SECP2613: SYSTEM ANALYSIS AND DESIGN

System Documentation (SD)

Software Testing Documentation (STD)

SmartComply

28th July 2024

Faculty of Computing

Prepared by:

UnixxCode

NURUL ERINA BINTI ZAINUDDIN A22EC0254

NUR ALEYSHA QURRATU'AINI BINTI MAT SALLEH A22EC0241

NIK ZULAIKHA BINTI ZURAIDI AFANDI A22EC0232

NUR ARINI FATIHAH BINTI MOHD SABIR A22EC0244

NUR FARAH ADIBAH BINTI IDRIS A22EC0245

Table of Contents

1.0 Dynamic Form Builder & Audit Forms Module

Section A: Requirements-based Testing

A1 Functional Requirements

A1.1 Test Requirements (TR)

Table 1. List of Functional Test Requirements

Use Case (UC)	TR ID	Test Requirements
UC <UC004> <Create Form>	TR_{004-01}	Verify that the system allows Admin to access the Create Form page
	TR_{004-02}	Verify that the system displays only active compliance categories
	TR_{004-03}	Verify that the Admin can enter the form title and description
	TR_{004-04}	Verify that the system saves the form with "Draft" status upon submission
	TR_{004-05}	Verify that the system redirects the Admin to the Form Builder page after form creation
UC <UC005> <Build Form>	TR_{005-01}	Verify that the Admin can add a new section with title and description
	TR_{005-02}	Verify that the system saves the new section and displays it dynamically in the UI
	TR_{005-03}	Verify that the Admin can add a question/item to a specific section
	TR_{005-04}	Verify that the system saves the new question and updates the section in real time

	TR_{005-05}	Verify that the Admin can delete a question from a section using the bin icon
	TR_{005-06}	Verify that the Admin can delete an entire section
	TR_{005-07}	Verify that the form structure is correctly maintained after multiple edits
	TR_{005-08}	Verify that the form is ready for preview after building is complete
UC <UC006> <Manage Status>	TR_{006-01}	Verify that the Admin can click the “Archived” button for a form
	TR_{006-02}	Verify that the system shows a confirmation alert before archiving
	TR_{006-03}	Verify that the system updates the form status to “Archived” after confirmation
	TR_{006-04}	Verify that the Admin can click the “Restore” button for an archived form
	TR_{006-04}	Verify that the system updates the form status back to “Draft” after restoring
	TR_{006-05}	Verify that the Admin can click the “Submit” button for a draft form
	TR_{006-06}	Verify that the system updates the form status to “Published” upon submission
UC <UC007> <Manage Status>	TR_{007-01}	Verify that the Admin can click the “Preview” button after building the form

	TR_{007-02}	Verify that the system loads the form in a read-only preview mode
	TR_{007-03}	Verify that all form sections, questions, and scoring settings are accurately displayed
	TR_{007-04}	Verify that the Admin can review logic such as required fields and looping configurations
	TR_{007-05}	Verify that the Admin can click the “Submit Audit Form” button from the preview page
	TR_{007-06}	Verify that the system updates the form status to “Published” after successful submission
UC <UC008> <Configure Question>	TR_{008-01}	Verify that the Admin can select a question item from a section
	TR_{008-02}	Verify that the Admin can click the pencil (edit) button to open the question configuration view
	TR_{008-03}	Verify that the Admin can update the question title
	TR_{008-04}	Verify that the Admin can toggle the question as “Required”
	TR_{008-05}	Verify that the Admin can enable looping for a question
	TR_{008-06}	Verify that the Admin can input the number of loops
	TR_{008-07}	Verify that the Admin can set the maximum score for the question
	TR_{008-08}	Verify that the system saves and reflects the updated configuration in the UI

A1.2 Test Cases

Table 2. List of Functional Test Cases

TR ID	Case No.	Data Entered	Expected Result
TC004	**TC004-01_01**	Admin clicks “Create Form” button on Dashboard	Form” button on DashboardCreate Form page is loaded
TC004	**TC004-03_01**	Admin enters "Safety Audit" in title and "Site Q3 audit" in description	Title and description fields are populated
TC004	**TC004-04_01**	Admin clicks “Submit” on Create Form	Form saved with status = Draft
TC005	**TC005-01_01**	Admin clicks “Add Section” and fills in section title and description	Section is added and shown on the form
TC005	**TC005-04_01**	Admin clicks section and choose items from Form Items	Item is added and shown on the form
TC005	**TC005-05_01**	Admin clicks bin icon on a question	Question removed from form
TC005	**TC005-06_01**	Admin clicks “Delete Section	Section deleted from form
TC006	**TC006-01_02**	Admin clicks “Archived” button	Alert popup is shown
TC006	**TC006-01_02**	Admin confirms archive action	Form status = Archived
TC006	**TC006-02_01**	Admin clicks “Restore” on archived form	Form status = Draft
TC006	**TC006-03_01**	Admin clicks “Submit” button	Form status = Published

TC007	**TC007-01_01**	Admin clicks “Preview”	Preview mode shows full form

A2 Non-Functional Requirements

A2.1 Test Requirements (TR)

Table 3. List of Non-Functional Test Requirements

Non-functional	TR ID	Test Requirements
Performance	TRNF001	Verify that the form loads within 3 seconds
Usability	TRNF002	Verify that all buttons have clear labels and tooltips
Compatibility	TRNF003	Verify that the system works on Chrome, Edge, and Firefox
Security	TRNF004	Verify that only Admin can access form management pages
Availability	TRNF005	Verify that the system is accessible 24/7 during testing

A2.2 Test Cases

Table 4. List of Non-Functional Test Cases

TR ID	Case No.	Data Entered	Expected Result
TRNF001	**TCTRNF001_01**	Admin opens form builder page	Page loads in less than 3 seconds
TRNF002	**TCTRNF001_02**	Hover over buttons (Add, Submit, Delete,Edit)	Tooltip appears with appropriate description
TRNF003	**TCTRNF001_03**	Open system in Chrome, Edge, Firefox	UI renders consistently across browsers

TRNF004	**TCTRNF001_04**	Try to access admin page as User	Access denied; redirected to error page/login
TRNF005	**TCTRNF001_05**	Access system at random times (day/night)	System is accessible; login page loads

A3 Summary

This testing module focuses on validating the core functionalities of the Form Management system used by Admins to create, build, configure, and publish forms. It ensures that features like adding sections and questions, configuring question properties, and managing form statuses (Draft, Archived, Published) work correctly. Test requirements and test cases are designed to cover each use case in detail, ensuring the form behaves as expected from creation to final submission.

Additionally, basic non-functional requirements like usability and responsiveness are considered to ensure Admins can perform actions smoothly without errors or delays. The goal of this module is to confirm that all form-related operations are stable, user-friendly, and reliable within the defined scope.

Section B: Black-box Testing

B1 Object Class

B1.1 Equivalence Partitioning and Boundary Value Analysis

Table 5. Equivalence Partition and Input Range

Object class	Attributes	Equivalence Partition and Input Range
JenisForm	***FormTypeId*** *(int)*	**Valid:** Any positive integer (e.g., 1, 100). **Invalid:** null, 0, negative integers, non-integer values.
	Name (string)	**Valid Partitions:** - String with 1 character. - String between 2 and 254 characters. - String with 255 characters. **Boundary Values:** 1, 255 characters. **Invalid Partitions:** - null or empty string. - String with > 255 characters.
	Description (string?)	**Valid Partitions:** - null or empty string. - String with 1 character. - String between 2 and 999 characters. - String with 1000 characters. **Boundary Values:** 0, 1, 1000 characters. **Invalid Partition:** - String with > 1000 characters.

