Bug Report

Group 44

Group Members

Index No	Name
214133E	Mendis B.N.D.
215023B	Danthanarayana S.
215131E	Dissanayake A. S. H.
215107L	Samarasinghe S. A. T. L. T.
215108P	Samaraweera P.K.L.P

1. Categories

Project: qa-training-app

Bug ID: BUG001 (API_TC_01)

Date Reported: 2026-02-01

Reported by: 214133E

Assigned to: Developer

Status: New

Priority: Low

Severity: Major

Component: API

Summary: [API - Categories - Create Category] System accepts Numeric data type for 'Name' field instead of enforcing String validation

Description:

According to the Swagger documentation, the 'name' field is explicitly defined as a String (e.g., "Anthurium"). However, when the user sends a raw Integer value (e.g., 54839) in the request body, the API successfully processes the request and creates the category with the numeric name.

This indicates a failure in input validation, as the system should reject data types that do not match the schema with a 400 Bad Request to maintain data integrity.

Steps to reproduce:

1. Ensure the API Service is running and an Admin Token is generated.
2. Send a POST request to /api/categories.
3. Set the request body to use a raw **Integer** for the name field:

   ```
   { "name": 54839, "parent": null }
   ```
4. Observe the response status code.

Expected result:

1. The API should return **400 Bad Request**.

2. The system should validate the data type and reject the request because the 'name' field requires a String.

Actual result:

1. The API returns 201 Created.
2. The system failed to perform data type validation and successfully created a category with the numeric name 54839.

Logs if exist:

Request: POST /api/categories

Request Body: { "name": 54839, "parent": null }

Actual Response Status: 201 Created

Cypress Assertion Error: expected 201 to equal 400

Project: qa-training-app

Bug ID: BUG002 (API_TC_02)

Date Reported: 2026-02-01

Reported by: 214133E

Assigned to: Developer

Status: New

Priority: High

Severity: Critical

Component: API

Summary: [API - Categories - Create Category] 500 Internal Server Error returned when creating a category with non-existent Parent ID instead of returning validation error

Description:

Executed Test Case: Verify that the API rejects creation with a non-existent Parent ID. The user attempted to create a category linking to a non-existent parent "99999" (sent as a String per Swagger docs). Instead of returning a 404 Not Found or 400 Bad Request, the system crashed with a 500 Internal Server Error.

Root Cause Analysis: The error log indicates a "JSON parse error: Cannot construct instance of... from String value". This suggests the API crashes on **ANY** String input for the 'parent' field, contradicting the Swagger documentation which defines it as a String.

Steps to reproduce:

1. Ensure API Service is running and Admin Token is generated.
2. Send a POST request to /api/categories.
3. Set Headers: Authorization: Bearer <Token>.
4. Set Body: {"name": "Ghost_Category", "parent": "99999"} (Non-existent parent string).
5. Execute request.

Expected result:

1. Status Code: 400 Bad Request or 404 Not Found.
2. Response should indicate "Parent not found" or "Validation failed".

Actual result:

1. Status Code: **500 Internal Server Error**.
2. The server crashed without a proper error message.

Logs if exist:

Request: POST /api/categories

Request Body: { "name": "Ghost_[Timestamp]", "parent": "99999" }

Actual Response Status: 500 Internal Server Error

Cypress Assertion Error: expected 500 to equal 400

Project: qa-training-app

Bug ID: BUG003 (API_TC_06)

Date Reported: 2026-02-01

Reported by: 214133E

Assigned to: Developer

Status: New

Priority: Low

Severity: Minor

Component: API

Summary: [API - Categories - Create Category] API silently ignores undefined field 'parentId' and creates a Root Category instead of throwing a validation error.

Description:

The user attempted to create a sub-category by including a 'parentId' field in the request body (e.g., { "parentId": "10" }). This field is not defined in the Swagger schema.
Instead of rejecting the request with a 400 Bad Request due to an unknown or invalid field, the API returned a 201 Created status. However, it **silently ignored** the 'parentId' field and created a Root Category (Parent = null) instead. This "silent failure" leads to data inconsistency, as the system confirmed success without performing the user's intended action of linking a parent.

