prev  
 FROM azbj\_new\_bbu\_field\_map\_gr  
 WHERE LIST\_NAME = :system.current\_item  
 AND LIST\_BLOCK\_NAME = :system.current\_block  
 AND proposal\_type = :proposalform.proposal\_type;  
 ```  
  
- The following query is used to get the maximum strip number for loading field data from an image:  
 ```sql  
 SELECT MAX(strip\_no)  
 INTO v\_strip\_no  
 FROM azbj\_strip\_field\_mapping a  
 WHERE block\_name = v\_current\_block  
 AND field\_name = v\_current\_item  
 AND proposal\_type = PK\_VARS.v\_proposal\_type  
 AND EXISTS (  
 SELECT 1  
 FROM azbj\_proposal\_stripes b  
 WHERE a.proposal\_type = b.proposal\_type  
 AND INSTR(stripe\_string, '~' || a.strip\_no || '~') > 0  
 );  
 ```

# Phone Number Field Validation and Navigation

Type: CORRECT\_VALUE

Title: Phone Number Field Validation and Navigation  
  
Acceptance Criteria:  
1. The phone number field should display a tooltip and hint that instructs the user to enter the phone number in the format: [STD CODE][NUMBER].  
2. The phone number field should automatically convert the input to uppercase.  
3. The phone number field should not allow insertion or updates directly by the user.  
4. When navigating to the previous item, if the current block is 'CORRECT', the system should fetch the previous item from a mapping table and navigate to it. Otherwise, it should navigate to the item prefixed with 'L\_'.  
5. When navigating to the next item, the system should clear any visual attributes and navigate to the 'APP\_PHONE' field.  
6. The system should load the field from an image when the phone number field is focused.  
  
Definition of Done:  
1. The phone number field displays the correct tooltip and hint.  
2. The phone number field converts input to uppercase.  
3. The phone number field does not allow direct insertion or updates.  
4. Navigation to the previous item works as specified.  
5. Navigation to the next item works as specified.  
6. The field is loaded from an image when focused.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- The query to fetch the previous item:  
 ```sql  
 SELECT key\_prev\_item  
 INTO v\_key\_prev  
 FROM azbj\_new\_bbu\_field\_map\_gr  
 WHERE LIST\_NAME = :system.current\_item  
 AND LIST\_BLOCK\_NAME = :system.current\_block  
 AND proposal\_type = :proposalform.proposal\_type;  
 ```  
  
- The query to get the strip number for loading the field from the image:  
 ```sql  
 SELECT MAX(strip\_no)  
 INTO v\_strip\_no  
 FROM azbj\_strip\_field\_mapping a  
 WHERE block\_name = v\_current\_block  
 AND field\_name = v\_current\_item  
 AND proposal\_type = PK\_VARS.v\_proposal\_type  
 AND EXISTS (  
 SELECT 1  
 FROM azbj\_proposal\_stripes b  
 WHERE a.proposal\_type = b.proposal\_type  
 AND INSTR(stripe\_string, '~' || a.strip\_no || '~') > 0  
 );  
 ```

# Validate Channel Field

Type: CORRECT\_VALUE

Title: Validate Channel Field  
  
Acceptance Criteria:  
1. When the "Channel" field is validated, the system should check if the application is for the desktop module.  
2. If the application is not for the desktop module, the system should:  
 - Query the `azbj\_batch\_items` table to retrieve the channel where `application\_no` matches the current application number and `transaction\_type` is 'FRP', and `PRINT` is not 'C'.  
 - If no channel is found, the system should then query the `azbj\_phub\_scrutiny\_prop\_extn` table to retrieve the channel where `application\_no` matches the current application number.  
3. If the retrieved channel is different from the entered channel, the system should highlight the "Channel" field in red.  
4. If the retrieved channel matches the entered channel, the system should highlight the "Channel" field in green.  
  
Definition of Done:  
- The "Channel" field validation logic is implemented and tested.  
- The system correctly queries the database tables and retrieves the channel information.  
- The "Channel" field is highlighted in red or green based on the validation result.  
- The feature is tested and verified to work as expected in different scenarios.  
  
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 = :correct\_value.appln\_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 = :correct\_value.appln\_no;  
 ```

# Next Tab Navigation and Validations

Type: CORRECT\_VALUE

Detailed description: As a user, I want to navigate to the next tab and perform various validations and actions based on the product code, package code, and other conditions, so that I can ensure the data entered is accurate and complete.  
  
Acceptance criteria:  
1. When the "Next Tab" button is pressed:  
 - If the product code is 315 and the package code is either 'FAMILY\_HEALTH\_CARE' or 'FAMILY\_HEALTH\_CARE\_PLUS', the system should:  
 - Enable the 'FAMILY\_MEMBERS' tab.  
 - Navigate to the 'POL\_MEMB\_NEXT\_TAB' field.  
 - Add or remove spouse details based on the presence of a spouse.  
 - If the product code is 316 and the SPW flag is 'Y' but SPW percentage or frequency is not entered, the system should display an error message prompting the user to enter the SPW percentage and frequency.  
 - Navigate to the 'AZBJ\_RIDER\_DETAILS' block and go to the first record.  
 - If the rider cover code starts with 'L' and the sum assured, benefit term, or premium term in the rider details do not match the main cover, the system should:  
 - Navigate to the 'sum\_assured' field.  
 - Display an error message indicating that the entered values should match the main cover values.  
 - If the solution name is greater than 0, navigate to the 'SUM\_ASSURED' field in the 'SOLUTION\_PRODUCT' block.  
 - If the group product flag is 'Y', enable the 'COVERS\_GROUP' tab and navigate to the 'COVER\_TYPE' field.  
 - If the unit link flag is 'Y' or 'P', enable the 'UNIT\_LINK' tab and:  
 - If the 'PORTFOLIO\_STRATEGY' field is enabled, navigate to it and execute the 'WHEN-LIST-CHANGED' trigger.  
 - Otherwise, navigate to the 'CORRECT\_FUNDNAME' field in the 'AZBJ\_FUND\_DETAILS1' block.  
 - If the product code is 301, reset the 'AZBJ\_SSO\_FUND\_DETAILS' and disable certain fields in 'AZBJ\_SSO\_FUND\_DETAILS1'.  
 - If the unit link flag is not 'Y' or 'P', enable the 'BANK\_DETAILS' tab.  
 - If the penny drop is successful, navigate to the 'PROPOSAL\_SIGN\_DATE' field.  
 - Otherwise, navigate to the 'BANK\_ACCOUNT\_NO' field.  
 - If the master policy is null and the insured person’s age is greater than 18, determine the safe district flag based on the address.  
 - If the safe district flag is 'Y', hide the 'SUSPECTED\_CASE' field and show the 'SAFE\_DISTRICT\_FLAG' field, setting its value to 'SAFE DISTRICT CASE'.  
 - Log debug information and check if the master policy number should disable the loan number field based on system constants.  
  