	CreatedAt (DateTime)	**Valid:** Any valid DateTime value (e.g., 2025-07-27 22:30:00). **Invalid:** null, non-date values.
	ComplianceCategoryId (int?)	**Valid:** null, any positive integer. **Invalid:** 0, negative integers, non-integer values.
	Status (FormStatus)	**Valid:** 1 (Draft), 2 (Published), 3 (Archived), 4 (Revised). **Invalid:** Integers other than 1, 2, 3, 4; null, non-integer values.
	TenantId (string)	**Valid Partitions:** - String with 1 character. - String between 2 and 35 characters. - String with 36 characters (e.g., a GUID). **Boundary Values:** 1, 36 characters. **Invalid Partitions:** - null or empty string. - String with > 36 characters.

Object class	**Attributes**	**Equivalence Partition and Input Range**
FormSection	***SectionId** (int)*	**Valid:** Any positive integer. **Invalid:** null, 0, negative integers, non-integer values.
	FormTypeId (int)	**Valid:** Any positive integer that exists in JenisForm. **Invalid:** null, 0, negative integers, non-integer values, or a FormTypeId that does not exist.
	T	**Valid Partitions:**

	Title (string)	- String with 1 character. - String between 2 and 254 characters. - String with 255 characters. **Boundary Values:** 1, 255 characters. **Invalid Partitions:** - null or empty string. - String with > 255 characters.
	Description (string)	**Valid Partitions:** - null or empty string. - String with 1 character. - String between 2 and 999 characters. - String with 1000 characters. **Boundary Values:** 0, 1, 1000 characters. **Invalid Partition:** - String with > 1000 characters.
	Order (int)	**Valid:** Any integer (positive, zero, or negative). **Invalid:** null, non-integer values.

Object class	**Attributes**	**Equivalence Partition and Input Range**
FormItem	***ItemId*** *(int)*	**Valid:** Any positive integer. **Invalid:** null, 0, negative integers, non-integer values.

	SectionId (int)	**Valid:** Any positive integer that exists in FormSection. **Invalid:** null, 0, negative integers, non-integer values, or a SectionId that does not exist.
	Question (string)	**Valid Partitions:** - String with 1 character. - String between 2 and 499 characters. - String with 500 characters. **Boundary Values:** 1, 500 characters. **Invalid Partitions:** - null or empty string. - String with > 500 characters.
	ItemType (ItemType)	**Valid:** 1 (Text) through 8 (Signature). **Invalid:** Integers other than 1-8; null, non-integer values.
	OptionsJson (string?)	**Valid:** null, empty string, or any valid JSON formatted string. **Invalid:** Malformed JSON string.
	IsRequired (bool)	**IsRequired (bool)**
	MaxScore (int?)	Valid: null, any positive integer. Invalid: 0, negative integers, non-integer values.

B1.2 Test Cases

Table 6. Object Class Based Test Cases

Object name: JenisForm

Method name: CreateJenisForm

Case No.	Equivalence Class	Representative (BVA)	Expected Result
TC007-01	Valid Name (Min Length Boundary)	Name: "A"	Record is created successfully.
TC007-02	Valid Name (Max Length Boundary)	Name: A string with 255 characters.	Record is created successfully.
TC007-03	Invalid Name (Required Field)	Name: null	Validation error: "The Name field is required.
TC007-04	Invalid Name (Exceeds Max Length)	Name: A string with 256 characters.	Validation error: "The field Name must be a string with a maximum length of 255.".
TC007-05	Valid Description (Max Length Boundary)	Description: A string with 1000 characters	Record is created successfully.
TC007-06	Invalid Description (Exceeds Max Length)	Description: A string with 1001 characters.	Validation error: "The field Description must be a string

			with a maximum length of 1000.".

Object name: FormSection

Method name: Add FormSection

Case No.	Equivalence Class	Representative (BVA)	Expected Result
TC004-01	Valid Title (Min Length Boundary)	Title: "A"	Record is created successfully.
TC004-02	Valid Title (Max Length Boundary)	Title: A string with 255 characters.	Record is created successfully.
TC004-03	Invalid Title (Required Field)	Title: null	Validation error: "The Title field is required.".
TC004-04	Invalid Title (Exceeds Max Length)	Title: A string with 256 characters.	Validation error: "The field Title must be a string with a maximum length of 255.".
TC004-05	Valid FormTypeId	FormTypeId: An integer corresponding to an existing JenisForm record.	Record is created successfully.

Object name: FormItem

Method name: Add FormItem

Case No.	Equivalence Class	Representative (BVA)	Expected Result
TC005-01	Valid Question (Min Length Boundary)	Question: "?"	Record is created successfully.
TC005-02	Valid Question (Max Length Boundary)	Question: A string with 500 characters.	Record is created successfully.
TC005-03	Invalid Title (Required Field)	Question: null	Validation error: "The Question field is required.
TC005-04	Invalid Question (Exceeds Max Length)	Question: A string with 501 characters.	Validation error: "The field Question must be a string with a maximum length of 500."‥
TC005-05	Valid ItemType	ItemType: ItemType.Dropdown (5)	Record is created successfully.

B2 Summary

This section details black-box testing for several object classes: JenisForm, FormSection, and FormItem. It begins by outlining the Equivalence Partitioning and Boundary Value Analysis for each attribute within these classes, defining valid and invalid input ranges and their corresponding boundary values. For example, JenisForm.Name is valid for strings between 1 and 255 characters, with 1 and 255 as boundary values, while FormItem.Question is valid for strings between 1 and 500 characters, with 1 and 500 as boundaries.

Following this, test cases are derived for each object class based on the defined equivalence partitions and boundary values. The test cases cover scenarios such as valid minimum and maximum length boundaries, required field validation, and exceeding maximum length for string attributes. They also include tests for valid and invalid integer and enum values, ensuring comprehensive coverage of the defined input ranges for methods.

2.0 Compliance Framework Setup Module

Section A: Requirements-based Testing

A1 Functional Requirements

A1.1 Test Requirements (TR)

Table 1. List of Functional Test Requirements

Use Case (UC)	TR ID	Test Requirements
UC <UC009> <Create Compliance Category>	$TR_{009\text{-}01}$	Verify that the system allows Admin to access the Create Compliance Category page.
	$TR_{009\text{-}02}$	Verify that the Admin can input valid Name, Code, and Description for the category.
	$TR_{009\text{-}03}$	Verify that the system saves the Compliance Category with 'Active' status.
	$TR_{009\text{-}04}$	Verify that validation errors are displayed when required fields are left empty.
	$TR_{009\text{-}05}$	Verify that the system prevents duplicate Compliance Category codes
UC <UC010> <Edit Compliance Category>	$TR_{010\text{-}01}$	Verify that the Admin can access the Edit page of an existing Compliance Category.
	$TR_{010\text{-}02}$	Verify that the Admin can update Name, Code, and Description fields.
	$TR_{010\text{-}03}$	Verify that the LastModifiedDate and LastModifiedBy fields are updated correctly.
	$TR_{010\text{-}04}$	Verify that invalid data (e.g., empty Name or invalid Code) is not accepted.

UC <UC011> <Manage Compliance Status>	$TR_{011\text{-}01}$	Verify that the Admin can archive a category if no active dependencies exist.
	$TR_{011\text{-}02}$	Verify that the system prevents archiving if associated Forms or Folders are active.
	$TR_{011\text{-}03}$	Verify that the Admin can unarchive an archived Compliance Category.
	$TR_{011\text{-}04}$	Verify that the category's Status is updated accordingly in the system.