Steps to reproduce:

1. Send POST request to /api/categories.
2. Set Body to include a field not defined in the Swagger schema:
   { "name": "SubTest", "parentId": "10" }
3. Send Request.

Expected result:

1. Status **400 Bad Request**.
2. The API should reject the request because parentId is an unknown field, or strict schema validation should fail.

Actual result:

4. Status **201 Created**.
5. The API ignored the unknown parentId field, treated the parent as null, and created a Main Category.

Logs if exist:

Request: POST /api/categories

Request Body: { "name": "Sub[RandomNumber]", "parentId": "10" }

Actual Response Status: 201 Created

Cypress Assertion Error: expected 201 to equal 400

Project: qa-training-app

Bug ID: BUG004 (API_TC_09)

Date Reported: 2026-02-02

Reported by: 214133E

Assigned to: Developer

Status: New

Priority: Medium

Severity: Major

Component: API

Summary: [API - Categories - Filter] 500 Internal Server Error occurs when filtering categories with a non-numeric 'parentId'

Description:

The user attempted to filter the category list using the parentId query parameter with a non-numeric string value (e.g., ?parentId=invalid_text).

Instead of validating the input type and returning a graceful 400 Bad Request message indicating that 'parentId' must be a number, the server crashed with a 500 Internal Server Error. The response body contained a NumberFormatException stack trace, exposing internal system details and indicating a lack of exception handling for query parameters.

Steps to reproduce:

1. Ensure the API Service is running and a User Token is generated.
2. Send a GET request to /api/categories/page.

3. Add a query parameter with a text string value instead of an integer: ?parentId=invalid_text
4. Execute the request.

Expected result:

1. The API should return **400 Bad Request**.
2. The system should gracefully validate the query parameter and reject the request with a clear message indicating that 'parentId' must be a number.

Actual result:

1. The API returns **500 Internal Server Error**.
2. The system failed to handle the string input gracefully and threw a NumberFormatException stack trace in the response body.

Logs if exist:

Request: GET /api/categories/page?parentId=invalid_text

Request Body: N/A

Actual Response Status: 500 Internal Server Error

Cypress Assertion Error: expected 500 to equal 400

Project: qa-training-app

Bug ID: BUG005 (API_TC_10)

Date Reported: 2026-02-02

Reported by: 214133E

Assigned to: Developer

Status: New

Priority: Medium

Severity: Major

Component: API

Summary: [API - Categories - Pagination] 500 Internal Server Error occurs when providing a non-numeric value for 'page' parameter

Description:

The user sent a GET request to retrieve a paginated list of categories but provided a string value for the page parameter (e.g., ?page=one).

Pagination parameters strictly require numeric integer values. However, the API failed to validate the data type or handle the conversion error gracefully. Instead of returning a 400 Bad Request, the system threw an unhandled exception resulting in a 500 Internal Server Error, causing the service endpoint to crash for that request.

Steps to reproduce:

1. Ensure the API Service is running and a User Token is available.
2. Send a GET request to /api/categories/page.
3. Set the 'page' query parameter to a string value: ?page=one&size=5
4. Execute the request.

Expected result:

1. The API should return **400 Bad Request**.
2. The system should validate the data type for pagination parameters and reject non-numeric values.