Definition of Done:  
- The "Next Tab" button performs all the specified actions and validations.  
- Appropriate error messages are displayed when conditions are not met.  
- Navigation to the correct fields and blocks is executed as per the conditions.  
- Debug information is logged correctly.  
- The loan number field is disabled based on the master policy number and system constants.  
  
DB queries for Table reference CRUD operations only(With Usage):  
```sql  
-- Fetch master policy number based on application number  
SELECT a.master\_policy\_no  
INTO v\_master\_policy\_no  
FROM azbj\_proposal\_appln\_det a  
WHERE a.appln\_no = TO\_NUMBER (:correct\_value.appln\_no)  
 AND de\_flag = 'D2';  
  
-- Count system constants for disabling loan number  
SELECT COUNT ()  
INTO v\_cnt  
FROM azbj\_system\_constants  
WHERE sys\_type = 'GROUP' AND sys\_code = 'DISABLE\_LOAN\_NUMBER'  
 AND sys\_desc LIKE '%|' || NVL (:correct\_value.master\_policy\_no, v\_master\_policy\_no) || '|%';  
```

# Validate PH Aadhaar Number

Type: CORRECT\_VALUE

Detailed description: As a user, I want to validate the Aadhaar number for a specific individual (PH) to ensure that the provided Aadhaar details are correct and verified.  
  
Acceptance criteria:  
1. When the "Verify PH Aadhaar" button is pressed, the system should enable the Aadhaar input field if it is currently disabled.  
2. The system should log the application number and Aadhaar details for debugging purposes.  
3. The system should format the date of birth in the 'DD/MON/YYYY' format.  
4. If the Aadhaar number or the new Aadhaar number is provided, the system should:  
 - Collect various details such as application number, mobile number, name, middle name, surname, date of birth, gender, and Aadhaar number.  
 - Validate the Aadhaar data using the collected details.  
5. The system should log the validation flag and Aadhaar details for debugging purposes.  
6. The system should update the Aadhaar flag and Aadhaar number fields with the validated details.  
7. If the Aadhaar number is invalid (null, incorrect length, or invalid flag), the system should navigate to the 'RESULT' block and handle the records accordingly.  
8. If the age proof type is 'AC' or 'ACS' and the age proof ID does not match the Aadhaar number, the system should update the age proof ID with the Aadhaar number.  
9. If the Aadhaar number is valid and the insured check is true, the system should:  
 - Navigate to the 'AML' block.  
 - Update the 'OTHERS' field in the 'AML' block with the Aadhaar number for records with proof type 'AC' or 'ACS'.  
10. The system should disable the Aadhaar input field after validation.  
11. If any error occurs during the process, the system should log the error message for debugging purposes.  
  
Definition of Done:  
- The Aadhaar validation process is implemented and tested.  
- The system correctly enables/disables fields, logs details, and validates Aadhaar data.  
- The system handles invalid Aadhaar numbers and updates related fields accordingly.  
- The system navigates between blocks and updates records as specified.  
- Error handling and logging are implemented for debugging purposes.  
  
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.

# Title Selection for Insured Person

Type: CORRECT\_VALUE

Title: Title Selection for Insured Person  
  
Acceptance Criteria:  
1. The title field should allow the user to select from a predefined list of titles.  
2. The available titles should include "Mr", "Mrs", "IP", "MOTHER", "SON".  
3. The default value for the title field should be "Mrs".  
4. The title field should be editable and allow updates.  
5. The title field should have a maximum length of 5 characters.  
6. The title field should be displayed on the "INSURED\_PERSON" tab.  
7. The title field should be visually styled with a white background, black text, and a specific font and size.  
  
Definition of Done:  
- The title field is implemented and integrated into the user interface.  
- The title field displays the predefined list of titles.  
- The default value "Mrs" is set when the form is initialized.  
- Users can insert and update the title field.  
- The title field adheres to the specified visual styling.  
- The functionality 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 database queries.

# Navigation and Data Loading for Email Field

Type: CORRECT\_VALUE

Title: Navigation and Data Loading for Email Field  
  
Acceptance Criteria:  
1. When the user navigates to the previous item in the 'CORRECT' block, the system should fetch the previous item from the database based on the current item and block name, and navigate to that item.  
2. When the user navigates to the next item, the system should automatically move to the 'SUM\_RS' field in the 'CORRECT\_VALUE' block.  
3. The system should load data from an image file into the current field when the user navigates to the 'APP\_EMAIL' field.  
4. The system should handle different blocks and items, setting visual attributes and loading the correct image based on the proposal type and other conditions.  
  
Definition of Done:  
- The system correctly navigates to the previous item in the 'CORRECT' block based on the database query.  
- The system correctly navigates to the 'SUM\_RS' field in the 'CORRECT\_VALUE' block when moving to the next item.  
- The system loads data from the image file into the 'APP\_EMAIL' field and sets the appropriate visual attributes.  
- The system handles different blocks and items, ensuring the correct image is loaded and displayed based on the proposal type and other conditions.  
- 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):  
- The query to fetch the previous item in the 'CORRECT' block:  
 ```sql  
 SELECT key\_prev\_item  
 INTO v\_key\_prev  
 FROM azbj\_new\_bbu\_field\_map\_gr  
 WHERE LIST\_NAME = :system.current\_item  
 AND LIST\_BLOCK\_NAME = :system.current\_block  
 AND proposal\_type = :proposalform.proposal\_type;  
 ```  
  
- The query to get the strip number for loading the image:  
 ```sql  
 SELECT MAX(strip\_no)  
 INTO v\_strip\_no  
 FROM azbj\_strip\_field\_mapping a  
 WHERE block\_name = v\_current\_block  
 AND field\_name = v\_current\_item  
 AND proposal\_type = PK\_VARS.v\_proposal\_type  
 AND EXISTS (  
 SELECT 1  
 FROM azbj\_proposal\_stripes b  
 WHERE a.proposal\_type = b.proposal\_type  
 AND INSTR(stripe\_string, '~' || a.strip\_no || '~') > 0  
 );  
 ```

# Validate IP Aadhaar Number

Type: CORRECT\_VALUE

Title: Validate IP Aadhaar Number  
  
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. If the e-KYC flag is set to 'Y', the system should display a message indicating that the Aadhaar is already verified and retrieve the Aadhaar reference number.  
3. If the age proof type is 'AC' or 'ACS' and the age proof ID does not match the Aadhaar number, the system should update the age proof ID to the Aadhaar number.  
4. If the insured check returns '0', the system should navigate to the AML block, iterate through the records, and update the AML details with the Aadhaar number where applicable.  
5. If the e-KYC flag is not set to 'Y', the system should format the date of birth and validate the Aadhaar details using the provided information.  
6. The system should update the Aadhaar flag, Aadhaar number, and new Aadhaar number fields with the validated details.  
7. If the Aadhaar number is invalid or the length of the new Aadhaar number is not 12 characters, the system should navigate to the RESULT block and prompt the user to enter a valid Aadhaar number.  
8. If the age proof type is 'AC' or 'ACS' and the age proof ID does not match the Aadhaar number, the system should update the age proof ID to the Aadhaar number.  
9. If the insured check returns '0', the system should navigate to the AML block, iterate through the records, and update the AML details with the Aadhaar number where applicable.  
10. If the Aadhaar number is not null, the system should disable the Aadhaar input field.  
  