A1.2 Test Cases

Table 2. List of Functional Test Cases

TR ID	Case No.	Data Entered	Expected Result
TR009_001	**TCTR009_001_01**	Admin clicks "Create Compliance Category"	Create Category page is successfully loaded
TR009_002	**TCTR009_002_01**	Name: "Finance", Code: "FIN001", Description: "Risk"	Category is saved and visible in list with 'Active' status
TR009_004	**TCTR009_004_01**	Name: (blank), Code: "HR001"	Validation message shown: "Name is required."
TR009_005	**TCTR009_005_01**	Code: "FIN001" (already exists in same tenant)	Validation error: "Code already exists."
TR010_001	**TCTR010_001_01**	Click Edit on category "Finance"	Edit Category page is successfully loaded

TR010_002	**TCTR010_002_01**	Change Name to "Financial Compliance", update Description	Fields updated; system displays success message
TR010_003	**TCTR010_003_01**	After editing, check LastModified fields	LastModifiedDate and LastModifiedBy are updated
TR010_004	**TCTR010_004_01**	Leave Code field blank	Validation error: “Code is required.”
TR011_001	**TCTR011_001_01**	Archive “Finance” (no active folders/forms)	Status updated to “Archived”; success message displayed
TR011_002	**TCTR011_002_01**	Attempt archive with linked active folder	System displays error: “Category has active dependencies.”
TR011_003	**TCTR011_003_01**	Unarchive category “Finance”	Status updated to “Active” and visible in active list
TR011_004	**TCTR011_004_01**	Check status field after archiving and unarchiving actions	Correct status reflected in database and UI

A2 Non-Functional Requirements

A2.1 Test Requirements (TR)

Table 3. List of Non-Functional Test Requirements

Non-functional	TR ID	Test Requirements
Performance	TRNF009_01	Verify that the Create and Edit Category pages load within 2 seconds.
Usability	TRNF009_02	Verify that all input fields have appropriate labels, error messages, and tooltips.
Security	TRNF009_03	Verify that only Admins are authorized to access category creation and management.

A2.2 Test Cases

Table 4. List of Non-Functional Test Cases

TR ID	Case No.	Data Entered	Expected Result
TRNF009_01	**TCTRNF009_01_01**	Admin opens Create Category page	Page loads in less than 2 seconds
TRNF009_02	**TCTRNF009_02_01**	Hover over input fields	Tooltips and inline validation hints appear
TRNF009_03	**TCTRNF009_03_01**	Try to access category page as User	Access denied; redirected to login or error page

A3 Summary

*The **requirements-based testing strategy** is most appropriate to be executed at the **system testing** level. This is because the Compliance Framework Setup features rely on multiple integrated components such as user authentication, form linkage, and data validation. By validating the defined functional and non-functional requirements, this testing ensures the system behaves correctly from an end-user perspective. This approach is critical for modules like Compliance Category management, where correctness, access control, and data integrity are essential for enabling other modules (Forms, Audit, Dashboard) to function accurately.*

Section B: Black-box Testing

B1 Object Class

B1.1 Equivalence Partitioning and Boundary Value Analysis

Table 5. Equivalence Partition and Input Range

Object class	Attributes	Equivalence Partition and Input Range
ComplianceCategory	*Name*	Valid: String with 1–100 characters Invalid: null, empty, > 100 characters
	Code	Valid: Unique string with 1–20 characters Invalid: null, empty, > 20 chars, duplicated code
	Description	Valid: null, empty string, 1–255 characters Invalid: > 255 characters
	Status	Valid: "Active", "Archived" Invalid: null, other non-enum strings
	TenantId	Valid: 1–36 characters (GUID or string) Invalid: null, empty string, > 36 characters
	CreatedDate	Valid: Any valid DateTime Invalid: null, non-date values
	LastModified Date	Valid: Any valid DateTime Invalid: null, non-date values

B1.2 Test Cases

Table 6. Object Class Based Test Cases

Object name: ComplianceCategory

Method name: CreateComplianceCategory

Case No.	Equivalence Class	Representative (BVA)	Expected Result
TC*009-01*	Valid Name and Code	Name = "Safety", Code = "SAF01"	Category created successfully
TC*009-02*	Invalid Name (empty)	Name = "", Code = "SAF02"	Validation error: "Name is required"
TC*009-03*	Invalid Code (duplicate)	Code = "SAF01" (already exists in tenant)	Validation error: "Code already exists"
TC*009-04*	Valid Description (max)	Description = 255 characters	Category created successfully
TC*009-05*	Invalid Description	Description = 256 characters	Validation error: "Description too long"

Object name: ComplianceCategory

Method name: UpdateComplianceCategory

Case No.	Equivalence Class	Representative (BVA)	Expected Result
TC*010-01*	Valid update	New Name = "Financial Risk", New Code = "FIN02"	Category updated successfully
TC*010-02*	Invalid update (empty Code)	Code = ""	Validation error: "Code is required"
TC*010-03*	Invalid Code (too long)	Code = 21 characters	Validation error: "Code exceeds max"
TC*010-04*	Valid Name (max boundary)	Name = 100 characters	Update successful

Object name: ComplianceCategory

Method name: ArchiveComplianceCategory

Case No.	**Equivalence Class**	**Representative (BVA)**	**Expected Result**
TC*011-01*	Valid archive (no active dependencies)	Category not linked to forms/folders	Status updated to "Archived"
TC*011-02*	Invalid archive (active dependencies)	Category linked to active JenisForm	Error: "Active dependencies exist"
TC*011-03*	Invalid ID	ID = -1 or non-existent ID	Error: "Category not found"

Object name: ComplianceCategory

Method name: UnarchiveComplianceCategory

Case No.	**Equivalence Class**	**Representative (BVA)**	**Expected Result**
TC*012-01*	Valid unarchive	Archived Category ID = 5	Status updated to "Active"
TC*012-02*	Invalid unarchive	Category already Active	Error: "Category is already active"

B2 Summary

The black-box testing strategy is best suited for system testing and acceptance testing levels in the context of the Compliance Framework Setup module. This is because the main focus of the module is on input validation, CRUD operations, and business rule enforcement, especially from the Admin's perspective. By treating the module as a black box, testers can verify the correctness of behavior (such as status updates, field validation, and uniqueness constraints) without needing to know the internal implementation. This approach also reflects how actual users will interact with the module, making it ideal for ensuring reliability, security, and usability before deployment.

3.0 Audit Module

Section A: Requirements-based Testing

A1 Functional Requirements

A1.1 Test Requirements (TR)

Table 1. List of Functional Test Requirements

Use Case (UC)	TR ID	Test Requirements
UC012 Fill Audit Form	TR_{012_001}	Verify that the system displays a list of available form templates to the User.
	TR_{012_002}	Verify that selecting a form template correctly loads an empty audit form.
	TR_{012_003}	Verify that all form items (questions, input fields, dropdowns, etc.) are rendered correctly.
	TR_{012_004}	Verify that the score is calculated correctly.
UC013 Manage Audit	TR_{013_001}	Verify that clicking 'Edit' on an audit record loads the form with pre-filled existing data.
	TR_{013_002}	Verify that the audit's status and scores are re-calculated and updated after editing and resubmitting.
	TR_{013_003}	Verify that clicking 'Archive' sets the audit record's status to 'Archived' in the database.
UC014 Add Corrective Action	TR_{014_001}	Verify that clicking 'Add Corrective Action' displays a form listing only items that did not receive a full score.
	TR_{014_002}	Verify that the User can input notes for each corrective action.
UC015 Add Follow Up Audit	TR_{015_001}	Verify that the follow-up form highlights or focuses on the items that had corrective actions.

