Virtual Try-On

Test Document

Session: 2021 – 2025

Submitted by:

Sehrish Saddique 2021-SE-12

Kausar Fatima 2021-SE-25

Laiba Amber Ejaz 2021-SE-37

Supervised by:

Ma'am Alina

Department of Computer Science, New Campus

University of Engineering and Technology

Lahore, Pakistan

Contents

List of Figures

List of Tables

Chapter 1

Experimentation and Testing

Testing methodologies are employed in the development process to ensure software functions effectively across various environments and platforms. They can be categorized into functional and non-functional testing. Functional testing aligns the application with business requirements, validating each software component's expected behavior based on provided use cases from the design team or business analyst. These tests are typically conducted sequentially and encompass:

- Unit testing
- Integration testing
- System testing
- Acceptance testing

Non-functional testing methods incorporate all test types focused on the operational aspects of a piece of software. These include:

- Performance testing
- Security testing
- Usability testing
- Compatibility testing

1.1 Unit Testing

Unit testing is a type of testing in software development that focuses on testing individual units or components of a system in isolation, typically at the function or method level. It is designed to verify that each unit of code performs as expected and meets its specified requirements. unit testing would involve testing individual functions or modules of the system to ensure their correctness and reliability[1].

1.1.1 Sign Up Unit Testing

The signup unit test in Table 1.1 is conducted to ensure the reliability, security, and functionality of the registration process. It verifies aspects such as username format, email validity, password strength, and error handling for incorrect credentials. Additionally, it confirms the successful creation of user accounts.

TABLE 1.1: Unit Test Case:Admin Sign-Up

Test Case ID	TC_001
Test Case Description	Verify the sign-up functionality for a new user.
Scenario	Ensure that users can register successfully with valid details and receive appropriate error messages for invalid inputs.
Prerequisites	The email should not already be registered.
Test Data	ValidName = John Doe, ValidEmail = user@exa.com, ValidPassword = Pass@123, ConfirmPassword = Pass@123 InvalidName = John123, TooLongNameExceedingLimit, InvalidEmail = userexample.com, user@.com, ExistingEmail = existing@admin.com, MismatchedPassword = Pass@321, WeakPassword = pass123, weakpass

Step No.	**Step Detail**	**Expected Results**	**Actual Results**	**Status**
1	Navigate to Sign-Up Page	Page should be opened	As Expected	Successful
2	Enter ValidName, ValidEmail, ValidPassword, ConfirmPassword and click SignUp button.	Registration successful, redirects to login page.	As Expected	Successful
3	Enter InvalidName	Display "Invalid name format. Use only letters, max 25 characters."	As Expected	Successful
4	Enter InvalidEmail	Display "Invalid email format."	As Expected	Successful
5	Enter ExistingEmail	Display "Email is already registered."	As Expected	Successful
6	Enter ValidPassword and MismatchedPassword	Display "Passwords do not match."	As Expected	Successful
7	Enter WeakPassword	Display "Password must be at least 8 characters, alphanumeric, with 1 special character (*, @, -)."	As Expected	Successful
8	Leave EmptyFields	Display "All fields are required."	As Expected	Successful

1.1.2 Login Unit Testing

The login unit test in Table 1.2 is conducted to ensure the login process functions reliably and securely. It verifies email format and password strength, authenticates users with correct credentials, and appropriately handles incorrect inputs. Additionally, it confirms that login actions do not inadvertently create accounts and correctly handle empty or nonexistent email and password combinations.

TABLE 1.2: Unit Test Case:Admin Login

Test Case ID	TC_002
Test Case Description	Verify the login functionality for the admin user.
Scenario	Ensure the admin can log in using valid credentials, and invalid credentials are rejected with appropriate messages.
Prerequisites	Admin account must already be registered.
Test Data	ValidEmail = rightemail@gmail.com ValidPassword = Admin@123 InvalidEmail = wrongemail@gmail.com InvalidPassword = wrong@123

Step No.	**Step Detail**	**Expected Results**	**Actual Results**	**Status**
1	Navigate to Login page	Page should be opened	As Expected	Successful
2	Enter Email only	Display "Please enter your password."	As Expected	Successful
3	Enter Password only	Display "Please enter your email."	As Expected	Successful
4	Click Login without entering Email and Password	Display "Email and password are required."	As Expected	Successful
5	Enter correct Email and Password and click Login button.	Login successful, redirects to home page.	As Expected	Successful
6	Enter correct Email but incorrect password	Display "Invalid password."	As Expected	Successful
7	Enter incorrect Email but correct password	Display "Email not found."	As Expected	Successful
8	Enter incorrect Email and password	Display "Invalid email or password."	As Expected	Successful

1.1.3 Update Profile Unit Testing

The update profile unit test in Table 1.3 ensures the reliability and security of the profile modification process. It validates name format, password strength, and error handling for incorrect credentials. Additionally, it verifies the successful update of user details.

TABLE 1.3: Unit Test Case:Admin Update Profile

Test Case ID	TC_003

Test Case Description	Verify the profile update functionality for the admin user.
Scenario	Ensure the admin can update their profile details with valid data and receive appropriate error messages for invalid inputs.
Prerequisites	Admin must be logged in.
Test Data	ValidName = "Alice Johnson" ValidPassword = "Secure@123" InvalidName = "TooLongAdminNameExceedingLimit" WeakPassword = "pass123" NoSpecialCharPassword = "Password123"

Step No.	**Step Detail**	**Expected Results**	**Actual Results**	**Status**
1	Navigate to Profile Update Page	Page should be opened	As Expected	Successful
2	Enter a valid Name and Password and click update button.	Profile updates successfully	As Expected	Successful
3	Enter an invalid Name (too long)	Display "Name cannot exceed 25 characters."	As Expected	Successful
4	Enter a weak Password (e.g., "pass123")	Display "Password must be at least 8 characters long and contain alphanumeric characters."	As Expected	Successful
5	Leave Name or Password empty	Display "Please fill in all required fields."	As Expected	Successful

1.1.4 Add Category Unit Testing

The add category unit test in Table 1.4 ensures that the category creation process functions correctly and enforces proper validation rules. It verifies the format of category names, ensures image selection is mandatory, handles errors for invalid inputs, and confirms successful category creation.

TABLE 1.4: Unit Test Case:Admin Add Category

Test Case ID	TC_004
Test Case Description	Verify the category creation functionality for the admin user.
Scenario	Ensure that the admin can successfully add a category with valid data and receive appropriate error messages for invalid inputs.
Prerequisites	Admin must be logged in.
Test Data	ValidName = "Clothing" ValidImage = "cloth.jpg.bytes" TooLongCategoryName = "ThisCategoryNameIsWayTooLong" InvalidCategoryName = "Sports@123" or "G@mes" ExistingCategory = "ExistingCategory"

Step No.	Step Detail	Expected Results	Actual Results	Status
1	Navigate to Add Category Page	Page should be opened	As Expected	Successful
2	Enter a valid Category Name, select an Image and click add category button.	Category should be added successfully	As Expected	Successful
3	Leave the Category Name empty	Display "Category name cannot be empty."	As Expected	Successful
4	Enter a Category Name exceeding 25 characters	Display "Category name cannot exceed 25 characters."	As Expected	Successful
5	Enter an invalid Category Name (contains special characters or numbers)	Display "Category name can only contain alphabetic characters and spaces."	As Expected	Successful
6	Try to add a Category without selecting an Image	Display "Please select an image."	As Expected	Successful
7	Try to add a Category with an existing name	Display "Category already exists."	As Expected	Successful

1.1.5 Update Category Unit Testing

The update category unit test in Table 1.5 ensures the accuracy and validation of the category modification process. It verifies name format, image updates, error handling for invalid inputs, and ensures successful category updates.

Table 1.5: Unit Test Case:Update Category

Test Case ID	TC_005
Test Case Description	Verify the update category functionality for admin users.
Scenario	Ensure admin can update a category name and image successfully, while invalid inputs are handled correctly.
Prerequisites	At least one category must exist.
Test Data	ValidCategoryName = Clothing ValidImage = clothing.jpg.bytes InvalidCategoryName = G@dgets123 DuplicateCategoryName = ExistingCategory TooLongCategoryName = ThisCategoryNameExceedsTwentyChars

Step No.	Step Detail	Expected Results	Actual Results	Status
1	Navigate to Category Update Page	Page should be opened	As Expected	Successful
2	Enter a valid category name and image and click save button.	Category updates successfully	As Expected	Successful
3	Leave empty fields.	Display "All fields are required."	As Expected	Successful
4	Enter a category name exceeding 20 characters	Display "Category name must be up to 20 characters."	As Expected	Successful
5	Enter an invalid category name	Display "Category name must be alphabetic and up to 20 characters long."	As Expected	Successful
6	Enter a duplicate category name	Display "Category with this name already exists."	As Expected	Successful
7	Image fails to update	Display "Failed to save image or update folder."	As Expected	Successful

1.1.6 Delete Category Unit Testing

The delete category unit test in Table 1.6 ensures that categories are properly removed when requested. It verifies that deleting a category also removes its associated items, prevents deletion of non-existent categories, and confirms appropriate error handling for failed deletions.

TABLE 1.6: Unit Test Case:Delete Category

Test Case ID	TC_006
Test Case Description	Verify the delete category functionality for admin users.
Scenario	Ensure admin can delete a category successfully, while invalid deletion attempts are handled correctly.
Prerequisites	At least one category must exist.
Test Data	ValidCategorySelected = Accessories CategoryWithItems = Jeans

Step No.	**Step Detail**	**Expected Results**	**Actual Results**	**Status**
1	Navigate to Category Management Page	Page should be opened	As Expected	Successful
2	Select a valid category and click delete	Category deleted successfully	As Expected	Successful
3	Attempt to delete without selecting a category	Display "Please select a category first."	As Expected	Successful
4	Attempt to delete a category containing items	Display confirmation: "Deleting category will delete items within category as well."	As Expected	Successful
5	Cancels deletion when prompted	User cancels, category remains	As Expected	Successful

1.1.7 Search Category Unit Testing

The search category unit in Table 1.7 test checks the functionality of category lookup based on user input. It verifies that valid search queries return correct results and ensures no results are displayed for non-matching queries. Additionally, it ensures that searches are case-insensitive.

TABLE 1.7: Unit Test Case:Search Category

Test Case ID	TC_007
Test Case Description	Verify the category search functionality for admin users.
Scenario	Ensure admin can search for categories successfully, and appropriate messages are shown when no matching categories are found.
Prerequisites	At least one category should exist.
Test Data	ValidQuery = "Shirts" InvalidQuery = "Accessories" (not present) CaseInsensitiveQuery = "shirts" (should return same results as "Shirts")

Step No.	**Step Detail**	**Expected Results**	**Actual Results**	**Status**
1	Navigate to Category Management Page	Page should be opened	As Expected	Successful
2	Enter a valid category name in the search bar	Returns categories matching "Shirts"	As Expected	Successful
3	Enter a non-existing category name in the search bar	No category should be found.	As Expected	Successful
4	Enter an empty search query	Resets to all fetched categories.	As Expected	Successful
5	Enter a category name in a different case (e.g., "shirts")	Returns same results as "Shirts" (case-insensitive search)	As Expected	Successful

1.1.8 Add Item Unit Testing

The add item unit test in Table 1.8 verifies the correct addition of an item within a category. It ensures required fields like name, description, price, quantity, fabric,

size, and color are filled correctly, validates .fbx file selection, checks for duplicate item names, and confirms proper error handling for invalid inputs.