Definition of Done:  
- The Aadhaar number validation process is triggered by pressing the "Verify IP Aadhaar" button.  
- The system correctly handles enabling/disabling fields, displaying messages, and updating records based on the provided logic.  
- The system navigates to the appropriate blocks and updates the necessary fields as per the acceptance criteria.  
- The functionality is tested and verified to ensure it meets the user requirements and acceptance criteria.

# Handle Employer Name Input Based on List Selection

Type: CORRECT\_VALUE

Title: Handle Employer Name Input Based on List Selection  
  
Acceptance Criteria:  
1. When the user selects "OTHERS" or "Others" from the list, the system should:  
 - Enable the employer's name input field if it is currently disabled.  
 - Clear any existing value in the employer's name input field.  
 - Display a message prompting the user to enter the employer's name.  
 - Move the cursor to the employer's name input field.  
  
2. When the user selects any other value from the list, the system should:  
 - Populate the employer's name input field with the selected value from the list.  
  
Definition of Done:  
- The system correctly enables and clears the employer's name input field when "OTHERS" or "Others" is selected.  
- The system displays the appropriate message and moves the cursor to the employer's name input field when required.  
- The system populates the employer's name input field with the selected value from the list when a value other than "OTHERS" or "Others" is selected.  
- All functionalities are tested and verified to work as expected without any Oracle Forms-specific terminology or dependencies.

# Handle Employer Name Input Efficiently

Type: CORRECT\_VALUE

Detailed description: As a user, I want the system to handle the input of the employer's name efficiently, ensuring that if the selected employer name is "OTHERS" or "Others", the system prompts me to enter the actual employer name and enables the corresponding input field.  
  
Acceptance criteria:  
1. When the user selects "OTHERS" or "Others" from the employer name list, the system should:  
 - Enable the input field for the actual employer name if it is currently disabled.  
 - Clear any existing value in the actual employer name input field.  
 - Display a warning message prompting the user to enter the employer name.  
 - Automatically move the cursor to the actual employer name input field.  
2. If the user selects any other value from the employer name list, the system should:  
 - Copy the selected value to the actual employer name input field.  
  
Definition of Done:  
- The system correctly handles the selection of "OTHERS" or "Others" by enabling the input field, clearing its value, displaying a warning message, and moving the cursor to the input field.  
- The system correctly copies any other selected value to the actual employer name input field.  
- All acceptance criteria are met and tested.  
- 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 queries.

# Next Tab Button Functionality

Type: CORRECT\_VALUE

Title: Next Tab Button Functionality  
  
Acceptance Criteria:  
1. When the "Next Tab" button is pressed, the system should:  
 - Check the count of specific transactions (FRP, APR) from the database where the payment mode is 'CHQ', 'BCHQ', or 'WCHQ' and the receipt number matches the provided parameter.  
 - If the count is greater than 0 and a specific variable (`v\_ph\_sign\_cnt`) is 0, prompt the user to check the PH Signature and navigate to the PH Signature button.  
 - Enable or disable specific tabs ('AML\_KYC', 'IP\_CP\_MERGING', 'PH\_CP\_MERGING') based on certain conditions.  
 - Navigate to the "Populate Questions" item and execute its associated trigger.  
 - If a specific variable (`v\_grp\_product`) is 'N', set a control variable (`CN\_PARTNER\_TYPE`) to 'P', navigate to it, and execute its associated trigger.  
 - Finally, navigate to the "PH\_NO\_PAN\_CARD" item.  
  
Definition of Done:  
- The "Next Tab" button performs all the specified checks and actions correctly.  
- The system navigates to the appropriate items and executes the necessary triggers.  
- The user is prompted with appropriate messages when conditions are met.  
- The relevant tabs are enabled or disabled based on the specified conditions.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- The query to count specific transactions:  
 ```sql  
 SELECT count(1)   
 INTO v\_chq\_cnt  
 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.transaction\_type IN ('FRP', 'APR')  
 AND NVL(a.PRINT, 'X') <> 'C'  
 AND (a.perm\_receipt\_no = :parameter.perm\_receipt\_no  
 OR a.frp\_receipt\_no = :parameter.perm\_receipt\_no)  
 AND pay\_mode IN ('CHQ', 'BCHQ', 'WCHQ');  
 ```  
  
- The query to populate questions:  
 ```sql  
 SELECT question\_id, sub\_question, answer\_type, MAX(ans1) ans1,  
 MAX(ans2) ans2, question\_desc, form\_question\_no, appln\_no  
 FROM (  
 SELECT b.question\_id, b.sub\_question, a.answer\_type, b.answer ans1,  
 NULL ans2, a.question\_desc, a.form\_question\_no, appln\_no  
 FROM azbj\_bbu\_questions a, azbj\_fcf\_questionnaire b  
 WHERE NVL(a.form\_question\_no, '0') = NVL(b.question\_no, '0')  
 AND a.question\_id = b.question\_id  
 AND a.sub\_question = b.sub\_question  
 AND b.de\_flag = 'D1'  
 AND partner\_no = :p\_partner\_no  
 AND member\_no = :p\_partner\_no  
 AND b.appln\_no = :correct\_value.appln\_no  
 UNION ALL  
 SELECT b.question\_id, b.sub\_question, a.answer\_type, NULL ans1,  
 b.answer ans2, a.question\_desc, a.form\_question\_no, appln\_no  
 FROM azbj\_bbu\_questions a, azbj\_fcf\_questionnaire b  
 WHERE NVL(a.form\_question\_no, '0') = NVL(b.question\_no, '0')  
 AND a.question\_id = b.question\_id  
 AND a.sub\_question = b.sub\_question  
 AND b.de\_flag = 'D2'  
 AND partner\_no = :p\_partner\_no  
 AND member\_no = :p\_partner\_no  
 AND b.appln\_no = :correct\_value.appln\_no  
 )  
 GROUP BY question\_id, sub\_question, answer\_type, question\_desc, form\_question\_no, appln\_no  
 ORDER BY question\_id, sub\_question;  
 ```

# Validate and Process Sub Id Code

Type: CORRECT\_VALUE

Title: Validate and Process Sub Id Code  
  
Acceptance Criteria:  
1. When the "Sub Id Code" is not null, the system should:  
 - Check if the code exists in the `azbj\_v\_agents` table and the agent status is not 'T' or 'TE'.  
 - If the code is invalid, display an error message "Invalid sub id code".  
 - If the code is valid, fetch and display the agent's full name in the corresponding field.  
2. The system should fetch the `channel\_code` from the `azbj\_v\_agents` table based on the `fsc\_code` and populate the "Channel" list with values from the `azbj\_sub\_id\_mapping` table.  
3. For desktop modules, the system should:  
 - Fetch the `sub\_id` from the `azbj\_batch\_items` table based on the `application\_no` and `transaction\_type` 'FRP', excluding records with `PRINT` value 'C'.  
 - If no `sub\_id` is found, fetch it from the `azbj\_phub\_scrutiny\_prop\_extn` table based on the `application\_no`.  
4. The system should compare the fetched `sub\_id` with the input "Sub Id Code":  
 - If they do not match, highlight the field in red.  
 - If they match, highlight the field in green.  
  