	TR_{015_002}	Verify that new scores are calculated based on the updated responses.
UC016 View Audit Records	TR_{016_001}	Verify that clicking 'Details' on an audit record displays a comprehensive report.
	TR_{016_002}	Verify that a Manager can access the 'Details' view for any audit record.

A1.2 Test Cases

Table 2. List of Functional Test Cases

TR ID	Case No.	Data Entered	Expected Result
TR*012_001*	**TC_{TR012_001}_01**	1. Log in as a user with access to multiple form templates. 2. Navigate to the "Start New Audit" section (e.g., via audit page).	The system displays a list of forms created by admin and form name should be clickable or selectable.
TR*012_002*	**TC_{TR012_002}_01**	1. Log in as a user. 2. Navigate to the list of form templates. 3. Click ‘Start Audit’ from any form template.	An empty audit form with title is loaded. All form items are displayed without pre-filled answers. Verification fields are empty.
TR*012_003*	**TC_{TR012_003}_01**	1. Log in as a user. 2. Navigate to the list of form templates. 3. Click ‘Start Audit’ from any form template.	Each form item is rendered with its correct question, and the appropriate input control is displayed (e.g., text box, radio buttons, dropdown, file upload button for File, signature pad for Signature).

TR*012_004*	**TC_{TR012_004}_01**	1. Log in as a user. 2. Load an empty audit form. 3. Fill out all questions with responses that yield the maximum possible score for each item. 4. Fill required verification fields. 5. Click 'Submit Audit'.	The submitted Audit Instance's TotalScore is 30, and PercentageScore is 100%. Audit status is 'Completed'.
TR*013_001*	**TC_{TR013_001}_01**	1. Log in as a user. 2. Complete and submit an audit form with various filled items and verification details (Audit ID: AI001). 3. Click 'Edit' for audit record AI001.	The audit form for AI001 is loaded. All previously saved responses are pre-filled in their respective input fields. Verification fields (Auditor, Outlet, Verifier) are pre-filled and the saved signature.
TR*013_002*	**TC_{TR013_002}_01**	1. Log in as a user. 2. Complete an audit. 3. Click 'Edit' for. 4. Change response for Item X. 5. Click 'Submit Audit' (as 'Completed').	The audit TotalScore is updated.
TR*013_003*	**TC_{TR013_003}_01**	1. Log in as a user. 2. Navigate to the Audit Listing page. 3. Locate audit record (e.g., AI004). 4. Click the 'Archive' action button/link for AI004.	The audit record AI004 is no longer visible in the default Audit Listing.

		5. Verify a confirmation dialog and confirm.	
TR*014_001*	**TC_{TR014_001}_01**	1. Log in as a user. 2. Complete an audit with Item A (Score 10/10), Item B (Score 5/10), and Item C (Score 0/5). 3. Navigate to the audit details page for this audit (e.g., AI006). 4. Click 'Add Corrective Action'.	The "Add Corrective Action" form displays only Item B and Item C (the deficient items), allowing input for their corrective action notes and due dates.
TR*014_002*	**TC_{TR014_002}_01**	1. Log in as a user. 2. Navigate to the "Add Corrective Action" form for an audit with deficient items (e.g., AI006). 3. For Item B (5/10), enter "Replace faulty part." 4. For Item C (0/5), enter "Conduct re-training session." 5. Select due dates for both. 6. Click 'Submit Corrective Action'.	The corrective actions are successfully saved. On the Audit Details page for AI006, the CorrectiveAction records for Item B and Item C display the entered notes: "Replace faulty part" and "Conduct re-training session" respectively.
TR*015_001*	**TC_{TR015_001}_01**	1. Log in as a user. 2. Have an audit record (AI008) with Status: NeedsCorrectiveAction, CA exists).. 3. Navigate to the Audit Listing.	The follow-up audit form loads AI008's data. Item X is prominently displayed and editable.

		4. Click 'Follow Up Audit' for AI008.	
TR*015_002*	**TC_{TR015_002}_01**	1. Log in as a user. 2. Initiate a follow-up audit for AI008 3. Change the response for Item X. 4. Complete verification. 5. Click 'Submit Audit'.	The TotalScore and PercentageScore for AI008 updated.
TR*016_001*	**TC_{TR016_001}_01**	1. Log in as a user. 2. Navigate to the Audit Listing page. 3. Click 'Details' for a completed audit record (e.g., AI009) that has various item types, scores, and verification details.	The Audit Details page for AI009 is displayed, showing: - General audit information (date, auditor, scores, current status). - A clear breakdown of all form questions and the corresponding submitted answers/values. - Individual scores for each scored item. - The "Checked by", "Acknowledged by", and "Verified by" sections showing the names, designations, dates, and images of signatures (if present). - A list of all corrective actions linked to this audit instance, including their notes, due dates, and current status (Pending, Completed, etc.).
TR*016_002*	**TC_{TR016_002}_01**	1. Log in as a Manager account.	The Manager is able to successfully access and view

		2. Navigate to the Audit Listing. 4. Click 'Details' for AI012.	the comprehensive details report for AI012.

A2 Non-Functional Requirements

A2.1 Test Requirements (TR)

Table 3. List of Non-Functional Test Requirements

Non-functional	TR ID	Test Requirements
Performance	TR*001*	The system shall display the Audit Listing page for a tenant with up to 1,000 audit records within 3 seconds.
	TR002	The system shall load an empty or pre-filled Audit Form (UC012) within 2 seconds.
Security	TR003	The system shall authenticate all users (Users and Managers) prior to granting access to any audit functionality.
Usability	TR004	The user interface for filling the audit form (UC012) shall be intuitive and easy to navigate for users with basic computer proficiency.

A2.2 Test Cases

Table 4. List of Non-Functional Test Cases

TR ID	Case No.	Data Entered	Expected Result
TR*002*	**TC_{TR002}_01**	1. Log in as a user. 2. Navigate to "Start New Audit" section.	The empty audit form loads and is fully displayed within 2 seconds.

		3. Click to select a form template.	
TR003	**TC_{TR003}_02**	1. Navigate to the application login page. 2. Enter valid standard user credentials. 3. Click 'Login'.	The user is successfully logged in and redirected to their default dashboard or home page, gaining access to audit functionalities.

Section B: Black-box Testing

B1 Object Class

B1.1 Equivalence Partitioning and Boundary Value Analysis

Table 5. Equivalence Partition and Input Range

Object class	Attributes	Equivalence Partition and Input Range
AuditInstance	TotalScore	**Valid:** - Valid positive integer below totalmaxscore **Invalid:** - Negative score - Above totalmaxscore
	PercentageScore	**Valid:** - Valid integer between 0 until 100 **Invalid:** - Negative value - Exceeds 100%
	Status	**Valid:** - “Draft”, “Completed”, “NeedsCorrectiveAction”, “NeedsFollowUp” **Invalid:** - Other status than stated above

B1.2 Test Cases

Table 6. Object Class Based Test Cases

Object name: AuditController

Method name: SubmitAudit

Case No.	Equivalence Class	Representative (BVA)	Expected Result
TC*001*	A dictionary containing audit data, including AuditID, AuditorName, Location, and a list of AuditItems.	Each AuditItem should have at least a QuestionID, ResponseValue (e.g., 'Yes', 'No', a numerical score like 5), and MaxScore (e.g., 10). For some items, CorrectiveActionNeeded might be True with CorrectiveActionNotes and CorrectiveActionDueDate.	The system successfully processes the audit submission.
TC002	Submit with invalid data format in a field.	A dictionary where a field has a value in an incorrect format (e.g., Location is an integer when it expects a string, or AuditItem.ResponseValue for a numeric question is "abc").	The system displays validation errors.
TC003	Submit with missing required form fields.	A dictionary with one or more required fields (e.g., AuditorName, Location, or a required	The system displays validation errors on the form indicating the

		AuditItem.ResponseValue) left empty.	incorrect data format for the respective fields.