Table 1.8: Unit Test Case:Add Item

Test Case ID	TC_008
Test Case Description	Verify the add item functionality for admin users.
Scenario	Ensure admin can add an item successfully, while invalid additions are handled correctly.
Prerequisites	At least one category must exist to add an item.
Test Data	ValidItem = Jacket, Leather Jacket, 2500, 10, Cotton, M, Black, jacket.fbx InvalidPrice = "-xyz" InvalidQuantity = -3

Step No.	**Step Detail**	**Expected Results**	**Actual Results**	**Status**
1	Navigate to Add Item Page	Page should be opened	As Expected	Successful
2	Enter valid item details and click add item.	Item added successfully	As Expected	Successful
3	Leave fields empty and submit	Display "Fill all the fields."	As Expected	Successful
4	Enter invalid price format	Display "Invalid price format."	As Expected	Successful
5	Enter negative quantity	Display "Quantity must be greater than zero."	As Expected	Successful
4	Enter invalid quantity format	Display "Invalid quantity format."	As Expected	Successful
5	Enter negative price	Display "Price must be non-negative."	As Expected	Successful

1.1.9 Update Item Unit Testing

The update item unit test in Table 1.9 ensures the reliability of updating item details. It verifies that the system enforces name formatting rules, description length, valid pricing, and quantity validation. Additionally, it checks for missing selections and duplicate names in the same category.

Table 1.9: Unit Test Case:Admin Update Item

Test Case ID	TC_009
Test Case Description	Verify that an admin can successfully update an item with valid data and receive proper error messages for invalid inputs.
Scenario	Ensure that the admin can modify item details while adhering to validation constraints.
Prerequisites	Admin must be logged in and have access to the item list.
Test Data	ValidName = "Winter Jacket" InvalidName = "SuperLongItemNameExceedingLimit" ValidPrice = 999.99 InvalidPrice = -50 or 100000001 ValidQuantity = 100 InvalidQuantity = -1 or 100000001 ValidDescription = "A stylish winter jacket with wool lining." InvalidDescription = (Empty) or more than 250 characters ValidSelections = Fabric, Size, Color chosen

Step No.	**Step Detail**	**Expected Results**	**Actual Results**	**Status**
1	Navigate to Update Item Page	Page should be displayed	As Expected	Successful
2	Enter a valid Name, Price, and Quantity and click update item	Item updates successfully	As Expected	Successful
3	Enter an invalid Name (too long)	Display "Item name must be under 20 characters."	As Expected	Successful
4	Leave Name empty	Display "Item name cannot be empty."	As Expected	Successful
5	Enter an invalid Price (-50 or 100000001)	Display "Price must be a positive number up to 1 crore."	As Expected	Successful
6	Enter an invalid Quantity (-1 or 100000001)	Display "Quantity must be a positive number up to 1 crore."	As Expected	Successful
7	Leave Description empty or exceed 250 characters	Display "Description must be between 1-250 characters."	As Expected	Successful
8	Leave Fabric, Size, or Color unselected	Display "Please fill all required fields and make selections."	As Expected	Successful
9	Try updating an item with a duplicate name in the same category	Display "An item with this name already exists in the selected category."	As Expected	Successful

1.1.10 Delete Item Unit Testing

The delete item unit test in Table 1.10 checks that an item can be removed successfully from a category. It ensures associated files are deleted, prevents deletion of non-existent items, verifies proper error handling for failed deletions, and confirms the UI updates accordingly.

TABLE 1.10: Unit Test Case:Delete Item

Test Case ID	TC_010			
Test Case Description	Verify that an admin can delete an item successfully and handle invalid scenarios properly.			
Scenario	Ensure that selected items are removed correctly, unselected items trigger an error message, and cancellation preserves the item.			
Prerequisites	At least one item must exist.			
Test Data	ValidItemSelected = "Shirt"			
Step No.	**Step Detail**	**Expected Results**	**Actual Results**	**Status**
1	Navigate to Item List Page	Page should be opened	As Expected	Successful
2	Select an item and click Delete	Item deleted successfully	As Expected	Successful
3	Click Delete without selecting an item	Display "Please select an item first."	As Expected	Successful
4	Click Delete and cancel the confirmation dialog	Item remains in the list	As Expected	Successful

1.1.11 Search Item Unit Testing

The search item unit test in Table 1.11 checks the functionality of item lookup based on user input. It verifies that valid search queries return correct results and ensures no results are displayed for non-matching queries. Additionally, it ensures that searches are case-insensitive.

TABLE 1.11: Unit Test Case:Search Item

Test Case ID	TC_011
Test Case Description	Verify the item search functionality for admin users.
Scenario	Ensure admin can search for items successfully, and appropriate messages are shown when no matching items are found.
Prerequisites	At least one item should exist.
Test Data	ValidQuery = "Shirts" InvalidQuery = "Accessories" (not present) CaseInsensitiveQuery = "shirts" (should return same results as "Shirts")

Step No.	**Step Detail**	**Expected Results**	**Actual Results**	**Status**
1	Navigate to Item Management Page	Page should be opened	As Expected	Successful
2	Enter a valid item name in the search bar	Returns items matching "Shirts"	As Expected	Successful
3	Enter a non-existing item name in the search bar	No item should be found.	As Expected	Successful
4	Enter an empty search query	Resets to all fetched items.	As Expected	Successful
5	Enter an item name in a different case (e.g., "shirts")	Returns same results as "Shirts" (case-insensitive search)	As Expected	Successful

1.1.12 Accept Terms Unit Testing

The accept terms unit test in Table 1.12 ensures that users can agree to the application's terms and conditions before proceeding. It verifies the proper handling of user consent and prevents access if terms are declined.

Table 1.12: Unit Test Case:User Accept Terms

Test Case ID	TC_012
Test Case Description	Verify that users must accept the terms and conditions before proceeding.
Scenario	Ensure that users can accept or decline terms, and prevent access if terms are not accepted.
Prerequisites	User must be on the terms and conditions screen.
Test Data	userAcceptTerms = true userAcceptTerms = false Accept Terms button

Step No.	**Step Detail**	**Expected Results**	**Actual Results**	**Status**
1	Navigate to Accept Terms page	Terms screen should be displayed	As Expected	Successful
2	Accept Terms (userAcceptTerms = true)	User proceeds to the next screen	As Expected	Successful
3	Decline Terms (userAcceptTerms = false)	User remains on the terms screen	As Expected	Successful

1.1.13 User Select Categories Unit Testing

The select categories unit test in Table 1.13 verifies that users can choose a category to explore clothing items. It ensures correct category highlighting and prevents selection if no categories exist.

Table 1.13: Unit Test Case:User Select Categories

Test Case ID	TC_013
Test Case Description	Verify that users can select a clothing category.
Scenario	Ensure correct category highlighting and prevent selection if no categories exist.
Prerequisites	Categories must be loaded and visible.

Test Data		categoryList = [“Shirts”, “Pants”, “Jackets”] selectedCategory = “Pants”		
Step No.	**Step Detail**	**Expected Results**	**Actual Results**	**Status**
1	Navigate to Categories page	Categories screen should be displayed	As Expected	Successful
2	Select a category (selectedCategory = “Pants”)	“Pants” category is highlighted	As Expected	Successful
3	Try selecting a category when no categories are loaded	Prevent selection and display “No categories available.”	As Expected	Successful

1.1.14 Try-On Clothing Unit Testing

The try-on clothing unit test in Table 1.14 verifies that users can view themselves wearing 3D clothing models. It checks for the proper overlaying of clothing on the user’s body.

Table 1.14: Unit Test Case:Try-On Clothing

Test Case ID	TC_014
Test Case Description	Verify that the selected 3D clothing model correctly overlays on the user’s body.
Scenario	Ensure the clothing model aligns with the user’s body position and handles cases where the user is not detected.
Prerequisites	1. Kinect sensor must be connected and tracking the user. 2. At least one item must be selected for try-on.

Test Data	selectedItem = “Jacket”, userPosition = detected userPosition = null			
Step No.	**Step Detail**	**Expected Results**	**Actual Results**	**Status**
1	Select a clothing item and stand in front of the Kinect sensor	Clothing overlays correctly on the user’s body	As Expected	Successful
2	Attempt try-on with no user detected	Displays “User not detected”	As Expected	Successful
3	Move around to check if the clothing follows the user’s body	Clothing moves correctly with the user	As Expected	Successful
4	Try-on with Kinect disconnected	Displays error “Kinect not connected”	As Expected	Successful

1.1.15 Change Background Unit Testing

The change background unit test in Table 1.15 ensures that users can switch between different backgrounds. It verifies that the selected backgrounds apply correctly.

Table 1.15: Unit Test Case:Change Background

Test Case ID	TC_015
Test Case Description	Verify that users can change the background to a selected option.
Scenario	Ensure background changes correctly when selected and handles cases where no backgrounds are available.
Prerequisites	Kinect sensor must be connected and capturing the user.
Test Data	selectedBg = “Beach”

Step No.	Step Detail	Expected Results	Actual Results	Status
1	Select a background from the available options	Background changes to the selected option	As Expected	Successful
2	Try switching backgrounds rapidly	Background updates smoothly without lag	As Expected	Successful
3	Change background with Kinect disconnected	Displays error "Kinect not connected"	As Expected	Successful

1.1.16 Save Picture Unit Testing

The save picture unit test in Table 1.16 ensures that users can capture and store images of their virtual try-on session. It verifies successful image saving and proper error handling.

Table 1.16: Unit Test Case:Save Picture

Test Case ID	TC_016
Test Case Description	Verify that users can successfully capture and save images of their try-on session.
Scenario	Ensure that the captured image is stored correctly and appropriate error messages are displayed when saving fails.
Prerequisites	Try-On session must be active. Camera feed must be available to capture the picture.
Test Data	imageCaptured = true imageCaptured = false

Step No.	Step Detail	Expected Results	Actual Results	Status
1	Capture an image while the try-on session is active	Image is saved to gallery	As Expected	Successful
2	Attempt to save an image when capture fails	Displays "Failed to save image"	As Expected	Successful
3	Save image with insufficient storage	Displays "Storage full. Unable to save image"	As Expected	Successful
4	Save image when Kinect camera is disconnected	Displays "Camera not available"	As Expected	Successful

1.2 Integration Testing

After each unit is thoroughly tested, it is integrated with other units to create modules or components that are designed to perform specific tasks or activities. These are then tested as group through integration testing to ensure whole segments of an application behave as expected (i.e, the interactions between units are seamless). These tests are often framed by user scenarios, such as logging into an application or opening files. Integrated tests can be conducted by either developers or independent testers and are usually comprised of a combination of automated functional and manual tests.

1.2.1 Login Integration Testing

The admin login integration tests in Table 1.17 ensure that the admin login process functions correctly, including handling both successful and unsuccessful login attempts.

Table 1.17: Admin Login Integration Test Case

Test Case ID	TCI_001
Test Case Description	Verify the integration of the admin login functionality.

Scenario	Ensure the admin can log in using valid credentials and handle invalid login attempts appropriately.
Prerequisites	Admin API should be running, and the system should be configured for admin login.
Test Data	ValidAdminCredentials = (Admin@123, rightemail@gmail.com) InvalidAdminCredentials = (wrong@123, wrongemail@gmail.com)

Step No.	**Step Detail**	**Expected Results**	**Actual Results**	**Status**
1	Admin logs in with correct credentials	API validates credentials and grants access.	As Expected	Successful
2	Admin enters incorrect credentials	API rejects login and displays: "Invalid username or password."	As Expected	Successful

1.2.2 Signup Integration Testing

The admin signup integration tests in Table 1.18 ensure that the admin registration process functions correctly, handling both successful and unsuccessful registration attempts.