Definition of Done:  
- The "Sub Id Code" validation logic is implemented and tested.  
- Error messages and field highlights are displayed correctly based on the validation results.  
- The "Channel" list is populated correctly based on the `channel\_code`.  
- The `sub\_id` is fetched and compared correctly for desktop modules.  
  
DB queries for Table reference CRUD operations only(With Usage):  
```sql  
-- Check if the sub\_id\_code is valid  
SELECT COUNT()  
INTO v\_valid\_cnt  
FROM azbj\_v\_agents  
WHERE reference\_code = UPPER(:correct\_value.sub\_id\_code)  
AND agent\_status NOT IN ('T', 'TE');  
  
-- Fetch the agent's full name  
SELECT full\_name  
INTO :correct\_value.sub\_id\_name  
FROM azbj\_v\_agents  
WHERE reference\_code = UPPER(:correct\_value.sub\_id\_code)  
AND agent\_status NOT IN ('T', 'TE');  
  
-- Fetch the channel\_code  
SELECT channel\_code  
INTO v\_channel  
FROM azbj\_v\_agents  
WHERE reference\_code = UPPER(:correct\_value.fsc\_code);  
  
-- Populate the Channel list  
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(:correct\_value.sub\_id\_code), 3, 2);  
  
-- Fetch the sub\_id for desktop modules  
SELECT sub\_id  
INTO v\_sub\_id  
FROM azbj\_batch\_items  
WHERE application\_no = :correct\_value.appln\_no  
AND transaction\_type = 'FRP'  
AND NVL(PRINT, 'X') <> 'C';  
  
-- Fetch the sub\_id from scrutiny table if not found in batch items  
SELECT sub\_id\_code  
INTO v\_sub\_id  
FROM azbj\_phub\_scrutiny\_prop\_extn  
WHERE application\_no = :correct\_value.appln\_no;  
```

# Title Selection for Policy Holder

Type: CORRECT\_VALUE

Title: Title Selection for Policy Holder  
  
Acceptance Criteria:  
1. The title field should be a dropdown list with the following options:  
 - Mr  
 - Mrs  
 - PH  
 - MOTHER  
 - SL  
2. The default value of the title field should be "Mrs".  
3. The title field should be located at the specified position on the form.  
4. The title field should have a maximum length of 5 characters.  
5. The title field should be displayed on the "POLICY\_HOLDER" tab page.  
6. The title field should be styled with the specified font and colors.  
  
Definition of Done:  
1. The title dropdown list is implemented and displays the correct options.  
2. The default value of the title field is set to "Mrs".  
3. The title field is positioned correctly on the form.  
4. The title field adheres to the maximum length of 5 characters.  
5. The title field is visible on the "POLICY\_HOLDER" tab page.  
6. The title field is styled according to the specified font and 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.

# Implement 'Sub IC Pan' Field in 'Agent' Tab

Type: CORRECT\_VALUE

Title: Implement 'Sub IC Pan' Field in 'Agent' Tab  
  
Acceptance Criteria:  
1. The 'Sub IC Pan' field should be visible and enabled within the 'Agent' tab.  
2. The field should display a list of predefined values when double-clicked.  
3. The list of values should include the following options:  
 - 5  
 - 8  
 - LA  
 - No  
4. The field should restrict input to uppercase characters.  
5. The field should have a maximum length of 30 characters.  
  
Definition of Done:  
1. The 'Sub IC Pan' field is implemented and visible within the 'Agent' tab.  
2. Double-clicking the field displays a list of predefined values.  
3. The list of values includes the specified options.  
4. The field restricts input to uppercase characters.  
5. The field's maximum length is set to 30 characters.  
6. The functionality 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 table references.

# Surrogate Income Proof Selection

Type: CORRECT\_VALUE

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 the selected proof type.  
2. The system should navigate to the specific item related to the selected proof type for further actions.  
3. If an error occurs during the selection process, the system should handle it gracefully without crashing.  
  
Definition of Done:  
1. The surrogate income proof list is populated with predefined values.  
2. Upon selection, the detailed view for the selected proof type is displayed.  
3. The system navigates to the specific item related to the selected proof type.  
4. Error handling is implemented to manage any exceptions during the selection process.  
  
DB queries for Table reference CRUD operations only(With Usage):  
- Not applicable as the provided XML content does not include any direct database queries.

# Implement Account Type Selection

Type: CORRECT\_VALUE

Title: Implement Account Type Selection  
  
Acceptance Criteria:  
1. The account type field should display a list of predefined options.  
2. The list should include the following options:  
 - SON  
 - 5  
 - 8  
 - 1 (LA)  
 - SL (No)  
3. The account type field should be located on the "INSURED\_PERSON" tab.  
4. The field should be editable, allowing users to select an option from the list.  
5. The field should be case-insensitive and convert all input to uppercase.  
6. The field should have a maximum length of 50 characters.  
7. The field should be visually styled with a white background and black text.  
8. The field should be positioned at coordinates (740, 73) on the form.  
  
Definition of Done:  
- The account type field is implemented and displays the predefined list of options.  
- Users can select an option from the list and the selection is saved correctly.  
- The field adheres to the specified visual and positional requirements.  
- The field is tested and verified to ensure it meets all acceptance criteria.

# Enable/Disable Percentage Input Field Based on Accumulate Checkbox and Flag Status

Type: CORRECT\_VALUE

Title: Enable/Disable Percentage Input Field Based on Accumulate Checkbox and Flag Status  
  
Acceptance Criteria:  
1. When the "Accumulate" checkbox is checked:  
 - If the specific flag is set to 'Y', the percentage input field should be enabled.  
 - If the specific flag is not set to 'Y', the percentage input field should remain disabled.  
2. When the "Accumulate" checkbox is unchecked:  
 - The percentage input field should be disabled regardless of the flag's status.  
  
Definition of Done:  
- The functionality should be implemented and tested.  
- The percentage input field should correctly enable or disable based on the checkbox state and the flag's value.  
- The user interface should reflect these changes immediately upon interaction with the checkbox.  
- All related unit tests should pass.  
  
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.

# Display Item Interaction for SP\_SUB\_IC\_NAME

Type: CORRECT\_VALUE

Title: Display Item Interaction for SP\_SUB\_IC\_NAME  
  
Acceptance Criteria:  
1. The display item "SP\_SUB\_IC\_NAME" should be visible and enabled within the "AGENT" tab.  
2. The item should be restricted to uppercase input.  
3. When double-clicked, a list of values should be displayed for selection.  
4. The list of values should include predefined options such as "5", "8", and "LA".  
5. The display item should be editable, allowing for both insertion and updates.  
  