4.0 Dashboard and Reporting Module

Section A: Requirements-based Testing

A1 Functional Requirements

A1.1 Test Requirements (TR)

Table 1. List of Functional Test Requirements

Use Case (UC)	TR ID	Test Requirements
UC017 View Audit Summary	TR_{017_001}	Verify that a user can successfully access and view their default dashboard.
	TR_{017_002}	Verify that dashboard widgets display correct and up-to-date data relevant to the user's tenant.
	TR_{017_003}	Verify that the dashboard refreshes data automatically or manually as configured.
	TR_{017_004}	Verify that a user can view dashboard data only for their assigned tenant.
UC018 Generate Report	TR_{018_001}	Verify that the system displays a list of completed audit instances relevant to the user's tenant.
	TR_{018_002}	Verify that the generated report file is available for download after successful generation.

A1.2 Test Cases

Table 2. List of Functional Test Cases

TR ID	Case No.	Data Entered	Expected Result
TR*017_001*	**TC_{TR017_001}_01**	Login as User. Navigate to Dashboard.	Dashboard page loads successfully, displaying default widgets.

TR*017_002*	**TC_{TR017_002}_01**	Login as User. View Audit Summary.	Audit Summary displays accurate data.
TR*017_003*	**TC_{TR017_003}_01**	Login as User. Remain on Dashboard page for configured auto-refresh interval	Dashboard widgets update with latest data without full page reload.
TR*017_004*	**TC_{TR017_004}_01**	Login as User from Tenant A. View Audit Summary.	Audit summary displays linked to Tenant A data only.
TR*018_001*	**TC_{TR018_001}_01**	Login as Tenant A User. Click "Export Audit to CSV" button. Download and open the generated CSV file.	The CSV file contains all 5 audit records specific to Tenant A, with correct data.
TR*018_002*	**TC_{TR018_002}_01**	Login as Tenant A User. Click "Export Audit to CSV" button.	The browser initiates a download of a CSV file

A2 Non-Functional Requirements

A2.1 Test Requirements (TR)

Table 3. List of Non-Functional Test Requirements

Non-functional	TR ID	Test Requirements
Performance	TR001	Dashboards should load within 5 seconds for a user.
	TR002	The "Export Audit to CSV" operation should complete within 10 seconds.
	TR003	The downloaded CSV file should be available within 2 seconds after report generation completes.
Security	TR004	Users can only view and generate audit reports for data belonging to their assigned tenant.

A2.2 Test Cases

Table 4. List of Non-Functional Test Cases

TR ID	Case No.	Data Entered	Expected Result
TR*002*	**TC_{TR002}_01**	Login as Tenant A User. Click "Export Audit to CSV".	The report generation process (from click to download prompt) completes within 10 seconds.
TR*003*	**TC_{TR003}_01**	Generate a 100-row audit report.	The browser starts the CSV download within 2 seconds of the report generation message appearing.

Section B: Black-box Testing

B1 Object Class

B1.1 Equivalence Partitioning and Boundary Value Analysis

Table 5. Equivalence Partition and Input Range

Object class	Attributes	Equivalence Partition and Input Range
AuditInstance	Number of Records	Valid: - 1 record (boundary) - 500 records (typical)
	TenantID (GUID/int)	Valid: - Existing valid TenantId Invalid: - TenantId not matching logged-in user's tenant

B1.2 Test Cases

Table 6. Object Class Based Test Cases

Object name: AuditInstance

Method name:ExportAuditToCsv

Case No.	Equivalence Class	Representative (BVA)	Expected Result
TC*001*	Valid Export - Single Audit Record List contains 1 AuditInstance.	Login as Tenant A User. Audit Records (Tenant A): 1. Click "Export Audit to CSV".	Report generated and downloaded. CSV contains 1 row of data (plus headers) matching the audit instance details.
TC*002*	Valid Export - Typical Number of Records	Login as Tenant A User.	Report generated and downloaded. CSV

	List contains 500 AuditInstance records.	Audit Records (Tenant A): 500 records. Click "Export Audit to CSV".	contains 500 rows of data.
TC*003*	Tenant User - Data Isolation Ensure only current tenant's data is exported.	Login as Tenant A User. Audit Records (Tenant A): 5. Audit Records (Tenant B): 3. Click "Export Audit to CSV".	The CSV file contains only the 5 audit records from Tenant A. Data from Tenant B is absent.

5.0 Filing & Document Repository Module

Section A: Requirements-based Testing

A1 Functional Requirements

A1.1 Test Requirements (TR)

Table 1. List of Functional Test Requirements

Use Case (UC)	TR ID	Test Requirements
UC015 Create Folder	$TR_{015\text{-}01}$	Verify that a new compliance folder can be created successfully with valid Name, Compliance Category, and Description.
	$TR_{015\text{-}02}$	Verify that the system assigns the current TenantId to the newly created compliance folder.
	$TR_{015\text{-}03}$	Verify that multiple required documents can be added to a new compliance folder during its creation.
	$TR_{015\text{-}04}$	Verify that an error message is displayed when attempting to create a compliance folder with invalid or missing required fields (e.g., Name, Compliance Category).
	$TR_{015\text{-}05}$	Verify that CreatedBy field is populated with the authenticated user's name or "System" if the user is not authenticated.
UC016 Upload Document	$TR_{016\text{-}01}$	Verify that a document can be successfully uploaded to an existing compliance folder.
	$TR_{016\text{-}02}$	Verify that a document can be successfully uploaded and associated with a specific required document within a compliance folder.
	$TR_{016\text{-}03}$	Verify that upon successful upload of a document to a required document, the IsSubmitted status of the

		RequiredDocument is updated to true, and SubmissionDate and SubmittedBy are populated.
	$TR_{016\text{-}04}$	Verify that the system handles invalid file uploads (e.g., no file selected) gracefully and displays an appropriate error message without uploading any files.
	$TR_{016\text{-}05}$	Verify that the system correctly stores document metadata (FileName, FilePath, FileType, FileSize, Description, UploadDate, UploadedBy, Status) in the database upon successful upload.

A1.2 Test Cases

Table 2. List of Functional Test Cases

TR ID	Case No.	Data Entered	Expected Result
TR*018-01*	**$TC_{018\text{-}01}$_01**	New folder with valid name, category, description, no required documents.	Folder created; TenantId, CreatedBy, CreatedDate, Status (Active) populated; Redirect to Index with success.
TR*018-01*	**$TC_{018\text{-}01}$_02**	New folder with valid name, category, description, and two required documents.	Folder created; two RequiredDocument entries linked to folder (IsRequired=true, IsSubmitted=false); Redirect to Index with success.
TR*018-04*	**$TC_{018\text{-}04}$_01**	Attempt to create folder with empty Name.	Folder not created; ModelState error for Name; remains on Create Folder view.