TABLE 1.18: Integration Test Case:Admin Signup

Test Case ID	TCI_002
Test Case Description	Verify the integration of the admin signup functionality.
Scenario	Ensure the admin can register using valid credentials and handle invalid signup attempts appropriately.
Prerequisites	Admin API should be running, and the system should be configured for admin registration.

Test Data	ValidAdminCredentials = (Admin, rightemail@gmail.com, Admin@123, Admin@123) InvalidAdminCredentials = (Admin, Admin@123, Adm123, wrongemail@gmail.com)			
Step No.	**Step Detail**	**Expected Results**	**Actual Results**	**Status**
1	Admin registers with valid credentials	API stores admin credentials and returns success response.	As Expected	Successful
2	Admin registers with invalid credentials	API rejects registration and displays: "Invalid entries."	As Expected	Successful

1.2.3 Update Profile Integration Testing

The admin update profile integration tests in Table 1.19 ensure that the admin can update their profile details and handle errors such as unauthorized access or invalid input.

Table 1.19: Integration Test Case:Admin Update Profile

Test Case ID	TCI_003
Test Case Description	Verify the integration of the admin update profile functionality.
Scenario	Ensure the admin can update their profile and handle invalid session, unauthorized access, or invalid data appropriately.
Prerequisites	Admin profile should exist in the system and should be able to update their details.
Test Data	ValidAdminUpdateData = (UpdatedAdmin@123, updatedemail@domain.com) InvalidAdminUpdateData = (wrongemail@domain.com)

Step No.	Step Detail	Expected Results	Actual Results	Status
1	Admin updates their profile with valid data	API returns success response, and profile details are updated.	As Expected	Successful
2	Admin attempts to update profile with invalid data (e.g., incorrect email format)	Display: "Invalid data. Please correct the inputs."	As Expected	Successful
3	Admin attempts to update profile with no credentials	Display: "Authorization required."	As Expected	Successful

1.2.4 Add Category Integration Testing

The add category integration tests in Table 1.20 ensure that the admin can successfully add new categories, and handle failure scenarios such as invalid data.

Table 1.20: Integration Test Case:Add Category

Test Case ID	TCI_004
Test Case Description	Verify the integration of the add category functionality.
Scenario	Ensure the admin can successfully add a new category and handle invalid input scenarios.
Prerequisites	Admin must be logged in.
Test Data	ValidCategoryData = (CategoryName: "Pants", CategoryImage: "model.fbx") InvalidCategoryData = (CategoryName: "", CategoryImage: "")

Step No.	Step Detail	Expected Results	Actual Results	Status
1	Admin adds a new category with valid data	API stores category in the database and returns a success response.	As Expected	Successful
2	Admin adds a new category with invalid data	System displays: "Failed to add category. Check connection."	As Expected	Successful

1.2.5 Update Category Integration Testing

The update category integration tests in Table 1.21 ensure that the admin can successfully update existing categories, and handle invalid data or API failures.

TABLE 1.21: Integration Test Case:Update Category

Test Case ID	TCI_005
Test Case Description	Verify the integration of the update category functionality.
Scenario	Ensure the admin can successfully update a category and handle failure scenarios appropriately.
Prerequisites	Admin must be logged in and category must exist for update.
Test Data	ValidCategoryUpdateData = (CategoryName: "Pants", CategoryDescription: "Pinkpant") InvalidCategoryUpdateData = (CategoryName: "", CategoryDescription: "")

Step No.	Step Detail	Expected Results	Actual Results	Status
1	Admin updates an existing category with valid data	API updates category details and returns success response.	As Expected	Successful
2	Admin updates an existing category with invalid data	System displays: "Failed to update category."	As Expected	Successful

1.2.6 Delete Category Integration Testing

The delete category integration tests in Table 1.22 ensure that the admin can successfully delete existing categories, and handle any failure cases like invalid category deletion attempts.

Table 1.22: Integration Test Case:Delete Category

Test Case ID	TCI_006
Test Case Description	Verify the integration of the delete category functionality.
Scenario	Ensure the admin can delete an existing category and handle failures such as missing categories or server issues.
Prerequisites	Admin must be logged in and category must exist for deletion.
Test Data	ValidCategoryDeleteData = (CategoryName: "Pants") InvalidCategoryDeleteData = (CategoryName: "NonExistentCategory")

Step No.	Step Detail	Expected Results	Actual Results	Status
1	Admin deletes an existing category	API removes category and UI updates accordingly.	As Expected	Successful
2	Admin attempts to delete a non-existing category	System displays: "Category deletion failed."	As Expected	Successful

1.2.7 Search Category Integration Testing

The search category integration tests in Table 1.23 ensure that the admin can successfully search for categories and handle failure scenarios like empty or incorrect search results.

TABLE 1.23: Integration Test Case:Search Category

Test Case ID	TCI_007
Test Case Description	Verify the integration of the search category functionality.
Scenario	Ensure the admin can search categories by name and handle failure cases like no results or invalid search input.
Prerequisites	Admin must be logged in and categories must exist in the system.
Test Data	ValidCategorySearchData = "Pants" InvalidCategorySearchData = "NonExistentCategory"

Step No.	Step Detail	Expected Results	Actual Results	Status
1	Admin searches for a valid category	API returns the searched category and its details.	As Expected	Successful
2	Admin searches for a non-existent category	System displays: "No categories found."	As Expected	Successful

1.2.8 Item Add Integration Testing

The item add integration tests in Table 1.24 ensure that the admin can successfully add an item to a selected category, handling both successful and unsuccessful attempts appropriately.

Table 1.24: Integration Test Case:Item Add

Test Case ID	TCI_008
Test Case Description	Verify the integration of the item addition functionality within a selected category.
Scenario	Ensure the admin can successfully add an item to a selected category and handle failure cases appropriately.
Prerequisites	Admin must be logged in, an existing category should be selected, and the API should be running and accessible.
Test Data	ValidItemData = (ItemName: "pinkpant", ItemDescription: "Latest", Category: "Pants") InvalidItemData = (ItemName: "", ItemDescription: "", Category: "")

Step No.	Step Detail	Expected Results	Actual Results	Status
1	Admin adds a new item to the selected category	API stores the item in the db under the selected category and returns a success response.	As Expected	Successful
2	Admin adds a new item to the selected category with validdata	API returns a "invalid data" response.	As Expected	Successful

1.2.9 Item Update Integration Testing

The item update integration tests in Table 1.25 ensure that the admin can successfully update the details of an existing item in a selected category, while handling both successful and unsuccessful attempts.

Table 1.25: Integration Test Case:Item Update

Test Case ID	TCI_009
Test Case Description	Verify the integration of the item update functionality within a selected category.
Scenario	Ensure the admin can successfully update an existing item and handle failure cases appropriately.
Prerequisites	Admin must be logged in, an existing item should be available in the selected category, and the API should be running and accessible.
Test Data	ValidItemData = (ItemName: "PinkPant", ItemDescription: "Latest", Category: "Pant") InvalidItemData = (ItemName: "", ItemDescription: "", Category: "")

Step No.	Step Detail	Expected Results	Actual Results	Status
1	Admin updates an existing item in the selected category	API updates item details and returns a success response.	As Expected	Successful
2	Admin updates an existing item in the selected category with invalid data	API returns "invalid data can't be edited".	As Expected	Successful

1.2.10 Item Delete Integration Testing

The item delete integration tests in Table 1.26 ensure that the admin can successfully delete an existing item from a selected category, while handling both successful and unsuccessful attempts.

Table 1.26: Integration Test Case:Item Delete

Test Case ID	TCI_010
Test Case Description	Verify the integration of the item deletion functionality within a selected category.
Scenario	Ensure the admin can successfully delete an existing item from the selected category and handle failure cases appropriately.
Prerequisites	Admin must be logged in, an existing item should be available in the selected category, and the API should be running and accessible.
Test Data	ValidItemData = (ItemName: "pinkPant", ItemDescription: "Latest", Category: "Pant") InvalidItemData = (ItemName: "", ItemDescription: "", Category: "")

Step No.	Step Detail	Expected Results	Actual Results	Status
1	Admin deletes an existing item from the selected category	API removes the item from the database and updates the UI accordingly.	As Expected	Successful
1	Admin attempts to delete an existing item from the selected category	API displays "item does not exist".	As Expected	Successful

1.2.11 Item Search Integration Testing

The item search integration tests in Table 1.27 ensure that the admin can successfully search for an item within a selected category, handling both successful and unsuccessful search results appropriately.

TABLE 1.27: Integration Test Case:Item Search

Test Case ID	TCI_011
Test Case Description	Verify the integration of the item search functionality within a selected category.
Scenario	Ensure the admin can successfully search for an item within a selected category and handle failure cases appropriately.
Prerequisites	Admin must be logged in, an existing category with items should be selected, and the API should be running and accessible.
Test Data	ValidItemData = (ItemName: "PinkPant", Category: "Pant") InvalidItemData = (ItemName: "NonExistentItem", Category: "Pant")

Step No.	Step Detail	Expected Results	Actual Results	Status
1	Admin searches for an item in the selected category	API returns the searched item and its details.	As Expected	Successful
2	Admin searches for a non-existing item in the selected category	API returns "no item found".	As Expected	Successful

1.2.12 Unity-API Integration Testing

The Unity-API integration tests in Table 1.28 ensure that the Unity application correctly communicates with the API to fetch categories, items, and handle failure cases appropriately.

TABLE 1.28: Unity-API Category Integration Test Case (User-Side)

Test Case ID	TCI_012
Test Case Description	Verify the integration of Unity application with the API for fetching categories, fetching items under a selected category, and handling errors appropriately.
Scenario	Ensure the Unity application successfully fetches categories, items, and handles API call failures gracefully.
Prerequisites	Unity application should be running and connected to the API, the database should have at least one category and one item, and the API should return valid responses when called.
Test Data	ValidCategory = (Clothing) ValidItem = (Pinkpant) InvalidCategory = (NonExistentCategory)

Step No.	Step Detail	Expected Results	Actual Results	Status
1	Unity fetches categories from API	API returns available categories	As Expected	Successful
2	Unity fetches items under a selected category	API returns correct items under the selected category.	As Expected	Successful

1.2.13 Kinect Integration Testing

The Kinect integration tests in Table 1.29 ensure that the Kinect sensor properly interacts with Unity, handling both successful and unsuccessful gesture recognition appropriately.

TABLE 1.29: Integration Test Case:Kinect

Test Case ID	TCI_013
Test Case Description	Verify the integration of Kinect gesture detection with Unity.
Scenario	Ensure the Kinect sensor detects user gestures and Unity updates the UI accordingly, and handle gesture recognition failures appropriately.
Prerequisites	Kinect sensor should be properly connected and calibrated, the user should be positioned within Kinect's tracking range, and Unity should be able to process gesture inputs.
Test Data	ValidGesture = "Swipe Left" InvalidGesture = "Unrecognized Gesture"

Step No.	Step Detail	Expected Results	Actual Results	Status
1	User navigates using hand gestures	Kinect detects movement, Unity UI updates.	As Expected	Successful
2	User tries to navigate with invalid gestures.	Display: "Gesture not recognized. Try again."	As Expected	Successful
3	User selects an item using hand gestures	Kinect registers selection, Unity loads item details.	As Expected	Successful
4	User tries to navigate when kinect is disconnected	Display: "kinect Disconnected. Try Again."	As Expected	Successful

1.2.14 3D Try-On Integration Testing

The 3D Try-On integration tests in Table 1.30 ensure that the Unity application correctly retrieves and applies the 3D clothing model on the user's avatar when Kinect detects the user.