Definition of Done:  
1. The display item "SP\_SUB\_IC\_NAME" is implemented and visible within the "AGENT" tab.  
2. The item correctly restricts input to uppercase.  
3. Double-clicking the item triggers the display of a list of values.  
4. The list of values includes the predefined options "5", "8", and "LA".  
5. The item allows for insertion and updates as required.  
  
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.

# Exit Button Functionality

Type: CORRECT\_VALUE

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 then move to the item labeled "AWG\_PER\_RATES\_LIST" within the "CORRECT\_VALUE" section.  
  
Definition of Done:  
- The "EXIT" button should be functional and perform the specified actions when pressed.  
- The current view and window should be hidden upon pressing the "EXIT" button.  
- The focus should correctly move to the "AWG\_PER\_RATES\_LIST" item in the "CORRECT\_VALUE" section.  
- The functionality should be tested and verified to ensure it works as expected.

# Gender Selection for Nominee

Type: CORRECT\_VALUE

Title: Gender Selection for Nominee  
  
Acceptance Criteria:  
1. The gender selection should be available as a dropdown list.  
2. The dropdown list should contain the following options:  
 - Option 1: Value "5"  
 - Option 2: Value "8"  
 - Option 3: Value "N" (No)  
 - Option 4: Value "SL" (No)  
3. The gender field should be positioned correctly on the form and should be visually aligned with other fields.  
4. The field should be labeled "Gender" and the label should be right-justified.  
5. The field should be case-insensitive and should convert all input to uppercase.  
6. The field should be part of the "NOMINEE\_HEALTH\_DTLS" tab page.  
  
Definition of Done:  
- The gender dropdown list is implemented and contains the specified options.  
- The field is correctly positioned and labeled on the form.  
- The field converts all input to uppercase.  
- The field is included in the "NOMINEE\_HEALTH\_DTLS" tab page.  
- The implementation is tested and verified to meet the acceptance criteria.

# User can select an annuity option from a predefined list

Type: CORRECT\_VALUE

Title: User can select an annuity option from a predefined list  
  
Acceptance Criteria:  
1. The user should be able to see a list of annuity options when interacting with the "Annuity Option" field.  
2. The list should contain the following options:  
 - Option 1: "5"  
 - Option 2: "8"  
 - Option 3: "No" (represented by the value "SL")  
3. The "Annuity Option" field should be located on the "Cover Details" tab.  
4. The field should be visually styled with a white background, black text, and a specific font style (Segoe UI, Demilight, Plain, size 900).  
5. The field should be positioned at coordinates (580, 51) on the tab and have a width of 91 and a height of 15.  
  
Definition of Done:  
- The "Annuity Option" field is implemented and displays the correct list of options.  
- The field is correctly positioned and styled as per the requirements.  
- The field is functional and allows the user to select one of the predefined annuity options.  
- 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 or operations.

# Gender Selection Dropdown

Type: CORRECT\_VALUE

Title: Gender Selection Dropdown  
  
Acceptance Criteria:  
1. The gender selection field should be a dropdown list.  
2. The dropdown list should contain the following options:  
 - Option 1: Value "5"  
 - Option 2: Value "8"  
 - Option 3: Value "N" (No)  
 - Option 4: Value "SL" (No)  
3. The dropdown list should be positioned at the specified location on the form.  
4. The dropdown list should be displayed on the "NOMINEE\_HEALTH\_DTLS" tab page.  
5. The prompt for the dropdown list should be "Gender" and should be right-justified.  
  
Definition of Done:  
1. The gender selection dropdown list is implemented and visible on the form.  
2. The dropdown list contains all specified options.  
3. The dropdown list is correctly positioned and aligned as per the requirements.  
4. The prompt "Gender" is displayed and right-justified.  
5. The functionality is tested and verified to ensure it meets the acceptance criteria.

# Dynamic Update of Payment Methods Based on Frequency and Conditions

Type: CORRECT\_VALUE

Title: Dynamic Update of Payment Methods Based on Frequency and Conditions  
  
Acceptance Criteria:  
1. When the frequency is '4' or '12', or when the frequency is '2' and the FSC code is not one of '2000003024', '7000005464', '7000002372', or '7000000702':  
 - The available payment methods should be:  
 - NACH  
 - Auto Debit Instruction (ADI)  
 - EMANDATE  
 - Credit Card - Standing instruction (CCSI)  
 - SSS  
2. For all other frequencies and FSC codes:  
 - The available payment methods should be:  
 - Cash/Cheque/DD  
 - SSS  
 - Direct Debit (DD)  
 - ECS  
 - Credit Card - Standing instruction (CCSI)  
 - Group Cash (GCSH)  
 - Group Cheque (GCHQ)  
 - Group Direct Credit (GDC)  
 - Bill Junction - ECS (BJECS)  
 - Auto Debit Instruction (ADI)  
 - BG  
 - NACH  
 - EMANDATE  
3. When the frequency is '01':  
 - The renewal payment field should be disabled.  
4. For all other frequencies:  
 - The renewal payment field should be enabled unless the selected payment method is 'CCSI' or 'EMAND', or if a specific flag (pk\_vars.v\_adi\_flag) is not 'Y'.  
  
Definition of Done:  
- The system correctly updates the list of available payment methods based on the specified conditions.  
- The renewal payment field is enabled or disabled based on the frequency and selected payment method.  
- The changes are tested and verified to ensure they meet the acceptance criteria.  
- The user interface reflects the updated payment methods and field states 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.

# Dynamic Payment Method Update Based on Payout Frequency and FSC Code

Type: CORRECT\_VALUE

Detailed description: As a user, I want the system to dynamically update the available payment methods based on the selected payout frequency and other conditions, so that I can choose the appropriate payment method for my needs.  
  
Acceptance criteria:  
1. When the payout frequency is '4' or '12', or if the payout frequency is '2' and the FSC code is not one of '2000003024', '7000005464', '7000002372', or '7000000702':  
 - The available payment methods should be updated to include:  
 - NACH  
 - Auto Debit Instruction (ADI)  
 - EMANDATE (EMAND)  
 - Credit Card - Standing instruction (CCSI)  
 - SSS  
2. For all other payout frequencies and FSC codes:  
 - The available payment methods should be updated to include:  
 - Cash/Cheque/DD (CA)  
 - SSS  
 - Direct Debit (DD)  
 - ECS  
 - Credit Card - Standing instruction (CCSI)  
 - Group Cash (GCSH)  
 - Group Cheque (GCHQ)  
 - Group Direct Credit (GDC)  
 - Bill Junction - ECS (BJECS)  
 - Auto Debit Instruction (ADI)  
 - BG  
 - NACH  
 - EMANDATE (EMAND)  
3. If the payout frequency is '01':  
 - The payment method selection should be disabled.  
4. If the payout frequency is not '01' and the selected payment method is not 'CCSI' or 'EMAND', and the ADI flag is not 'Y':  
 - The payment method selection should be enabled.  
  