TR*019-01*	**TC_{019-01}01**	Upload general document to existing ComplianceFolder (ID: 1) with no specific RequiredDocument.	File uploaded to server; Document entry created for folder 1 (RequiredDocumentId is null); Redirect to FolderDetails with success.
TR*019-02*	**TC_{019-02}_01**	Upload document for specific RequiredDocument (ID: 201) in ComplianceFolder (ID: 2).	File uploaded; Document entry created for folder 2, linked to required doc 201; RequiredDocument 201 updated (IsSubmitted=true, SubmissionDate, SubmittedBy); Redirect to FolderDetails with success.
TR*019-04*	**TC_{019-04}_01**	Attempt to upload with no file selected for ComplianceFolder (ID: 3).	No file uploaded; no Document entry created; error message "No files were uploaded"; remains on Upload Document view.

A2 Non-Functional Requirements

A2.1 Test Requirements (TR)

Table 3. List of Non-Functional Test Requirements

Non-functional	TR ID	Test Requirements
Security	TR018_06 TR019_06	Verify that only authenticated and authorized users can access document repository functionalities (e.g., Create, Upload, Edit, Delete).

	TR018_07 TR019_07	Verify that file uploads are scanned for common malware signatures.
Usability	TR018_09 TR019_09	Verify that error messages are clear, concise, and provide actionable feedback to the user.

A2.2 Test Cases

Table 4. List of Non-Functional Test Cases

TR ID	Case No.	Data Entered	Expected Result
TR018_06	**$TC_{TR018\text{-}06}$_01**	User attempts to access Create Folder page without being authenticated.	User is redirected to login page or an "Access Denied" page.
TR018_06	**$TC_{TR018\text{-}06}$_02**	Unauthenticated user attempts to upload a document via a direct POST request to DocumentRepositoryController/UploadDocument.	Request is rejected with a 401 Unauthorized or 403 Forbidden status.
TR018_07	**$TC_{TR018\text{-}07}$_01**	User uploads a test file containing a known EICAR test string.	The system detects the EICAR string and either rejects the upload or quarantines the file. An appropriate error/warning message is displayed.
TR018_08	**$TC_{TR018\text{-}08}$_01**	Attempt to create a folder with an empty name field.	An error message "The Name field is required." (or similar clear message) is displayed

			next to the Name field, and the user remains on the Create Folder page
TR018_08	**$TC_{TR018\text{-}08_}$02**	Attempt to upload a document without selecting a file.	An error message "No files were uploaded. Please select at least one file." is displayed as TempData["ErrorMessage"], and the user remains on the Upload Document view.

Section B: Black-box Testing

B1 Object Class

B1.1 Equivalence Partitioning and Boundary Value Analysis

Table 5. Equivalence Partition and Input Range

Object class	Attributes	Equivalence Partition and Input Range
ComplianceFolder	Name (string)	**Valid:** - Valid string (1 to 200 characters) **Invalid:** - Empty string (0 characters) - String exceeding 200 characters - Null string
	ComplianceCategoryId (int)	**Valid:** - Valid existing ComplianceCategory ID (e.g., 1, 5, 100) **Invalid:** - Non-existent ComplianceCategory ID (e.g., 0, -1, 99999) - Non-integer input (e.g., "abc")
	Description (string)	**Valid:** - Empty string (0 characters)

		- Valid string (1 to 1000 characters) - Null string **Invalid:** - String exceeding 1000 characters
	Status (Enum FolderStatus)	**Valid:** - Active - Archived - Completed **Invalid:** - Any other integer value not defined in FolderStatus enum (e.g., 3, -1) - Non-enum string (e.g., "Pending")

Object class	**Attributes**	**Equivalence Partition and Input Range**
RequiredDocument	DocumentName (string)	**Valid:** - Valid string (1 to 200 characters) **Invalid:** - Empty string (0 characters) - String exceeding 200 characters - Null string

	IsRequired (bool)	**Valid:** - Empty string (0 characters) - Valid string (1 to 1000 characters) - Null string **Invalid:** - String exceeding 1000 characters
	IsRequired (bool)	**Valid:** - True - False
	ComplianceFolderId (int)	**Valid:** - Valid existing ComplianceFolder ID (e.g., 1, 5) **Invalid:** - Non-existent ComplianceFolder ID (e.g., 0, -1, 99999) - Non-integer input

Object class	**Attributes**	**Equivalence Partition and Input Range**
Document	FileName (string)	**Valid:** - Valid string (1 to 200 characters), e.g., "report.pdf" **Invalid:** - Empty string (0 characters)

		- String exceeding 200 characters - Null string
	Description (string)	**Valid:** - Empty string (0 characters) - Valid string (1 to 1000 characters) - Null string **Invalid:** - String exceeding 1000 characters
	FileType (string)	**Valid:** - Recognized file types (e.g., ".pdf", ".docx", ".jpg") **Invalid:** - Unrecognized/unsupported file types (e.g., ".exe", ".zip")
	FileSize (long)	Valid: - Positive long integer (e.g., 1, 1024, 5242880 for 5MB) Invalid: - Zero (0) or negative integer - Exceeds server's max file size limit (e.g., 2GB, 4GB, depends on server config)
	ComplianceFolderId (int)	**Valid:** - Valid existing ComplianceFolder ID (e.g., 1, 5)

		Invalid: - Non-existent ComplianceFolder ID (e.g., 0, -1, 99999) - Non-integer input
	RequiredDocumentId (int?)	**Valid:** - Valid existing RequiredDocument ID (e.g., 1, 5) - Null (optional link) **Invalid:** - Non-existent RequiredDocument ID (e.g., 99999) - Non-integer input when provided (e.g., "abc")
	Status (Enum DocumentStatus)	**Valid:** - Active, Archived, PendingReview, Approved, Rejected **Invalid:** - Any other integer value not defined in DocumentStatus enum (e.g., 5, -1) - Non-enum string

B1.2 Test Cases

Table 6. Object Class Based Test Cases

Object name: ComplianceFolder

Method name: CreateFolder (POST)

Case No.	Equivalence Class	Representative (BVA)	Expected Result
TC*018_01*	Valid Name (Min Length), Valid Category, Valid Description (Empty)	Name: "A", CategoryId: 1, Description: ""	Folder created successfully. Redirects to Index with success message.
TC*018_02*	Valid Name (Max Length), Valid Category, Valid Description (Max Length)	Name: (200 chars), CategoryId: 1, Description: (1000 chars)	Folder created successfully. Redirects to Index with success message.
TC*018_03*	Invalid Name (Empty)	Name: "", CategoryId: 1, Description: "Test Desc" .	Model state invalid. Returns to View with error for Name
TC*018_04*	Invalid Name (Exceeds Max Length)	Name: (201 chars), CategoryId: 1, Description: "Test Desc"	Model state invalid. Returns to View with error for Name.
TC*018_05*	Invalid Category ID (Non-existent)	Name: "Valid Name", CategoryId: 99999, Description: "Test Desc"	Model state invalid or database error. Returns to View with error. (Assuming validation for

			FK is handled by DB/framework)
TC*018_06*	Description (Null)	Name: "Valid Name", CategoryId: 1, Description: null	Folder created successfully. Redirects to Index with success message. (Description is nullable)
TC*018_07*	Invalid Description (Exceeds Max Length)	Name: "Valid Name", CategoryId: 1, Description: (1001 chars)	Model state invalid. Returns to View with error for Description.
TC*018_08*	Missing Tenant ID (system-level error)	N/A (simulated by empty _tenantService.GetCurrentTenantId())	Returns to View with error message: "Cannot create folder: Current tenant not identified."