Table 1.30: Integration Test Case:3D Try-On

Test Case ID	TCI_014
Test Case Description	Verify the integration of 3D try-on functionality.
Scenario	Ensure that when a user selects a clothing item, it is correctly applied to their avatar, and the Kinect sensor detects the user.
Prerequisites	Kinect sensor must successfully detect the user, '.fbx' model files should exist in the correct directory, and API should return a valid 3D model path.

Test Data		Valid3DModelData = (ClothingItem: "Shirt", ModelPath: "/models/shirt.fbx")		
Step No.	**Step Detail**	**Expected Results**	**Actual Results**	**Status**
1	User selects a 3D clothing item from the catalog	Unity retrieves the '.fbx' model for the item and applies it to the user's avatar.	As Expected	Successful
2	Kinect captures the user	Clothing appears correctly on the user's avatar.	As Expected	Successful

1.2.15 Save Image Integration Testing

The Save Feature integration tests in Table 1.31 ensure that the Unity application successfully stores an image of the user wearing the selected clothing.

Table 1.31: Integration Test Case:Save Feature

Test Case ID	TCI_015
Test Case Description	Verify the integration of the image saving functionality.
Scenario	Ensure that the user can save the image of their avatar wearing the selected clothing item, and handle failure cases appropriately.
Prerequisites	Storage permissions must be enabled for saving images, and Kinect must have successfully detected the user and applied the clothing.
Test Data	ValidSaveData = (ClothingItem: "Shirt", ImagePath: "/savedimages/userimage.png")

Step No.	Step Detail	Expected Results	Actual Results	Status
1	User saves the image of their avatar wearing the clothing item	System successfully stores the image of the user wearing the selected clothing.	As Expected	Successful
2	User saves the image of their avatar wearing the clothing item when memory is full.	System displays: "Memory is Full."	As Expected	Successful

1.3 System Testing

System testing is a black box testing method used to evaluate the completed and integrated system as a whole, ensuring it meets the specified requirements. The functionality of the software is tested from end to end, including API communication, Kinect tracking, and Unity-based rendering.

1.3.1 Admin System Testing

The system test in Table 1.32 for admin system evaluates the smooth flow of the operations performed in admin-side application, ensuring seamless interaction between the UI, API, and database.

Table 1.32: System Test Case:AdminSide

Test Case ID	ST_001
Test Case Description	Verify that the admin app functions correctly, ensuring data storage, and API interaction.
Scenario	Test whether the system correctly handles the adminside operations.
Prerequisites	Admin API must be running. Database should be accessible.

Test Data	Admin = (Username: Admin1, Email: admin1@example.com, Password: Admin@123) Category = (CategoryName: “Clothing”) ValidItem = (“ItemName: T-Shirt, Price: 1000, Category: Clothing”)

Step No.	**Step Detail**	**Expected Results**	**Actual Results**	**Status**
1	Admin enters valid details and submits signup request	API validates data, stores it in the database, and returns a success response	As Expected	Successful
2	Admin enters valid credentials and clicks "Login"	System verifies credentials via API and grants access	As Expected	Successful
3	Admin enters a valid category name and submits the form	API stores the category in the database and returns success response	As Expected	Successful
4	Admin selects a category and updates it with valid data, then submits the update	API updates the category in the database and returns success response	As Expected	Successful
5	Admin selects a category and confirms deletion	API removes the category from the database and returns success response	As Expected	Successful
6	Admin enters a valid category name in the search bar	System filters and displays the matching category from the database	As Expected	Successful
7	Admin selects a category from the category list and add valid item details	API stores the item within respected category and returns success response.	As Expected	Successful
8	Admin selects an item and updates it with valid data, then submits the	API updates the item within category in the database and returns success	As Expected	Successful

1.3.2 UserSide System Testing

The system test case in Table 1.33 for user system evaluates the smooth flow of the operations performed in user-side application, ensuring kinect integration, interaction between the UI, API, and database.

Table 1.33: System Test Case:UserSide

Test Case ID	ST_002
Test Case Description	Verify that users can interact with kinect, virtually tryon the clothes, save pictures and change background.
Scenario	Test whether the system correctly handles the userside operations.
Prerequisites	System must be running and Kinect is properly connected. Admin API must be running. Database should be accessible.
Test Data	IsAcceptTerms=True Category=Frock Cloth=PinkFrock

Step No.	**Step Detail**	**Expected Results**	**Actual Results**	**Status**
1	User reads and accepts terms and conditions via hand gesture.	System stores acceptance status and category screen is displayed.	As Expected	Successful
2	User selects a category via hand gesture.	System displays respective items within category.	As Expected	Successful
3	User selects a clothing item via hand gesture and stands in front of the Kinect sensor	The selected clothing overlays correctly on the user's body along with respective cloth details displayed.	As Expected	Successful
4	User selects a background from the available options	Background should change smoothly to the selected background without delay or glitches	As Expected	Successful
5	User clicks the capture button while the try-on session is active	Image is successfully captured and saved to the gallery with a confirmation message	As Expected	Successful

1.4 Acceptance Testing

It is formal testing according to user needs, requirements, and business processes conducted to determine whether a system satisfies the acceptance criteria or not and to enable the users, customers, or other authorized entities to determine whether to accept the system or not. It is the last phase of software testing

performed after System Testing and before making the system available for actual use. [2]

1.4.1 Signup Acceptance Testing

The admin signup acceptance test in Table 1.34 ensures that the signup functionality meets business and user requirements. It verifies whether an admin can register successfully and whether the system handles invalid inputs properly.

TABLE 1.34: Acceptance Test Case:Admin Signup

Test Case ID	ATC_001
Test Case Description	Validate that the admin signup process aligns with business requirements.
Scenario	Ensure the system correctly registers an admin and provides appropriate error handling for invalid signup attempts.
Prerequisites	The Admin API should be running. The system should be configured for admin registration.
Test Data	ValidAdminCredentials = (Admin@123, rightemail@gmail.com) InvalidAdminCredentials = (wrong@123, wrongemail@gmail.com)

Step No.	Step Detail	Expected Results	Actual Results	Status
1	Navigate to the Signup Page	Signup screen should be displayed	As Expected	Successful
2	Enter valid admin credentials	Admin account is created, and a success message is shown	As Expected	Successful
3	Enter invalid credentials (wrong password or email)	Displays error message: "Invalid credentials. Try again."	As Expected	Successful
4	Try signing up with an already registered email	Displays error: "Email already in use"	As Expected	Successful
5	Submit empty form	Displays validation errors for missing fields	As Expected	Successful
6	Complete successful signup and attempt login	Admin can log in successfully with registered credentials	As Expected	Successful

1.4.2 Login Acceptance Testing

The admin login acceptance test in Table 1.35 ensures that the login functionality meets business and user requirements. It verifies whether an admin can log in successfully and whether the system handles invalid login attempts properly.

Table 1.35: Acceptance Test Case:Admin Login

Test Case ID	ATC_002
Test Case Description	Validate that the admin login process aligns with business requirements.
Scenario	Ensure the system correctly authenticates an admin and provides appropriate error handling for invalid login attempts.

Prerequisites	The Admin API should be running. An admin account must be registered in the system.
Test Data	ValidAdminCredentials = (Admin@123, rightemail@gmail.com) InvalidAdminCredentials = (wrong@123, wrongemail@gmail.com)

Step No.	**Step Detail**	**Expected Results**	**Actual Results**	**Status**
1	Navigate to the Login Page	Login screen should be displayed	As Expected	Successful
2	Enter valid admin credentials	Admin is logged in, and the dashboard is displayed	As Expected	Successful
3	Enter incorrect credentials (wrong password or email)	Displays error message: "Invalid credentials. Try again."	As Expected	Successful
4	Attempt login with an unregistered email	Displays error: "Account not found"	As Expected	Successful
5	Submit an empty form	Displays validation errors for missing fields	As Expected	Successful

1.4.3 Add Category Acceptance Testing

The admin add category acceptance test in Table 1.36 ensures that the category creation functionality meets business and user requirements. It verifies whether an admin can successfully add a category with a name and an image while handling errors for invalid inputs.

Table 1.36: Acceptance Test Case:Admin Add Category

Test Case ID	ATC_003

Test Case Description	Validate that the admin can successfully add a category with a name and an image.
Scenario	Ensure that a new category is added correctly and proper error messages are displayed for invalid inputs.
Prerequisites	The Admin API should be running. Admin should be logged into the system.
Test Data	ValidCategory = ("Men's Wear", "menwear.jpg") InvalidImage = ("Men's Wear", "invalidformat.txt")

Step No.	**Step Detail**	**Expected Results**	**Actual Results**	**Status**
1	Navigate to the Add Category Page	Add category screen should be displayed	As Expected	Successful
2	Enter a valid category name and upload a valid image	Category is added successfully	As Expected	Successful
3	Enter an empty category name and try to submit	Displays error message: "Category name is required."	As Expected	Successful
4	Upload an invalid image format (e.g., '.fbx file)	Displays error: "Invalid image format. Please upload a valid image."	As Expected	Successful
5	Attempt to add a category without uploading an image	Displays error: "Category image is required."	As Expected	Successful
6	Try adding a category with a name that already exists	Displays error: "Category already exists."	As Expected	Successful

1.4.4 Update Category Acceptance Testing

The admin update category acceptance test in Table 1.37 ensures that the category update functionality meets business and user requirements. It verifies whether an admin can successfully update a category's name and image while handling errors for invalid inputs.

TABLE 1.37: Acceptance Test Case:Admin Update Category

Test Case ID	ATC_004
Test Case Description	Validate that the admin can successfully update a category's name and image.
Scenario	Ensure that an existing category is updated correctly and proper error messages are displayed for invalid inputs.
Prerequisites	The Admin API should be running. Admin should be logged into the system. At least one category should exist.
Test Data	ValidUpdate = ("Men's Wear", "menwearupdated.jpg") InvalidImage = ("Men's Wear", "invalidformat.txt")

Step No.	Step Detail	Expected Results	Actual Results	Status
1	Navigate to the Category List	List of categories should be displayed	As Expected	Successful
2	Select an existing category and click update	Update category screen should be displayed	As Expected	Successful
3	Enter a valid category name and upload a valid image	Category is updated successfully	As Expected	Successful
4	Enter an empty category name and try to submit	Displays error message: "Category name is required."	As Expected	Successful
5	Upload an invalid image format (e.g., '.fbx' file)	Displays error: "Invalid image format. Please upload a valid image."	As Expected	Successful
6	Attempt to update a category without uploading an image	Displays error: "Category image is required."	As Expected	Successful
7	Try updating a category with a name that already exists	Displays error: "Category name already in use."	As Expected	Successful

1.4.5 Delete Category Acceptance Testing

The admin delete category acceptance test in Table 1.38 ensures that the category deletion functionality meets business and user requirements. It verifies whether an admin can successfully delete a category while handling errors for invalid or restricted deletions.