Definition of Done:  
- The system dynamically updates the available payment methods based on the selected payout frequency and FSC code.  
- The payment method selection is disabled when the payout frequency is '01'.  
- The payment method selection is enabled under the specified conditions.  
- 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 logic is specific to Oracle Forms and cannot be executed directly in the database without modification.

# Retrieve and Display Rate Details on Button Press

Type: CORRECT\_VALUE

Title: Retrieve and Display Rate Details on Button Press  
  
Acceptance Criteria:  
1. When the button is pressed, the system should fetch the following details from the database for the given application number and top indicator:  
 - Online rate  
 - Staff rate  
 - HPMU rate  
 - Income factor  
 - Auto pay rate  
 - Loyalty factor  
 - WMM rate  
 - AWG total percentage  
2. If the details are successfully fetched, they should be displayed in the corresponding fields.  
3. If there is an error during the fetch operation, the system should handle the error gracefully and set the fields to NULL.  
  
Definition of Done:  
1. The button press should trigger the retrieval of rate details from the database.  
2. The retrieved details should be correctly displayed in the respective fields.  
3. Error handling should be implemented to manage any issues during data retrieval.  
4. 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 online\_rate, staff\_rate, hpmu\_rate, income\_factor, auto\_pay\_rate, loyality\_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 = :correct\_value.appln\_no AND top\_indicator = 'Y';  
```

# Manage Portfolio Strategy Based on Conditions and Product Definitions

Type: CORRECT\_VALUE

Detailed description: As a user, I want the system to manage the portfolio strategy based on specific conditions and product definitions, so that the correct portfolio strategy is applied and appropriate fund details are updated accordingly.  
  
Acceptance criteria:  
1. If the portfolio strategy is 'AT' and the frequency is '12', the system should display an error message indicating that the Auto Transfer Portfolio is not applicable for Monthly Mode.  
2. If the product definition is 'MONEY\_SECURE\_PLAN', the portfolio strategy should be set to 'IS'.  
3. If the product definition is 'PRINCIPAL\_GAIN', the portfolio strategy should be set to 'WL'.  
4. If the portfolio strategy is not null and not 'SO', the system should:  
 - Navigate to the 'AZBJ\_SSO\_FUND\_DETAILS' block.  
 - Clear the block.  
 - Disable the 'CORRECT\_FUNDLIST', 'CORRECT\_APPORT', and 'CORRECT\_FUND\_ID' fields if they are enabled.  
5. If the portfolio strategy is 'SO', the system should:  
 - Enable the 'CORRECT\_FUNDLIST' and 'CORRECT\_APPORT' fields if they are disabled.  
6. If the product code is in (307, 316, 331, 351, 353) and the portfolio strategy is in ('TB', 'AT'), the system should:  
 - Navigate to the 'UL\_AT\_PORTFOL' block.  
 - Clear the block without validation.  
 - Navigate to the 'UL\_AT\_PORTFOL\_2' block.  
 - Clear the block without validation.  
 - Navigate to the 'AZBJ\_FUND\_DETAILS' block.  
 - Enable the 'APPORTIONMENT' field for all records.  
7. If the product code is in (307, 316, 331) and the portfolio strategy is in ('TB', 'WL', 'CP'), the system should disable the 'CMD\_FUND\_VAL' field. Otherwise, it should enable the 'CMD\_FUND\_VAL' field.  
8. If the product code is 311, the portfolio strategy is 'SO', and the package code does not contain 'SINGLE', the system should:  
 - Set the portfolio strategy to null.  
 - Display an error message indicating that the SSO Option is applicable only for Single premium.  
9. The system should call the 'AZBJ\_WOL\_FUND\_POPULATE' procedure to populate fund details based on the portfolio strategy and other conditions.  
  
Definition of Done:  
- The system correctly manages the portfolio strategy based on the specified conditions.  
- Appropriate error messages are displayed when conditions are not met.  
- Fund details are updated or cleared as per the portfolio strategy and product definitions.  
- The 'AZBJ\_WOL\_FUND\_POPULATE' procedure is executed to populate fund details when required.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- The 'AZBJ\_WOL\_FUND\_POPULATE' procedure includes a query to fetch fund details based on the product code, benefit term, and other conditions:  
 ```sql  
 SELECT fund\_short\_name, fund\_full\_name, allocation\_perc  
 FROM azbj\_wheel\_of\_life a, azbj\_tfv\_fund\_definition b  
 WHERE product\_id = :correct\_value.product\_code  
 AND a.fund\_id = b.fund\_id  
 AND yrs\_to\_maturity = :correct\_value.benefit\_term  
 AND cover\_code = :AZBJ\_RIDER\_DETAILS.RIDER\_COVER\_CODE  
 AND allocation\_perc > 0  
 AND NVL (a.portfolio, 'WL') = CASE  
 WHEN :CORRECT\_VALUE.portfolio\_strategy = 'CP' THEN 'CP'  
 ELSE 'WL'  
 END;  
 ```

# Manage Fund Details in Proposal

Type: AZBJ\_SOL\_FUND\_DETAILS

Title: Manage Fund Details in Proposal  
  
Acceptance Criteria:  
1. The system should allow users to view a list of funds associated with a proposal.  
2. Users should be able to add new funds to the proposal by selecting from a list of available funds.  
3. The system should display the name and ID of each fund.  
4. Users should be able to specify the apportionment percentage for each fund.  
5. The system should calculate and display the total fund value based on the apportionment percentages.  
6. Users should have the option to mark a fund for deletion.  
7. The system should ensure that the apportionment percentage is a numeric value and within a valid range.  
8. The system should fetch fund details based on specific criteria, including product code and date range.  
  
Definition of Done:  
1. The user interface displays a list of funds with their respective IDs, names, and apportionment percentages.  
2. Users can add new funds to the list by selecting from a predefined list of available funds.  
3. The total fund value is automatically calculated and displayed based on the apportionment percentages.  
4. Users can mark funds for deletion, and the system reflects these changes.  
5. The system validates the apportionment percentage to ensure it is numeric and within a valid range.  
6. The system fetches and displays fund details based on the specified criteria (product code and date range).  
  
DB queries for Table reference CRUD operations only (With Usage):  
```sql  
SELECT fund\_short\_name AS FUND\_ID, 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 = :correct\_value.product\_CODE  
AND :susac.sa\_daterecd >= NVL(start\_date, '01-apr-2005')  
AND :susac.sa\_daterecd <= NVL(end\_date, '01-jan-3000');  
```  
Usage: This query is used to fetch the list of available funds based on the product code and date range criteria.

# Manage Funding Detail Record Status with Checkbox

Type: AZBJ\_SOL\_FUND\_DETAILS

Detailed description: As a user, I want to manage the status of a funding detail record using a checkbox, so that I can easily mark records for deletion or other actions.  
  
Acceptance criteria:  
1. When the checkbox is checked, the value should be set to 'Y'.  
2. When the checkbox is unchecked, the value should be set to 'N'.  
3. The checkbox should be initialized with the value 'N'.  
4. When navigating to the next item, if the current record is the last one, the focus should move to a specific command button. Otherwise, it should move to the next record's fund name field. If the fund name field is empty, the focus should again move to the command button.  
  
Definition of Done:  
1. The checkbox functionality should be implemented and tested.  
2. The navigation logic should be implemented and tested.  
3. The user interface should reflect the changes and be user-friendly.  
4. All acceptance criteria should be met.  
5. The feature should be 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 SQL queries or table references.

# Validate Apportionment Percentage in Fund Details

Type: AZBJ\_SOL\_FUND\_DETAILS

Title: Validate Apportionment Percentage in Fund Details  
  
Acceptance Criteria:  
1. When the user enters an apportionment percentage, the system should validate that the value is between 0 and 100.  
2. If the entered value is outside this range, the system should display a warning message: "Apportionment percentage should be between 0 and 100" and prevent the user from proceeding.  
3. The system should sum the apportionment percentages of all records in the fund details section.  