Object name: ComplianceFolder

Method name: EditFolder (POST)

Case No.	**Equivalence Class**	**Representative (BVA)**	**Expected Result**
TC*018_09*	Valid Folder Update (Name, Category, Description)	FolderId: 1, New Name: "Updated Folder", New CategoryId: 2, New Desc: "Updated Desc"	Folder updated successfully. Redirects to Index with success message.
TC*018_10*	Update with Invalid Name (Empty)	FolderId: 1, New Name: "", CategoryId: 1	Model state invalid. Returns to View with error for Name.

TC*018_11*	Add New Required Document	FolderId: 1, New Required Doc: { "New Req Doc", "Desc", true }	Folder updated successfully. New RequiredDocument added to folder. Redirects to Index with success message.
TC*018_12*	Update Existing Required Document	FolderId: 1, Existing Req Doc (ID 5): { "Updated Req Doc Name", "Updated Desc", true }	Folder updated successfully. Existing RequiredDocument properties (DocumentName, Description) updated. Redirects to Index with success message.
TC*018_13*	Remove Required Document (No Uploads)	FolderId: 1, Remove Req Doc (ID 6, no associated Documents)	Folder updated successfully. RequiredDocument removed from folder. Redirects to Index with success message.
TC*018_14*	Attempt to Remove Required Document (With Uploads)	FolderId: 1, Remove Req Doc (ID 7, has associated Documents)	Model state invalid. Returns to View with error: "Cannot remove '...' because it has associated uploaded documents. Please remove the uploads first if you wish to remove this requirement."

TC*018_15*	Invalid Folder ID for Edit	FolderId: 99999 TempData["ErrorMessage"]:	"Folder not found or does not belong to your organization." Returns NotFound().

Object name: ComplianceFolder

Method name: ArchiveFolder (POST)

Case No.	Equivalence Class	Representative (BVA)	Expected Result
TC*018_66*	Valid Archive (Active Folder)	id: 1 (Active Folder)	Folder status updated to 'Archived'. LastModifiedDate and LastModifiedBy updated. TempData["SuccessMessage"] shown. Redirects to Index.
TC*018_17*	Folder Not Found	id: 99999	TempData["ErrorMessage"]: "Folder not found." Returns NotFound().

Object name: ComplianceFolder

Method name: ActivateFolder (POST)

Case No.	Equivalence Class	Representative (BVA)	Expected Result
TC*018_18*	Valid Activate (Archived Folder)	id: 1 (Archived Folder)	Folder status updated to 'Active'. LastModifiedDate and LastModifiedBy updated. TempData["SuccessMessage"] shown. Redirects to Index.

TC*018_19*	Folder Not Found	id: 99999	TempData["ErrorMessage"]: "Folder not found." Returns NotFound().

Object name: Document

Method name: UploadDocument (POST)

Case No.	Equivalence Class	Representative (BVA)	Expected Result
TC*019_01*	Valid File, Valid Folder, No Required Doc	File: "test.pdf" (1MB), FolderId: 1, RequiredDocId: null	Document uploaded successfully. Document record created in DB. TempData["SuccessMessage"] shown. Redirects to FolderDetails.
TC*019_02*	Valid File, Valid Folder, Valid Required Doc	File: "report.docx" (200KB), FolderId: 1, RequiredDocId: 5	Document uploaded successfully. Document record created in DB. Associated RequiredDocument's IsSubmitted set to true, SubmissionDate and SubmittedBy updated. TempData["SuccessMessage"] shown. Redirects to FolderDetails.
TC*019_03*	File Name Exceeds Max Length	File: (201 char .txt), FolderId: 1	Model state invalid. Returns to View with error. (Assuming

			validation on client or server-side ViewModel)
TC*019_04*	Invalid Folder ID	File: "a.pdf", FolderId: 99999	TempData["ErrorMessage"]: "Compliance folder not found." Redirects to Index.
TC*019_05*	Invalid Required Document ID	File: "a.pdf", FolderId: 1, RequiredDocId: 99999	TempData["ErrorMessage"]: "Required Document not found or mismatch." (during GET action for ViewModel population) or RequiredDocument not updated if ID mismatch. Returns to View with model errors if ModelState.IsValid fails.
TC*019_06*	Unsupported File Type	File: "malware.exe", FolderId: 1	ModelState error for file type (if client-side validation) or "application/octet-stream" type assigned (server-side default). Could result in "Error uploading file...". Returns to View.
TC*019_07*	File Size Zero	File: "empty.txt" (0 bytes), FolderId: 1	TempData["ErrorMessage"]: "No files were uploaded. Please select

			at least one file." Returns to View.
TC*019_08*	Multiple Files Upload (Mix of Valid/Invalid)	File1: "valid.pdf", File2: "invalid.exe", FolderId: 1	Valid files upload successfully; invalid files generate errors. TempData["SuccessMess age"] for successful uploads, ModelState errors for failed ones.
TC*019_09*	Upload to Archived Folder (if disallowed)	File: "doc.pdf", FolderId: (Archived Folder ID)	System prevents upload (e.g., ModelState error or TempData["ErrorMessage "] if logic exists in controller to prevent uploads to non-active folders). (Implicit, based on typical business rules, not explicit in provided code)

Object name: Document

Method name: DeleteDocument (POST)

Case No.	**Equivalence Class**	**Representative (BVA)**	**Expected Result**
TC*019_10*	Valid Delete (Document with No RequiredDoc)	id: 10, folderId: 1 (Doc not linked to ReqDoc)	Document record removed from DB. Physical file deleted from server. TempData["SuccessMess

			age"] shown. Redirects to FolderDetails.
TC*019_11*	Valid Delete (Document for ReqDoc, but not sole)	id: 11, folderId: 1 (Doc linked to ReqDoc 5, but ReqDoc 5 has other docs)	Document record removed from DB. Physical file deleted from server. RequiredDocument's IsSubmitted remains true. TempData["SuccessMessage"] shown. Redirects to FolderDetails
TC*019_12*	Valid Delete (Document for ReqDoc, and sole)	id: 12, folderId: 1 (Doc linked to ReqDoc 6, ReqDoc 6 has NO other docs)	Document record removed from DB. Physical file deleted from server. RequiredDocument's IsSubmitted set to false, SubmissionDate and SubmittedBy nulled. TempData["SuccessMessage"] shown. Redirects to FolderDetails.

6.0 User and Team Management Module

Section A: Requirements-based Testing

A1 Functional Requirements

A1.1 Test Requirements (TR)

Table 1. List of Functional Test Requirements

Use Case (UC)	TR ID	Test Requirements
UC021 Manage User	TR_{021_001}	Verify that a new user can be successfully created with valid details.
	TR_{021_002}	Verify that an existing user's details can be successfully updated.
	TR_{021_003}	Verify that a user's status (e.g., active/inactive) can be changed.
	TR_{021_004}	Verify that a user can be successfully deleted.
	TR_{021_005}	Verify that the system prevents the creation of a user with an already existing username or email.
	TR_{021_006}	Verify that the system displays an appropriate error message for invalid user input during creation/update.
	TR_{021_007}	Verify that a list of all users can be viewed, including their basic details.
UC022 Manage Role	TR_{022_001}	Verify that an administrator can assign one or more roles to a user.
	TR_{022_002}	Verify that an administrator can remove a role from a user.
	TR_{022_003}	Verify that the system correctly reflects the assigned roles for a user.