TABLE 1.38: Acceptance Test Case:Admin Delete Category

Test Case ID	ATC_005
Test Case Description	Validate that the admin can successfully delete an existing category.
Scenario	Ensure that an existing category can be deleted correctly and proper error messages are displayed for restricted or invalid deletions.
Prerequisites	The Admin API should be running. Admin should be logged into the system. At least one category should exist.
Test Data	ValidCategory = ("Men's Wear") CategoryWithProducts = ("Winter Collection")

Step No.	**Step Detail**	**Expected Results**	**Actual Results**	**Status**
1	Navigate to the Category List	List of categories should be displayed	As Expected	Successful
2	Select an existing category and click delete	A confirmation prompt should appear	As Expected	Successful
3	Confirm the deletion of a valid category	Category is deleted successfully	As Expected	Successful
4	Try deleting a category that contains products	Displays error: "Category and its items would also be deleted."	As Expected	Successful
5	Cancel the delete action when prompted	Category remains unchanged	As Expected	Successful

1.4.6 Add Item Acceptance Testing

The admin add item acceptance test in Table 1.39 ensures that the admin can successfully add a new item with a name, details, price, and an '.fbx' model. It

verifies that the item is stored correctly and proper validation messages appear when incorrect data is entered.

Table 1.39: Acceptance Test Case:Admin Add Item

Test Case ID	ATC_006
Test Case Description	Validate that the admin can add a new item with a name, details, price, and an '.fbx' model.
Scenario	Ensure the item is saved correctly in the system and validation errors appear for missing or invalid inputs.
Prerequisites	The Admin API should be running. Admin should be logged into the system. At least one category should exist.
Test Data	ValidItem = ("Winter Jacket", "Warm and stylish", 59.99, "jacket.fbx") InvalidItem = ("", "", -10, "invalid.txt")

Step No.	Step Detail	Expected Results	Actual Results	Status
1	Navigate to the Add Item page	Displays input fields for Name, Details, Price, and '.fbx' model upload	As Expected	Successful
2	Enter valid details and select a valid '.fbx' model	Item is successfully added to the database	As Expected	Successful
3	Leave fields empty and attempt to submit	Displays validation errors	As Expected	Successful
4	Enter negative price	Displays error: "Price must be positive"	As Expected	Successful
5	Upload a file that is not '.fbx' (e.g., 'invalid.txt')	Displays error: "Only '.fbx' files are allowed"	As Expected	Successful

1.4.7 Update Item Acceptance Testing

The admin update item acceptance test in Table 1.40 ensures that the admin can successfully update an existing item's name, details, and price without changing the '.fbx' model. It verifies that the changes are stored correctly and appropriate validation messages appear when incorrect data is entered.

Table 1.40: Acceptance Test Case:Admin Update Item

Test Case ID	ATC_007
Test Case Description	Validate that the admin can update an item's name, details, and price without modifying the '.fbx' model.
Scenario	Ensure the item updates correctly in the system and validation errors appear for missing or invalid inputs.

Prerequisites	The Admin API should be running. Admin should be logged into the system. At least one item should exist in the database.
Test Data	ValidItemUpdate = (“Updated Jacket”, “Stylish and comfortable”, 69.99) InvalidItemUpdate = (“”, “”, -20)

Step No.	**Step Detail**	**Expected Results**	**Actual Results**	**Status**
1	Navigate to the Update Item page	Displays input fields for Name, Details, and Price	As Expected	Successful
2	Modify item details and save changes	Item updates successfully in the database	As Expected	Successful
3	Leave all fields empty and attempt to update	Displays validation errors: “Name is required”, “Details required”, “Invalid price”	As Expected	Successful
4	Enter negative price	Displays error: “Price must be positive”	As Expected	Successful
5	Try updating an item when API is down	Displays error: “Unable to update item. Try again later.”	As Expected	Successful

1.4.8 Delete Item Acceptance Testing

The admin delete item acceptance test in Table 1.41 ensures that the admin can successfully remove an item from the system. It verifies that the item is deleted correctly and appropriate validation messages appear when deletion fails.

Table 1.41: Acceptance Test Case:Admin Delete Item

Test Case ID	ATC_008
Test Case Description	Validate that the admin can delete an item from the system.
Scenario	Ensure the item is removed from the database, and appropriate errors appear when deletion is not possible.
Prerequisites	The Admin API should be running. Admin should be logged into the system. At least one item should exist in the database.
Test Data	ValidItemID = 001 InvalidItemID = 999 (Non-existent)

Step No.	**Step Detail**	**Expected Results**	**Actual Results**	**Status**
1	Navigate to the Item List page	Displays a list of items with delete buttons	As Expected	Successful
2	Click the delete button for an existing item	Item is removed from the database	As Expected	Successful
3	Attempt to delete an item that does not exist	Displays error: "Item not found"	As Expected	Successful
4	Try deleting an item when the API is down	Displays error: "Unable to delete item. Try again later."	As Expected	Successful
5	Refresh the item list after deletion	Deleted item no longer appears in the list	As Expected	Successful

1.4.9 User Accept Terms and Conditions Acceptance Testing

The user accept terms acceptance test in Table 1.42 ensures that users can successfully read and accept the terms and conditions before accessing the AR-based

system. It verifies that Kinect interaction works properly for accepting terms.

Table 1.42: Acceptance Test Case:User Accept Terms and Conditions

Test Case ID	ATC_009
Test Case Description	Validate that users can accept terms and conditions before proceeding with the AR system.
Scenario	Ensure that users can locate, understand, and accept the terms before accessing the system, with proper error handling.
Prerequisites	The terms and conditions screen should be accessible. Kinect sensor should be connected and tracking the user. Users must accept the terms before proceeding.
Test Data	userAcceptTerms = true userAcceptTerms = false

Step No.	Step Detail	Expected Results	Actual Results	Status
1	Kinect detects user presence	User is identified, and terms screen is displayed	As Expected	Successful
2	User reads the terms and conditions	Text is fully visible, and scrolling works correctly	As Expected	Successful
3	User accepts terms using hand gesture (or selecting checkbox via Kinect)	Acceptance is registered, and user is allowed to proceed	As Expected	Successful
4	User attempts to proceed without accepting terms (userAcceptTerms = false)	System prevents access and displays "You must accept the terms to continue."	As Expected	Successful

1.4.10 User Select Categories Acceptance Testing

The select categories acceptance test in Table 1.43 ensures that users can successfully identify and select a category to explore clothing items. It verifies the clarity of category selection, responsiveness, and system behavior when no categories are available.

Table 1.43: Acceptance Test Case:User Select Categories

Test Case ID	ATC_010
Test Case Description	Validate that users can select a clothing category and navigate to the relevant items page.
Scenario	Ensure that users can select a category, recognize highlighted selections, and receive appropriate feedback when no categories exist.

Prerequisites	The categories page must be accessible. Categories should be preloaded in the system.
Test Data	categoryList = [“Shirts”, “Pants”, “Jackets”] selectedCategory = “Pants”

Step No.	**Step Detail**	**Expected Results**	**Actual Results**	**Status**
1	User navigates to the categories page	Categories should be clearly displayed	As Expected	Successful
2	User selects a category and navigates to the items page	Correct category’s items should be displayed	As Expected	Successful
3	User tries to click update/delete buttons without selecting category.	Selection is disabled, and feedback is provided	As Expected	Successful

1.4.11 User Select Items Acceptance Testing

The acceptance test in Table **??** for selecting items ensures that users can intuitively choose a clothing item from the available list. It verifies the ease of item selection, the clarity of feedback upon selection, and the system’s response when no items are available.

1.4.12 Try-On Clothing Acceptance Testing

The acceptance test in Table 1.44 for the try-on clothing feature ensures that users can accurately interact with the virtual try-on system. It verifies the correctness of clothing alignment, system responsiveness, and user satisfaction.

Table 1.44: Acceptance Test Case:Try-On Clothing

Test Case ID	ATC_011

Test Case Description	Validate that users can successfully try on clothing virtually, ensuring proper overlay, responsiveness, and seamless interaction.
Scenario	Test how accurately the system overlays clothing on users and ensures smooth tracking and usability under different conditions.
Prerequisites	Kinect sensor must be connected and tracking the user. At least one clothing item must be selected for try-on.
Test Data	validUser = (User successfully tries on clothing with proper alignment) invalidUser = (User faces difficulties in overlay accuracy or responsiveness) kinectStatus = (Connected, Disconnected)

Step No.	**Step Detail**	**Expected Results**	**Actual Results**	**Status**
1	User selects a clothing item and stands in front of the Kinect sensor	Selected clothing overlays correctly on the user's body	As Expected	Successful
2	User moves around while wearing the virtual clothing	Clothing follows the user's body movements accurately	As Expected	Successful
3	User attempts try-on without standing in the Kinect's view	Displays "User not detected" message	As Expected	Successful
4	User tries different clothing items rapidly	Application responds without lag or glitches	As Expected	Successful
5	User attempts try-on with Kinect disconnected	Displays error "Kinect not connected"	As Expected	Successful

1.4.13 Change Background Acceptance Testing

The acceptance test in Table 1.45 for the change background feature ensures that users can seamlessly switch between different backgrounds, verifying transition smoothness and system responsiveness.

TABLE 1.45: Acceptance Test Case:Change Background

Test Case ID	ATC_012
Test Case Description	Validate that users can successfully change backgrounds, ensuring smooth transitions and an intuitive selection process.
Scenario	Test how efficiently users can switch between backgrounds and evaluate transition quality, performance, and error handling.
Prerequisites	Kinect sensor must be connected and tracking the user. At least one background must be available for selection.
Test Data	validBackground = ("FittingRoom1", "FittingRoom2", "FittingRoom3") kinectStatus = (Connected, Disconnected)

Step No.	**Step Detail**	**Expected Results**	**Actual Results**	**Status**
1	User selects a background from the available options	Background changes smoothly to the selected option	As Expected	Successful
2	User rapidly switches between different backgrounds	Background transitions occur without noticeable lag or flickering	As Expected	Successful
3	User tries changing backgrounds with Kinect disconnected	Displays error "Kinect not connected"	As Expected	Successful

1.4.14 Save Picture Acceptance Testing

The acceptance test in Table 1.46 for the save picture feature ensures that users can successfully capture and store images of their virtual try-on session, verifying the system's responsiveness, reliability, and error handling.

TABLE 1.46: Acceptance Test Case:Save Picture

Test Case ID	ATC_013
Test Case Description	Validate that users can capture and save images from their virtual try-on session, ensuring proper system feedback and error handling.
Scenario	Test how efficiently users can capture and save images, ensuring seamless system response and proper error messages in case of failure.
Prerequisites	Try-on session must be active. Camera feed must be available. Adequate storage space must be available.
Test Data	validCapture = (Image captured successfully) invalidCapture = (Capture failed, storage full, camera unavailable)

Step No.	Step Detail	Expected Results	Actual Results	Status
1	User clicks the capture button while try-on session is active	Image is saved to gallery with a confirmation message	As Expected	Successful
2	User attempts to save an image when the capture fails	Displays "Failed to save image" with a retry option	As Expected	Successful
3	User tries saving an image with insufficient storage	Displays "Storage full. Unable to save image"	As Expected	Successful
4	User attempts to save an image when Kinect camera is disconnected	Displays "Camera not available" with troubleshooting guide	As Expected	Successful

1.5 Performance Testing

Performance testing is a non-functional testing technique used to determine how an application will behave under various conditions. The goal is to test its responsiveness and stability in real user situations. Performance testing can be broken down into four types:

- Load testing
- Stress testing
- Endurance testing
- Spike testing

1.5.1 Load Testing

Load testing evaluates the system's performance under varying loads, ensuring that the system can handle a significant number of requests and interactions.

This section contains load test cases for both the admin and user sides.
The load test in Table 1.47 verifies system's ability to handle a varying number of requests through the API under load conditions.