4. If the total apportionment percentage equals 100, the system should enable specific fields and navigate to the next appropriate section based on the product code and product definition.  
5. If the total apportionment percentage does not equal 100, the system should navigate to a specific field for correction.  
  
Definition of Done:  
- The apportionment percentage validation is implemented and tested.  
- The warning message is displayed correctly when the entered value is outside the valid range.  
- The system correctly sums the apportionment percentages and enables/disables fields based on the total.  
- Navigation to the next section or correction field works as specified.  
- 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 CRUD operations that can be directly executed in the database.

# Display Fund Details Based on Selected Fund Name

Type: AZBJ\_SOL\_FUND\_DETAILS

Title: Display Fund Details Based on Selected Fund Name  
  
Acceptance Criteria:  
1. When the user navigates to the fund name field, the system should load the field details from the image.  
2. If the proposal type is 'N' and the number of pages is 50, the system should zoom into the image at 50% zoom level.  
3. On double-clicking the fund name field, the system should display a list of values (LOV) for the user to select from.  
4. When the user presses the key to list values, the system should show the LOV for fund selection.  
5. When the user navigates to the next item, the system should clear any visual attributes set on the current item and move to the 'Apportionment' field.  
6. The system should load the field details from the image whenever the user navigates to the fund name field.  
  
Definition of Done:  
1. The fund name field should correctly load and display details from the image.  
2. The system should handle the zoom functionality based on the proposal type and number of pages.  
3. The LOV for fund selection should be displayed correctly on double-click and key press.  
4. Navigation to the next item should work seamlessly, clearing any visual attributes from the current item.  
5. The field details should be loaded from the image whenever the user navigates to the fund name field.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- The system should execute the following query to fetch the fund details for the LOV:  
```sql  
SELECT fund\_short\_name AS FUND\_ID, 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 = :correct\_value.product\_CODE  
AND :susac.sa\_daterecd >= NVL(start\_date, '01-apr-2005')  
AND :susac.sa\_daterecd <= NVL(end\_date, '01-jan-3000');  
```

# Manage Fund Details

Type: AZBJ\_SOL\_SSO\_FUND\_DETAILS

Title: Manage Fund Details  
  
Acceptance Criteria:  
1. The system should display a list of fund details, including Fund ID, Name of Fund, Apportionment percentage, and a checkbox for deletion.  
2. The Fund ID should be displayed in uppercase and should be linked to a list of values (LOV) for selection.  
3. The Name of Fund should be displayed but not editable.  
4. The Apportionment percentage should be editable and restricted to a maximum length of 3 characters.  
5. The system should calculate and display the total fund value based on the apportionment percentages.  
6. The checkbox should allow users to mark a fund for deletion.  
7. The system should fetch fund details from the database based on specific conditions, including product code and date range.  
  
Definition of Done:  
1. The user can view a list of fund details with the specified fields.  
2. The user can select a Fund ID from a predefined list.  
3. The user can edit the Apportionment percentage.  
4. The system calculates and displays the total fund value correctly.  
5. The user can mark a fund for deletion using a checkbox.  
6. The system fetches and displays fund details based on the specified conditions.  
  
DB queries for Table reference CRUD operations only (With Usage):  
```sql  
SELECT fund\_short\_name AS FUND\_ID, 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 = :correct\_value.product\_CODE  
AND :susac.sa\_daterecd >= NVL(start\_date, '01-apr-2005')  
AND :susac.sa\_daterecd <= NVL(end\_date, '01-jan-3000');  
```

# Manage Fund Details with Checkbox for Deletion

Type: AZBJ\_SOL\_SSO\_FUND\_DETAILS

Title: Manage Fund Details with Checkbox for Deletion  
  
Acceptance Criteria:  
1. When the checkbox is checked, the value should be set to 'Y'.  
2. When the checkbox is unchecked, the value should be set to 'N'.  
3. The checkbox should be labeled "Delete" and be visible to the user.  
4. The checkbox should be located at the specified position within the form.  
5. When navigating to the next item, if the current record is the last fund record, the focus should move to the "cmd\_fund" item.  
6. If the current record is not the last fund record, the focus should move to the "FUNDLIST" item and proceed to the next record.  
7. If the "FUNDLIST" item is null, the focus should move to the "cmd\_fund" item.  
  
Definition of Done:  
- The checkbox functionality for marking records for deletion is implemented and tested.  
- The navigation logic for moving between items based on the checkbox state and record position is implemented and tested.  
- The form layout is updated to include the checkbox at the specified position with the correct label and visibility.  
- All acceptance criteria are met and verified through testing.

# Validate Apportionment Percentage in Fund Details

Type: AZBJ\_SOL\_SSO\_FUND\_DETAILS

Detailed description: As a user, I want to ensure that the apportionment percentage entered in the fund details section is validated to be between 0 and 100, so that I can maintain data integrity and avoid incorrect entries.  
  
Acceptance criteria:  
1. When the user enters an apportionment percentage, the system should validate that the value is between 0 and 100.  
2. If the entered value is outside this range, the system should display a warning message indicating that the apportionment percentage should be between 0 and 100.  
3. The system should reset the total apportionment value to 0 if the entered value is invalid.  
4. The system should navigate to the next appropriate field based on the total apportionment value:  
 - If the total apportionment value across all records equals 100, navigate to the spouse detail section.  
 - If the total apportionment value is not 100, navigate to the fund list section.  
  
Definition of Done:  
- The apportionment percentage validation logic is implemented and tested.  
- The warning message is displayed correctly for invalid entries.  
- The total apportionment value is reset to 0 for invalid entries.  
- The navigation logic based on the total apportionment value is implemented and tested.  
- All acceptance criteria are met, and the feature is reviewed and approved by stakeholders.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- Not applicable as the provided XML content does not include any direct database queries for CRUD operations.

# Display Fund Name in Read-Only Text Field

Type: AZBJ\_SOL\_SSO\_FUND\_DETAILS

Title: Display Fund Name in Read-Only Text Field  
  
Acceptance Criteria:  
1. The text field should be read-only and not allow any updates or insertions.  
2. When the text field is focused, it should load the relevant data from an image file based on the current item and block.  
3. Upon pressing the "Next" key, the system should navigate to the "Apportionment" field in the "Fund Details" section.  
4. The system should apply specific visual attributes to the text field based on the current block and item.  
5. The system should determine the appropriate image file to load based on the proposal type and other conditions, and display the image in the designated area.  
  
Definition of Done:  
1. The text field displays the name of the fund correctly.  
2. The text field is read-only and does not allow updates or insertions.  
3. The system loads the correct data from the image file when the text field is focused.  
4. The system navigates to the "Apportionment" field upon pressing the "Next" key.  
5. The correct visual attributes are applied to the text field based on the current block and item.  
6. The appropriate image file is determined and displayed correctly based on the proposal type and other conditions.  
  
DB queries for Table reference CRUD operations only (With Usage):  
- The system should execute a query to retrieve the maximum strip number from the `azbj\_strip\_field\_mapping` table based on the block name, field name, and proposal type, and ensure the strip number exists in the `azbj\_proposal\_stripes` table.  
  