A1.2 Test Cases

Table 2. List of Functional Test Cases

TR ID	Case No.	Data Entered	Expected Result
TR*021_001*	**TC_{TR021_001}_01**	User Creation - Valid: Username: "testuser1" Email: "testuser1@example.com" Password: "Password123!" IsActive: True	User "testuser1" is created and visible in the user list.
	TC_{TR021_001}_02	User Creation - Valid (Another): Username: "newadmin" Email: "newadmin@domain.com" Password: "Admin@2025" IsActive: True	User "newadmin" is created and visible in the user list.
TR*021_002*	**TC_{TR021_002}_01**	User Update - Email: Select "testuser1" New Email: "updated.testuser1@example.com"	Email for "testuser1" is updated to "updated.testuser1@example.com".
	TC_{TR021_002}_02	User Update - Username: Select "newadmin" New Username: "corporate_admin"	Username for "newadmin" is updated to "corporate_admin".
TR*021_003*	**TC_{TR021_003}_01**	Toggle User Status - Deactivate: Select "testuser1"	"testuser1" status changes to inactive. User cannot log in.

		Toggle IsActive to False	
	TC_{TR021_003}_02	Toggle User Status - Activate: Select "testuser1" Toggle IsActive to True	"testuser1" status changes to active. User can log in.
TR*021_004*	**TC_{TR021_004}_01**	Delete User: Select "testuser1"	"testuser1" is successfully deleted from the system.
TR*022_001*	**TC_{TR022_001}_01**	Select “Manager” role to “testuser1”	"Manager" role is assigned to "testuser1".
TR*022_002*	**TC_{TR022_002}_01**	Unselect “Manager” role from “testuser1”	"Manager" role is removed from "testuser1".
TR*022_001*	**TC_{TR022_001}_01**	Select "newadmin" View assigned roles	The system correctly displays "Admin" as the assigned role for "newadmin".

A2 Non-Functional Requirements

A2.1 Test Requirements (TR)

Table 3. List of Non-Functional Test Requirements

Non-functional	TR ID	Test Requirements
Performance	TR001	The user list page should load within 3 seconds.
	TR002	User creation/update operations should complete within 1 second.
Security	TR003	User passwords must be stored securely (hashed).
	TR004	Only authorized users with appropriate roles (SuperAdmin) can access and perform user and role management operations.

A2.2 Test Cases

Table 4. List of Non-Functional Test Cases

TR ID	Case No.	Data Entered	Expected Result
TR*001*	**$TC_{TR001_}$01**	Simulate 100 users in the database. Navigate to User List page.	Page loads and displays all users within 3 seconds.
TR002	**$TC_{TR002_}$01**	Create a new user with valid data.	User creation completes within 1 second.
	$TC_{TR002_}$02	Update an existing user's details.	User update completes within 1 second.
TR003	**$TC_{TR003_}$01**	Create a new user with password "TestPass123!".	Password for the created user is stored as a hashed value, not plain text.
TR004	**$TC_{TR004_}$01**	Log in as a non-superadmin user. Attempt to access User/Role Management.	Access denied or redirect to an unauthorized page.
	$TC_{TR004_}$02	Log in as a superadmin admin. Attempt to access User/Role Management.	Access granted. SuperAdmin can perform all management operations.

Section B: Black-box Testing

B1 Object Class

B1.1 Equivalence Partitioning and Boundary Value Analysis

Table 5. Equivalence Partition and Input Range

Object class	Attributes	Equivalence Partition and Input Range
IdentityUser	UserName (string)	Valid: - 5 to 50 alphanumeric characters (e.g., "user123", "JohnDoe") Invalid: - Less than 5 characters (e.g., "abc") - More than 50 characters - Contains special characters (e.g., "user@123") - Empty string - Already existing username
	Email (string)	Valid: - Standard email format (e.g., "name@domain.com") - Minimum length email (e.g., "a@b.co") Invalid: - Missing "@" (e.g., "https://www.google.com/search?q=name.domain.com") - Missing domain (e.g., "name@.com") - Missing local part (e.g., "@domain.com") - Empty string - Already existing email
	PasswordHash (string) - *For*	Valid:

	password input, before hashing	- Strong password (e.g., minimum 8 chars, mix of upper/lower/digit/special) - Maximum allowed length password Invalid: - Too short (e.g., "abc") - Too long - Weak password (e.g., "password123") - Empty string
	Is Active (boolean)	Valid: - True - False
	TenantId	Valid: - Valid TenantId

Object class	**Attributes**	**Equivalence Partition and Input Range**
AspNetRoles	Name (string)	Name (string)Valid: - 3 to 30 alphanumeric characters

B1.2 Test Cases

Table 6. Object Class Based Test Cases

Object name: IdentityUser

Method name: Create

Case No.	Equivalence Class	Representative (BVA)	Expected Result
TC*001*	Valid Username, Valid Email, Valid Password	Username: "user1" Email: "user1@example.com" Password: "P@ssw0rd1"	User "user1" created successfully.
TC*002*	Invalid Username	Username: "abc" Email: "test@example.com" Password: "ValidPass123!"	Error: Username too short
TC*003*	Invalid Username (too long)	Username: "ThisIsAVeryLongUsernameThatExceedsTheFiftyCharacterLimitForSure!" Email: "test2@example.com" Password: "ValidPass123!"	Error: Username too long
TC*004*	Invalid Email (missing @)	Email: "https://www.google.com/search?q=invalid.email.com" Password: "ValidPass123!"	Error: Invalid email format.
TC*005*	Invalid Email (empty)	Username: "validuser2" Email: "" Password: "ValidPass123!"	Error: Email cannot be empty.
TC*006*	Invalid Password (too short)	Username: "validuser3"	Error: Password too short (e.g.,

		Email: "test4@example.com" Password: "abc"	"Password must be at least 8 characters").
TC*007*	Invalid Password (weak)	Username: "validuser4" Email: "test5@example.com" Password: "password"	Error: Password is too weak (e.g., "Password must contain a mix of character types").
TC*008*	Existing Email	Username: "newuser" Email: (An already existing email in the system) Password: "ValidPass123!"	Error: Email already exists.

Object name: IdentityUser

Method name: Edit

Case No.	**Equivalence Class**	**Representative (BVA)**	**Expected Result**
TC*009*	Valid username on update	New Username: "userone" Email: "user1@example.com" Password: "P@ssw0rd1"	New username updated successfully
TC*010*	Invalid username on update	New Username: "u" Email: "user1@example.com" Password: "P@ssw0rd1"	Error: Username too short
TC*011*	Invalid email on update	New Username: "userone" Email: "user1example" Password: "P@ssw0rd1"	Error: Invalid email format.

TC*012*	Update user's TenantId (SuperAdmin only - Valid)	Username: "userone Email: "user1@example.com" Password: "P@ssw0rd1" New TenantId: TenantB	userone is now belongs to TenantB

Object name: IdentityUser

Method name: CreateTenant

Case No.	Equivalence Class	Representative (BVA)	Expected Result
TC*013*	Valid Tenant Creation	New Username: "userone" Email: "user1@example.com" Password: "P@ssw0rd1" Tenant: GlobalEnterprises	Tenant "GlobalEnterprises" created.

Object name: AspNetRoles

Method name: ManageRoles

Case No.	Equivalence Class	Representative (BVA)	Expected Result
TC*014*	Valid Role Assignment to user. Assign an existing role to a user.	Login as: Superadmin Username: “userone” Role: “Manager”	Role “Manager” successfully assigned to userone.
TC*015*	Valid Role Changes to users. Change an existing role for a user.	Login as: Superadmin Username: “userone” Current Role: “Manager” New Role: “Admin”	New role “Admin” successfully

			assigned to userone.