TABLE 1.47: Load Test Case

Test Case ID	LT_001
Test Case Description	Verify the system's ability to handle a varying number of requests through the API under load conditions.
Scenario	Test the API's performance when simulating 75 requests simultaneously to determine how well the system handles requested data under load.
Prerequisites	Admin API must be running. API should be able to handle requests. Ensure database is prepared for load testing (can handle multiple entries).
Test Data	UserData = (User@123, useremail@domain.com) ItemData = (Item Name, Item Description, Category ID) CategoryData = (Category Name, Category Description)

Step No.	**Step Detail**	**Expected Results**	**Actual Results**	**Status**
1	Admin simulates 75 requests consecutively	API should process all requests within 15 seconds without errors.	<15 seconds for all	Successful

1.5.2 Stress Testing

Stress testing evaluates the system's performance under extreme conditions to identify its breaking point. Unlike load testing, which measures how the system performs under normal to high loads, stress testing pushes the system beyond its expected capacity to observe how it reacts under overload scenarios. It helps

uncover vulnerabilities such as system crashes, performance degradation, or unexpected behavior when subjected to stress, and ensures that the system recovers gracefully from failure.

The stress test in Table 1.48 verifies system's ability to handle a varying number of requests through the API under stress conditions.

TABLE 1.48: Stress Test Case

Test Case ID	STR_001
Test Case Description	Verify the system's ability to handle a varying number of requests through the API under stress conditions.
Scenario	Test the API's performance when simulating 100 users simultaneously to determine how the system performs under stress.
Prerequisites	Admin API must be running. API should be able to handle user registration requests. Ensure database is prepared for stress testing (can handle multiple entries).
Test Data	UserData = (User@123, useremail@domain.com) ItemData = (Item Name, Item Description, Category ID) CategoryData = (Category Name, Category Description)

Step No.	**Step Detail**	**Expected Results**	**Actual Results**	**Status**
1	Admin simulates 100 users consecutively	API should process all requests within 30 seconds without errors.	<30 seconds for all	Successful

1.5.3 Endurance Testing

Endurance testing evaluates the system's ability to handle a constant load over an extended period, ensuring that the system can function continuously without

failure or significant degradation. This section contains endurance test cases for both the admin and user sides.

The endrance test in Table 1.49 verifies system's ability to handle requests continuously over an extended period under constant load conditions.

Table 1.49: Endurance Test Case

Test Case ID	ET_001
Test Case Description	Verify the system's ability to handle requests continuously over an extended period under constant load conditions.
Scenario	Test the API's performance while simulating requests consistently over 2 hours, simulating 75 requests every 10 minutes.
Prerequisites	Admin API must be running. API should be able to handle user registration requests. Ensure database is prepared for endruence testing (can handle multiple entries).
Test Data	UserData = (User@123, useremail@domain.com) ItemData = (Item Name, Item Description, Category ID) CategoryData = (Category Name, Category Description)

Step No.	**Step Detail**	**Expected Results**	**Actual Results**	**Status**
1	simulate 75 requests every 10 minutes for 2 hours	System remains stable, successfully processing all requests with a response time under 7 seconds per user.	As Expected	Successful

1.5.4 Spike Testing

Spike testing evaluates the system's ability to handle sudden traffic spikes, ensuring that it can handle abrupt increases in load without crashing or degrading

significantly. This section contains test cases for both the admin and user sides, focusing on their reactions to sudden large spikes in traffic.

The spike test in Table 1.50 verifies the API's ability to handle sudden traffic spikes during request simulation.

TABLE 1.50: Spike Test Case

Test Case ID	SPT_001
Test Case Description	Verify the API's ability to handle sudden traffic spikes during request simulation.
Scenario	Test the API's behavior when hit with 100 requests within a short time window (e.g., 10-30 seconds). The traffic surge should reflect real user actions that can lead to system overload.
Prerequisites	Admin API must be running. Database should be capable of handling large spikes in traffic. Admin credentials must be available for testing. Simulated users should be accessing the system at random intervals to mimic real user traffic.
Test Data	UserData = (User@123, useremail@domain.com) ItemData = (Item Name, Item Description, Category ID) CategoryData = (Category Name, Category Description)

Step No.	Step Detail	Expected Results	Actual Results	Status
1	Simulate 100 requests within 10 seconds	The system should be able to process 80-90 percent of these requests within expected response times (under 5 seconds per request). Some delays and timeouts may occur for the last few requests.	Some delays experienced. Response time for the last 10 requests exceeded 10 seconds.	Partially Successful

1.6 Security Testing

With the rise of cloud-based testing platforms and cyber attacks, there is a growing concern and need for the security of data being used and stored in software. Security testing is a non-functional software testing technique used to determine if the information and data in a system is protected. The goal is to purposefully find loopholes and security risks in the system that could result in unauthorized access to or the loss of information by probing the application for weaknesses. There are multiple types of this testing method, each of which aimed at verifying five basic principles of security:

- Integrity
- Confidentiality
- Authentication
- Authorization
- Availability

1.6.1 Confidentiality Testing

The confidentiality tests ensure that sensitive data such as admin credentials, user photos, and Kinect body data are securely stored and accessed only by authorized

users. These tests also validate that the system respects user consent for data usage.

1.6.1.1 Secure Token Storage

The confidentiality test in Table 1.51 present that tokens would be stored securely to present relevent logged-in user data.

Table 1.51: Confidentiality Test Case:Token Storage

Test Case ID	CT_001			
Test Case Description	Verify that admin session tokens are securely stored and not exposed to unauthorized parties.			
Scenario	Ensure that session tokens are securely stored, and access to tokens is restricted.			
Prerequisites	Admin sessions must be authenticated successfully. Session tokens must be securely stored.			
Test Data	ValidAdminCredentials = (Admin@123, adminemail@gmail.com)			
Step No.	**Step Detail**	**Expected Results**	**Actual Results**	**Status**
1	Admin logs in and the session token is generated and stored.	Token is stored securely.	As Expected	Successful

1.6.1.2 User Consent

The confidentiality test in Table 1.52 present that user body data would be captured by kinect with user consent only.

Table 1.52: Confidentiality Test Case:User Consent

Test Case ID	CT_002
Test Case Description	Verify that users provide explicit consent for the use of their body data from Kinect and are informed that their data will not be used by third parties.

Scenario	Ensure that users are aware of and consent to the collection of their body data, and that they are informed about the confidentiality of this data.
Prerequisites	Kinect body tracking must be active during the virtual try-on session. A consent form should be presented to the user before starting the try-on session.
Test Data	ValidConsent = ("I agree to allow body data collection for virtual try-on") InvalidConsent = ("User does not accept the terms")

Step No.	**Step Detail**	**Expected Results**	**Actual Results**	**Status**
1	User is presented with a consent form before the virtual try-on session begins	User is informed that their body data will not be used by third parties. User must provide consent to proceed.	As Expected	Successful
2	User accepts the consent form	Kinect body data is collected, and user can proceed with the virtual try-on session.	As Expected	Successful
3	User refuses to accept the consent form	User cannot proceed with the virtual try-on session, and data is not collected.	As Expected	Successful

1.6.2 Integrity Testing

Integrity testing ensures that the data stored in the database or transferred through the system remains accurate, consistent, and unaltered during the system's operations.

1.6.2.1 API Integrity

The Integrity test in Table 1.53 presents that the data transmitted through the API remains consistent and unaltered.

TABLE 1.53: Integrity Test Case:API

Test Case ID	IT_001
Test Case Description	Verify the integrity of the data transmitted through the API, ensuring that it is not altered during requests and responses.
Scenario	Test the accuracy of data transmitted between the client, API, and database, ensuring that no data corruption or modification occurs.
Prerequisites	API must be running and accessible. Database must be populated with test data. The system must have endpoints for adding, updating, and deleting clothing models or user data.
Test Data	ValidAPIData = (Clothing Model Name: "Jeans", Model Path: "path/to/jeans.fbx") InvalidAPIData = (Clothing Model Name: "Jeans", Model Path: NULL)

Step No.	Step Detail	Expected Results	Actual Results	Status
1	Send a valid API request to insert a new clothing model	The API correctly inserts the model and the data matches the sent request	As Expected	Successful
2	Send an API request to update an existing model's details	The API successfully updates the model and the changes are reflected in the database	As Expected	Successful
3	Send an API request to delete an existing model	The model is successfully removed from the database	As Expected	Successful
4	Send an invalid API request with missing data (e.g., model path)	The API returns an error and does not update the database with invalid data	As Expected	Successful
5	Verify that data in the API response matches the expected format and values	The API response is accurate and contains unaltered data	As Expected	Successful

1.6.2.2 Virtual TryOn Integrity

The Integrity test in Table 1.54 present that the clothing models are accurately applied to the user's body without distortion.

TABLE 1.54: Integrity Test Case:Virtual Try-On

Test Case ID	IT_002

Test Case Description	Verify the integrity of the virtual try-on data, ensuring that clothing models are accurately applied to the user's body without distortion.
Scenario	Test the accurate overlay of clothing models on users during the try-on session. Ensure the models are not altered and align correctly with the user's body.
Prerequisites	Kinect sensor must be connected and tracking the user. The system must have a clothing model ready for try-on. The virtual try-on feature must be enabled.
Test Data	ValidTryOnData = (Clothing Model: "Shirt", Model Path: "path/to/shirt.fbx") InvalidTryOnData = (Clothing Model: "Shirt", Invalid Model Path: "invalid/-path/to/shirt.fbx")

Step No.	**Step Detail**	**Expected Results**	**Actual Results**	**Status**
1	User selects a clothing model for try-on	The clothing model is accurately displayed and overlaid on the user's body	As Expected	Successful
2	User moves around with the virtual clothing applied	The clothing model moves and adjusts to the user's body correctly, without distortion or misalignment	As Expected	Successful

1.6.3 Authentication Testing

The authentication tests verify that the login processes works correctly, ensuring that only authorized users can access the system.

1.6.3.1 Admin Authentication

The admin authentication test in Table 1.55 verify that the admin login process works correctly, ensuring that only authorized admins can access the system.

TABLE 1.55: Authentication Test Case:Admin

Test Case ID	AAT_001
Test Case Description	Verify the admin login functionality using valid and invalid credentials.
Scenario	Test if the admin can successfully log in using correct credentials and if the system denies access for incorrect credentials.
Prerequisites	Admin login page should be accessible. Admin credentials (username and password) should be available.
Test Data	ValidAdminCredentials = (admin@tryon.com, Admin123) InvalidAdminCredentials = (wronguser@tryon.com, wrongpassword)

Step No.	**Step Detail**	**Expected Results**	**Actual Results**	**Status**
1	Admin enters valid credentials	Admin is logged in successfully and redirected to the dashboard.	As Expected	Successful
2	Admin enters invalid credentials	Login attempt is denied, and an error message "Invalid credentials" is displayed.	As Expected	Successful

1.6.3.2 User Authentication

The user authentication test in Table 1.56 verify that the user term acceptance process works correctly.

TABLE 1.56: Authentication Test Case:User Accept Terms

Test Case ID	AAT_002
Test Case Description	Verify that the user is prompted to reaccept the terms if the Kinect sensor gets disconnected.
Scenario	Ensure the user is prompted to reaccept the terms and conditions after a sensor disconnection.
Prerequisites	Kinect sensor must be connected and tracking the user. User must be logged in and have accepted terms.
Test Data	IsTermsAccepted=True

Step No.	**Step Detail**	**Expected Results**	**Actual Results**	**Status**
1	User accepts terms	User is able to proceed with the virtual try-on session.	As Expected	Successful
2	Kinect sensor is disconnected during the session	User is prompted to reaccept the terms before continuing.	As Expected	Successful

1.6.4 Authorization Testing

The authorization tests in Table 1.57 for the system ensure that users and admins only have access to the resources and functionalities they are authorized for, based on their roles.