```sql  
SELECT MAX(strip\_no)  
INTO v\_strip\_no  
FROM azbj\_strip\_field\_mapping a  
WHERE block\_name = v\_current\_block  
 AND field\_name = v\_current\_item  
 AND proposal\_type = PK\_VARS.v\_proposal\_type  
 AND EXISTS (  
 SELECT 1  
 FROM azbj\_proposal\_stripes b  
 WHERE a.proposal\_type = b.proposal\_type  
 AND INSTR(stripe\_string, '~' || a.strip\_no || '~') > 0  
 );  
```

# Manage Fund Allocation in Portfolio

Type: UL\_SOL\_AT\_PORTFOL

Title: Manage Fund Allocation in Portfolio  
  
Acceptance Criteria:  
1. The system should allow the user to view and select a fund from a list of available funds.  
2. The system should display the name of the selected fund.  
3. The system should allow the user to input the apportionment percentage for the selected fund.  
4. The system should calculate and display the total fund value based on the apportionment percentages entered.  
5. The system should provide an option to mark a fund for deletion.  
6. The system should ensure that the apportionment percentage is a numeric value and does not exceed three digits.  
  
Definition of Done:  
1. The user can successfully view and select a fund from a list.  
2. The selected fund's name is displayed correctly.  
3. The user can input and save the apportionment percentage for the selected fund.  
4. The total fund value is calculated and displayed accurately based on the entered apportionment percentages.  
5. The user can mark a fund for deletion, and the system reflects this change.  
6. The system validates the apportionment percentage to ensure it is numeric and within the allowed length.  
  
DB queries for Table reference CRUD operations only(With Usage):  
```sql  
SELECT fund\_short\_name AS FUND\_ID, 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 = :correct\_value.product\_CODE  
AND :susac.sa\_daterecd >= NVL(start\_date, '01-apr-2005')  
AND :susac.sa\_daterecd <= NVL(end\_date, '01-jan-3000');  
```  
  
This query is used to fetch the list of available funds based on certain conditions, such as product code and date range.

# Validate Apportionment Percentage and Navigate Based on Total

Type: UL\_SOL\_AT\_PORTFOL

Title: Validate Apportionment Percentage and Navigate Based on Total  
  
Acceptance Criteria:  
1. When the user enters an apportionment percentage, the system should validate that the value is between 0 and 100.  
2. If the entered percentage is outside this range, the system should display a warning message: "Apportionment percentage should be between 0 and 100" and prevent further processing.  
3. The system should sum up all apportionment percentages and ensure that the total equals 100.  
4. If the total apportionment percentage equals 100 and the product code is 71 with a specific product definition, the system should enable certain fields and navigate to the appropriate section for further data entry.  
5. If the total apportionment percentage does not equal 100, the system should navigate to a specific field for correction.  
  
Definition of Done:  
- The validation logic for apportionment percentage is implemented and tested.  
- The system displays the correct warning message when the percentage is out of range.  
- The system correctly sums up the apportionment percentages and validates the total.  
- The system enables the appropriate fields and navigates to the correct sections based on the total apportionment percentage and product details.  
- 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 direct database CRUD operations that can be executed independently of Oracle Forms constructs.

# Display and Select Fund Name

Type: UL\_SOL\_AT\_PORTFOL

Detailed description: As a user, I want the system to display the appropriate fund name in the "Fund Name" field based on the selected fund from a list of values (LOV) and ensure that the field is read-only to prevent unauthorized modifications.  
  
Acceptance criteria:  
1. When the "Fund Name" field is double-clicked, the system should display a list of available funds (LOV) for selection.  
2. The "Fund Name" field should be populated with the selected fund's name from the LOV.  
3. The field should be read-only, meaning users cannot manually enter or modify the fund name.  
4. The system should load the field's value from an image if the proposal type is 'N' and the number of pages is 50.  
5. The system should handle navigation to the next item when the user presses the key to move to the next field.  
  
Definition of Done:  
- The "Fund Name" field displays the correct fund name based on the user's selection from the LOV.  
- The field is read-only and cannot be modified manually by the user.  
- The system correctly loads the field's value from an image under specified conditions.  
- Navigation to the next item works seamlessly without errors.  
  
DB queries for Table reference CRUD operations only (With Usage):  
```sql  
SELECT fund\_short\_name AS FUND\_ID, 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 = :correct\_value.product\_CODE  
 AND :susac.sa\_daterecd >= NVL(start\_date, '01-apr-2005')  
 AND :susac.sa\_daterecd <= NVL(end\_date, '01-jan-3000');  
```  
- This query is used to fetch the list of available funds for the LOV based on the product code and date received.

# Manage PAN Details

Type: AZBJ\_PAN\_DET

Title: Manage PAN Details  
  
Acceptance Criteria:  
1. The system should allow the user to input and display the following fields:  
 - PAN Number  
 - PAN Status  
 - Name Match  
 - DOB Match  
2. Each field should have a maximum length as follows:  
 - 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 and justified as specified:  
 - PAN Number, PAN Status, Name Match, and DOB Match should be center-aligned.  
4. The system should provide a scrollbar for navigating through the records if there are more than three records.  
5. The system should ensure that the fields are editable and can be updated by the user.  
  
Definition of Done:  
- The user can successfully input and view PAN details, including PAN number, PAN status, name match, and DOB match.  
- The fields adhere to the specified maximum lengths and alignment requirements.  
- The user can navigate through multiple records using a scrollbar.  
- The fields are editable and can be updated by the user.  
- 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 table references.