Table 1.57: Authorization Test Case

Test Case ID	AT_001
Test Case Description	Verify that users can access only user-related resources, and admins have access to admin-level resources.
Scenario	Test that unauthorized access to resources is restricted based on user roles.

Prerequisites	Admin user is logged in with admin credentials. Regular user is logged in with standard user credentials. The system is configured with role-based access for admin and user functionalities.
Test Data	ValidAdminCredentials = (admin@admin.com, adminpassword) IsTermsAccepted=True

Step No.	**Step Detail**	**Expected Results**	**Actual Results**	**Status**
1	Admin logs in with valid admin credentials	Admin dashboard is displayed with access to all admin features (e.g., adding models, viewing user data)	As Expected	Successful
2	User accepts terms and conditions	User dashboard is displayed, restricting access to user-level features (e.g., virtual try-on)	As Expected	Successful

1.6.5 Availability Testing

The availability test in Table 1.58 ensures that the system remains accessible and responsive, even in adverse conditions, such as when the API is unavailable or when the client UI requires refreshing.

Table 1.58: Availability Test Case

Test Case ID	AUT_001

Test Case Description	Verify the availability of the system, ensuring that both admin and user can access functionalities and receive appropriate error handling in case of failures.
Scenario	Test system availability by simulating different failure scenarios, ensuring the system recovers smoothly and provides proper user feedback.
Prerequisites	The system must be running and accessible. The user must have an active session. The server and API should be functioning.
Test Data	ValidUserSession = (“Active session”) APIUnavailable = (“Simulated server downtime”)

Step No.	**Step Detail**	**Expected Results**	**Actual Results**	**Status**
1	User refreshes the page after modification in admin panel	Client UI refreshes correctly and displays updated content without errors	As Expected	Successful
2	Admin or user tries to access the system while the API is down (simulate server downtime)	Displays a user-friendly error message: "Service unavailable. Please try again later."	As Expected	Successful
3	User tries to perform an action while the API is temporarily unavailable	Displays appropriate message: "Action cannot be completed due to service unavailability."	As Expected	Successful
4	Admin attempts to upload a new model or image while the server is down	Displays error message: "Unable to upload due to server unavailability."	As Expected	Successful
5	User interacts with the virtual try-on while the backend server is recovering from downtime	System provides a prompt stating: "System recovering. Some features may be temporarily unavailable."	As Expected	Successful
6	Admin accesses system after server recovery	Admin access is restored, and functionality is seamless without requiring login again	As Expected	Successful

1.7 Usability Testing

Usability testing is a testing method that measures an application's ease-of-use from the end-user perspective and is often performed during the system or acceptance testing stages. The goal is to determine whether or not the visible design and aesthetics of an application meet the intended workflow for various processes, such as logging into an application[4].

1.7.1 Signup View Usability Testing

The signup view usability testing in Table 1.59 ensures that users can efficiently and intuitively complete the registration process. It evaluates the visibility and clarity of input fields, the effectiveness of error messages, and the ease of completing the signup process. The test verifies whether users can locate and interact with essential elements, such as the name, email, password, confirm password fields, and the signup button.

TABLE 1.59: Usability Test Case:Signup View

Test Case ID	UT_001
Test Case Description	Verify the usability of the signup view, ensuring users can sign up without confusion or difficulty.
Scenario	Test whether users can successfully navigate the signup page, enter valid details, and complete the registration without usability issues.
Prerequisites	The signup page should be accessible, and the system should allow new user registrations.
Test Data	ValidUser = (John Doe, user@exa.com, Pass@123, Pass@123) InvalidUser = (John123, userexample.com, Pass@321, weakpass, existing@admin.com)

Step No.	**Step Detail**	**Expected Results**	**Actual Results**	**Status**
1	User easily locates the signup fields (name, email, password, confirm password)	Fields are clearly visible and labeled.	As Expected	Successful
2	User enters valid details	Signup fields accept input without issues.	As Expected	Successful
3	User clicks the signup button after entering valid details	User is successfully registered and redirected to the login page.	As Expected	Successful
4	User enters an invalid name	Error message appears: "Invalid name format."	As Expected	Successful
5	User enters an invalid email format	Error message appears: "Invalid email format."	As Expected	Successful
6	User enters an existing email	Error message appears: "Email already in use."	As Expected	Successful
7	User enters mismatched passwords	Error message appears: "Passwords do not match."	As Expected	Successful
8	User enters a weak password	Error message appears: "Password must meet security requirements."	As Expected	Successful
9	User signup with empty fields	Error message appears: "All fields are required."	As Expected	Successful

1.7.2 Login View Usability Testing

The login view usability testing in Table 1.60 ensures that users can efficiently and intuitively navigate the login interface. It evaluates the visibility and clarity of input fields, the effectiveness of error messages, and the ease of completing the login process. The test verifies whether users can locate and interact with essential elements, such as the email and password fields, login button, password.

TABLE 1.60: Usability Test Case:Login View

Test Case ID	UT_002
Test Case Description	Verify the usability of the login view, ensuring users can log in without confusion or difficulty.
Scenario	Test whether users can successfully navigate the login page, enter valid credentials, and access the system without usability issues.
Prerequisites	The login page should be accessible, and a set of valid user credentials should be available.
Test Data	ValidUser = (user@example.com, Admin@123) InvalidUser = (invalid@example.com, wrongpass)

Step No.	Step Detail	Expected Results	Actual Results	Status
1	User easily locates the login fields (email and password)	Fields are clearly visible and labeled.	As Expected	Successful
2	User enters valid credentials	Login fields accept input without issues.	As Expected	Successful
3	User clicks the login button after entering valid credentials	User is successfully logged in and redirected to the dashboard.	As Expected	Successful
4	User enters invalid credentials	Error message is clear and helpful.	As Expected	Successful
5	User attempts login without filling fields	Appropriate validation messages are displayed.	As Expected	Successful

1.7.3 Update Profile Usability Testing

The update profile usability testing in Table 1.61 ensures that users can efficiently and intuitively edit their name and email while understanding the functionality of the save and cancel buttons. It evaluates the visibility and clarity of input fields, the effectiveness of feedback messages, and the ease of completing or discarding changes. The test verifies whether users can locate and interact with essential elements without usability confusion.

Table 1.61: Usability Test Case:Update Profile

Test Case ID	UT_003
Test Case Description	Verify the usability of the update profile view, ensuring users can edit their name and email, then save or cancel changes as expected.

Scenario	Test whether users can successfully edit their profile details, understand the function of the save and cancel buttons, and receive appropriate feedback.
Prerequisites	The user should be logged in, and the update profile page should be accessible.
Test Data	ValidUser = (User edits name and email, then saves or cancels changes correctly) InvalidUser = (User struggles with UI, unclear validation, or unexpected button behavior)

Step No.	**Step Detail**	**Expected Results**	**Actual Results**	**Status**
1	User easily locates and identifies the editable name and email fields	Fields are clearly visible and distinguishable from non-editable ones.	As Expected	Successful
2	User enters a valid name and email	System accepts the input without issues.	As Expected	Successful
3	User enters an invalid name (e.g., numbers or special characters)	System displays a validation message for proper name format.	As Expected	Successful
4	User enters an invalid email (e.g., missing '@' or domain)	System displays an appropriate error message.	As Expected	Successful
5	User clicks the Save button after making valid changes	Changes are successfully saved, and feedback confirms the update.	As Expected	Successful
6	User clicks the Save button with invalid input	System prevents saving and displays appropriate validation errors.	As Expected	Successful
7	User clicks the Cancel button after making changes	Changes are discarded, and the previous profile data remains unchanged.	As Expected	Successful
8	User expects an immediate response after clicking save	System provides loading feedback or success confirmation.	As Expected	Successful

1.7.4 Add Category Usability Testing

The add category usability testing in Table 1.62 ensures that users can efficiently navigate the add category interface, enter a category name, upload an image, and use the add button correctly. It evaluates the visibility and clarity of input fields, the effectiveness of error messages for invalid inputs, and the ease of completing the process.

TABLE 1.62: Usability Test Case:Add Category

Test Case ID	UT_004
Test Case Description	Verify the usability of the add category view, ensuring users can enter a category name, upload an image, and successfully add a category.
Scenario	Test whether users can easily enter valid category details, handle validation errors, and understand the add button's functionality.
Prerequisites	The add category page should be accessible.
Test Data	ValidUser = (User enters a valid category name and image, then clicks add) InvalidUser = (User enters invalid/missing details or faces UI confusion)

Step No.	**Step Detail**	**Expected Results**	**Actual Results**	**Status**
1	User easily locates the category name and image upload fields	Fields are clearly visible and properly labeled.	As Expected	Successful
2	User enters a valid category name	System accepts the input without issues.	As Expected	Successful
3	User uploads a valid image	System accepts the image file and displays a preview if applicable.	As Expected	Successful
4	User clicks the Add button with valid inputs	System successfully adds the category and provides success feedback.	As Expected	Successful
5	User clicks the Add button without entering a category name	System displays an error: "Category name is required."	As Expected	Successful
6	User clicks the Add button without uploading an image	System displays an error: "Category image is required."	As Expected	Successful
7	User enters an invalid category name (e.g., special characters or too long)	System displays an appropriate validation message.	As Expected	Successful
8	User uploads an unsupported file format as an image	System prevents upload and displays an error message.	As Expected	Successful

1.7.5 Update Category Usability Testing

The update category usability testing in Table 1.63 ensures that users can efficiently navigate the update category interface, modify the category name, update the image, and use the update button correctly. It evaluates the visibility and clarity of input fields, the effectiveness of validation messages for incorrect inputs, and the ease of completing the update process.

TABLE 1.63: Usability Test Case:Update Category

Test Case ID	UT_005
Test Case Description	Verify the usability of the update category view, ensuring users can modify the category name, update the image, and successfully save changes.
Scenario	Test whether users can easily edit valid category details, handle validation errors, and understand the update button's functionality.
Prerequisites	The update category page should be accessible, and a category should already exist for editing.
Test Data	ValidUser = (User modifies the category name and/or image, then updates successfully) InvalidUser = (User enters invalid/missing details or faces UI confusion)

Step No.	**Step Detail**	**Expected Results**	**Actual Results**	**Status**
1	User easily locates the category name and image upload fields	Fields are clearly visible and properly labeled.	As Expected	Successful
2	User modifies the category name	System accepts the input without issues.	As Expected	Successful
3	User uploads a new valid image	System accepts the image file and displays a preview if applicable.	As Expected	Successful
4	User clicks the Update button with valid inputs	System successfully updates the category and provides success feedback.	As Expected	Successful
5	User clicks the Update button without entering a category name	System displays an error: "Category name is required."	As Expected	Successful
6	User clicks the Update button without uploading an image (if required)	System displays an error if the image is mandatory.	As Expected	Successful
7	User enters an invalid category name (e.g., special characters or too long)	System displays an appropriate validation message.	As Expected	Successful
8	User uploads an unsupported file format as an image	System prevents upload and displays an error message.	As Expected	Successful

1.7.6 Add Item Usability Testing

The add item usability testing in Table 1.64 ensures that users can efficiently navigate the add item interface, input all required details, select appropriate options, and successfully add an item. It evaluates the visibility of input fields, the effectiveness of validation messages, and the ease of completing the add process.

TABLE 1.64: Usability Test Case:Add Item

Test Case ID	UT_006
Test Case Description	Verify the usability of the add item view, ensuring users can enter item details correctly and submit the form without confusion.
Scenario	Test whether users can locate and fill out all fields properly, handle validation errors, and understand the add button's functionality.
Prerequisites	The add item page should be accessible, and users should have permissions to add new items.
Test Data	ValidUser = (User fills all fields correctly and adds the item) InvalidUser = (User enters invalid/missing details or faces UI confusion)

Step No.	**Step Detail**	**Expected Results**	**Actual Results**	**Status**
1	User easily locates the name, price, quantity, model, and details fields	Fields are clearly visible and labeled correctly.	As Expected	Successful
2	User enters a valid name, price, quantity, model, and details	System accepts the input without issues.	As Expected	Successful
3	User selects valid options from the Color, Size, and Fabric dropdowns	Selected options are correctly displayed.	As Expected	Successful
4	User clicks the Add button after filling in all details correctly	System successfully adds the item and provides success feedback.	As Expected	Successful
5	User clicks the Add button without entering any details	System displays validation messages: "Fields cannot be empty."	As Expected	Successful
6	User uploads an invalid model file format	System prevents upload and displays an error message.	As Expected	Successful
7	User clicks the Add button without selecting a color, size, or fabric	System displays an error message: "Please select all required options."	As Expected	Successful

1.7.7 Update Item Usability Testing

The update item usability testing in Table 1.65 ensures that users can efficiently navigate the update item interface, modify item details, select appropriate options, and successfully update an item. It evaluates the visibility of input fields, the effectiveness of validation messages, and the ease of completing the update process.

TABLE 1.65: Usability Test Case:Update Item

Test Case ID	UT_007
Test Case Description	Verify the usability of the update item view, ensuring users can modify item details correctly and submit the update without confusion.
Scenario	Test whether users can locate and edit all fields properly, handle validation errors, and understand the update button's functionality.
Prerequisites	The update item page should be accessible, and users should have permissions to update existing items.
Test Data	ValidUser = (User modifies item details correctly and updates successfully) InvalidUser = (User enters invalid/missing details or faces UI confusion)

Step No.	Step Detail	Expected Results	Actual Results	Status
1	User easily locates the name, price, quantity, model, and details fields	Fields are clearly visible and prefilled with existing item details.	As Expected	Successful
2	User modifies name, price, quantity, model, and details with valid inputs	System accepts the input without issues.	As Expected	Successful
3	User selects valid options from the Color, Size, and Fabric dropdowns	Selected options are correctly displayed.	As Expected	Successful
4	User clicks the Update button after modifying details correctly	System successfully updates the item and provides success feedback.	As Expected	Successful
5	User clears all fields and clicks the Update button	System displays validation messages: "Fields cannot be empty."	As Expected	Successful
6	User clicks the Update button without selecting a color, size, or fabric	System displays an error message: "Please select all required options."	As Expected	Successful

1.7.8 Accept Terms Usability Testing

The accept terms usability testing in Table 1.66 ensures that users can clearly understand and interact with the terms and conditions agreement interface. It

evaluates whether users can locate the accept terms option, comprehend its necessity, and proceed without confusion.

Table 1.66: Usability Test Case:Accept Terms

Test Case ID	UT_008
Test Case Description	Verify the usability of the accept terms functionality, ensuring users can accept the terms before proceeding.
Scenario	Test whether users can locate and interact with the accept terms checkbox/button and understand the consequences of their choice.
Prerequisites	The accept terms page should be accessible, and users should be required to accept terms before proceeding.
Test Data	ValidUser = (User understands and accepts terms successfully) InvalidUser = (User struggles to locate or interact with the terms checkbox)

Step No.	Step Detail	Expected Results	Actual Results	Status
1	User easily locates the terms and conditions text and checkbox/button	Terms text and checkbox/button are visible and clearly labeled	As Expected	Successful
2	User reads the terms and conditions before making a selection	Text is readable without UI issues, and scrolling works properly	As Expected	Successful
3	User selects the checkbox or clicks the accept button (userAcceptTerms = true)	User is allowed to proceed to the next screen	As Expected	Successful
4	User attempts to proceed without accepting the terms (userAcceptTerms = false)	System prevents access and displays a message: "You must accept the terms to continue."	As Expected	Successful
5	User encounters unclear instructions or placement issues	System provides proper guidance or improves layout if needed	As Expected	Successful

1.7.9 User Select Categories Usability Testing

The select categories usability testing in Table 1.67 ensures that users can easily identify and select a category to explore clothing items. It evaluates the clarity of category selection, responsiveness, and system behavior when no categories are available.

TABLE 1.67: Usability Test Case:Select Category

Test Case ID	UT_009
Test Case Description	Verify the usability of selecting a clothing category, ensuring users can clearly understand and interact with the selection process.
Scenario	Test whether users can easily select a category, recognize highlighted selections, and receive appropriate feedback when no categories exist.
Prerequisites	The categories page must be accessible, and categories should be loaded.
Test Data	ValidUser = (User successfully selects and highlights a category) InvalidUser = (User faces difficulty in selecting or identifying selected categories)

Step No.	**Step Detail**	**Expected Results**	**Actual Results**	**Status**
1	User navigates to the categories page	Categories should be clearly displayed	As Expected	Successful
2	User selects a category and navigates to the items page	Correct category's items should be displayed	As Expected	Successful

1.7.10 Try-On Clothing Usability Testing

The usability test in Table 1.68 for the try-on clothing feature evaluates how intuitively users can interact with the virtual try-on system. It measures the accuracy of clothing alignment, responsiveness, and user satisfaction.

TABLE 1.68: Usability Test Case:Try-On Clothing

Test Case ID	UT_010
Test Case Description	Verify the usability of the try-on clothing functionality, ensuring accurate overlay and user-friendly interaction.

Scenario	Test how easily users can view themselves wearing virtual clothing and assess the system's accuracy and responsiveness.
Prerequisites	Kinect sensor must be connected and tracking the user. At least one item must be selected for try-on.
Test Data	ValidUser = (User successfully tries on clothing with proper alignment) InvalidUser = (User faces difficulties in overlay accuracy or responsiveness)

Step No.	**Step Detail**	**Expected Results**	**Actual Results**	**Status**
1	User selects a clothing item and stands in front of the Kinect sensor	Selected clothing overlays correctly on the user's body	As Expected	Successful
2	User moves around while wearing the virtual clothing	Clothing follows the user's body movements accurately	As Expected	Successful
3	User attempts try-on without standing in the Kinect's view	Displays "User not detected"	As Expected	Successful
4	User tests the accuracy of clothing fit by turning sideways	Clothing maintains realistic alignment with the user's body	As Expected	Successful
5	User tries different clothing items rapidly	Application responds without lag or glitches	As Expected	Successful
6	User attempts try-on with Kinect disconnected	Displays error "Kinect not connected"	As Expected	Successful

1.7.11 Change Background Usability Testing

The usability test in Table 1.69 for the change background feature evaluates how easily users can switch between different backgrounds and assesses the smoothness of transitions.

TABLE 1.69: Usability Test Case:Change Background

Test Case ID	UT_011
Test Case Description	Verify the usability of the change background functionality, ensuring smooth transitions and intuitive selection.
Scenario	Test how efficiently users can switch between backgrounds and evaluate transition quality.
Prerequisites	Kinect sensor must be connected and tracking the user. At least one background must be available for selection.
Test Data	ValidBackground = (“Beach”, “City”, “Studio”) InvalidBackground = (No available backgrounds)

Step No.	Step Detail	Expected Results	Actual Results	Status
1	User selects a background from the available options	Background changes smoothly to the selected option	As Expected	Successful
2	User rapidly switches between different backgrounds	Background transitions occur without noticeable lag	As Expected	Successful
3	User attempts to change background when no options are available	Displays "No backgrounds available"	As Expected	Successful
4	User tries changing backgrounds with Kinect disconnected	Displays error "Kinect not connected"	As Expected	Successful

1.7.12 Save Picture Usability Testing

The usability test in Table 1.70 for the save picture feature evaluates how easily users can capture and store images of their virtual try-on session. It assesses the intuitiveness of the process and system feedback.

Table 1.70: Usability Test Case:Save Picture

Test Case ID	UT_012
Test Case Description	Verify the usability of the save picture functionality, ensuring a smooth and user-friendly experience.
Scenario	Test how efficiently users can capture and save images, ensuring proper system response and error handling.

Prerequisites	Try-on session must be active. Camera feed must be available. Adequate storage space must be available.
Test Data	ValidCapture = (Image captured successfully) InvalidCapture = (Capture failed, storage full, camera unavailable)

Step No.	**Step Detail**	**Expected Results**	**Actual Results**	**Status**
1	User clicks the capture button while try-on session is active	Image is saved to gallery with confirmation message	As Expected	Successful
2	User tries saving an image when capture fails	Displays "Failed to save image" with retry option	As Expected	Successful
3	User attempts to save an image with insufficient storage	Displays "Storage full. Unable to save image"	As Expected	Successful
4	User tries saving an image when Kinect camera is disconnected	Displays "Camera not available" with troubleshooting guide	As Expected	Successful

1.8 Compatibility Testing

Compatibility testing in Table 1.71 ensures that the software application is compatible with hardware, operating systems, web browsers, and other third-party software. [3].

TABLE 1.71: Compatibility Test Case

Test Case ID	COT_001
Test Case Description	Verify compatibility of virtual try-on system.

Scenario	Ensure the virtual try-on system works on specified operating systems, hardware, and different lighting conditions.
Prerequisites	System is set up with required hardware and software versions.
Test Data	Operating system = Windows 10 Kinect v2 sensor Camera Lighting = Normal, Low

Step No.	**Step Detail**	**Expected Results**	**Actual Results**	**Status**
1	Test OS Compatibility (Success)	Virtual try-on works on Windows 10.	As expected	Successful
2	Test Camera Compatibility (Success)	Kinect v2 sensor tracks body movements accurately for proper alignment under normal lighting.	As expected	Successful
3	Test Camera Compatibility (Failure)	Kinect v2 sensor does not track body movements accurately under low lighting conditions.	As expected	Successful
4	Test Camera Compatibility (Failure)	Kinect v2 sensor fails to track body movements when placed far from the user.	As expected	Successful
5	Test OS Compatibility (Success)	Virtual try-on works on Windows 10 with required updates installed.	As expected	Successful
6	Test Kinect v1 SDK Compatibility (Failure)	Kinect v1 SDK does not work with Kinect v2 sensor.	As expected	Successful

References

[1] Unit testing definition - searchsoftwarequality. URL https://www.techtarget.com/searchsoftwarequality/definition/unit-testing#:~{}:text=Unit%20testing%20is%20a%20software,independently%20scrutinized%20for%20proper%20operation.

[2] GeeksforGeeks. Acceptance testing — software testing - geeksforgeeks, 2019. URL https://www.geeksforgeeks.org/acceptance-testing-software-testing/. Accessed: 2025-03-24.

[3] GeeksforGeeks. Compatibility testing in software engineering, 2021. URL https://www.geeksforgeeks.org/compatibility-testing-in-software-engineering/. [Online; accessed 18-April-2023].

[4] SmartBear. Software testing methodologies, 2025. URL http://smartbear.com/learn/automated-testing/software-testing-methodologies/. Accessed: Mar. 20, 2025.