Actual result:

1. The API returns **500 Internal Server Error**.
2. The system crashed with a type conversion error.

Logs if exist:

Request: GET /api/categories/page?page=one&size=5

Request Body: N/A

Actual Response Status: 500 Internal Server Error

Cypress Assertion Error: expected 500 to equal 400

Project: qa-training-app

Bug ID: BUG006 (UI_TC_43)

Date Reported: 2026-02-03

Reported by: 215131E

Assigned to: Developer

Status: New

Priority: Medium

Severity: Major

Component: UI

Summary: [UI - Categories – Delete] Delete confirmation modal does not load; default browser confirmation dialog is shown instead

Description:

When a user attempts to delete a category from the Category Management page, the system is expected to display a custom-designed confirmation modal to confirm or cancel the delete action. However, instead of rendering the application's modal component, the system triggers the browser’s native confirmation dialog. This indicates that the UI modal logic is not being executed or the modal component is not being rendered properly.

Steps to reproduce:

1. Launch the qa-training-app UI.
2. Navigate to the Categories management page.
3. Locate an existing category in the list.
4. Click on the **Delete** button for the category.

Expected result:

1. A custom delete confirmation modal designed by the application should be displayed.
2. The modal should ask the user to confirm or cancel the delete action

Actual result:

1. The custom delete confirmation modal does not load.
2. Instead, the default browser confirmation dialog is displayed.

Logs if exist:

- Observation: Browser-native confirmation popup appears instead of application modal
- No application modal DOM element rendered on delete action

Project: qa-training-app

Bug ID: BUG007 (UI_TC_48)

Date Reported: 2026-02-03

Reported by: 215131E

Assigned to: Developer

Status: New

Priority: High

Severity: Critical

Component: UI

Summary: [UI - Categories - Edit] Edit button is visible and enabled for Non-Admin users, allowing unauthorized access to Edit functionality

Description:

The system should restrict editing capabilities to Admin users only. However, when logged in as a Non-Admin user, the Edit button is still visible and fully functional in the Category Management page. This allows unauthorized users to access the Edit Category form, indicating that role-based access control is not being enforced at the UI level. This may also suggest a missing permission validation before rendering action buttons.

Steps to reproduce:

1. Log in to the system as a **Non-Admin user**.
2. Ensure at least one Category exists.
3. Navigate to the **Category Management Page**.
4. Observe the action buttons displayed for each Category.
5. Click on the **Edit** button.

Expected result:

3. The **Edit** button should be **hidden or disabled** for Non-Admin users.
4. Unauthorized users should **not be able to access** the Edit Category functionality.
5. The system should restrict Edit access based on user role.

Actual result:

3. The **Edit** button is visible and enabled for Non-Admin users.
4. Clicking the Edit button redirects the user to the **Edit Category form**.
5. The system fails to enforce role-based access control at the UI level.

Logs if exist:

Observation: Edit button visible and clickable

Result: Unauthorized user can access Edit Category page

Project: qa-training-app

Bug ID: BUG008 (UI_TC_50)

Date Reported: 2026-02-04

Reported by: 215131E

Assigned to: Developer

Status: New

Priority: Medium

Severity: Major

Component: UI

Summary: [UI - Categories – Sorting Parent] Sorting by "Parent Category" column updates the sort icon but fails to reorder the list rows

Description:

When a user clicks the "Parent" column header in the Category List, the sort arrow icon toggles (Ascending/Descending), but the actual rows in the table do not change their order. Categories with empty parents (-) remain mixed/interleaved with categories that have defined parents, rather than being grouped together as expected in a standard sort.

Steps to reproduce:

1. Navigate to the **Category Management** page.
2. Observe the default order of the list (usually sorted by ID Descending).
3. Click the **"Parent"** column header to sort in Ascending order.
4. Observe the order of the rows in the "Parent" column.
5. Click the header again to sort in Descending order.

Expected result:

1. **Ascending:** Empty parents (-) should typically appear first (or last) as a group, followed by parent names A-Z.
2. **Descending:** The order should reverse.
3. In all cases, identical parent values should be grouped together.

Actual result:

1. The row order does not change.
2. The list likely remains sorted by ID, resulting in "Parent" values being scattered (e.g., an Empty parent row, followed by a Named parent row, followed by another Empty parent row).

Logs if exist:

*Test Log**:** AssertionError: expected [Array(10)] to deeply equal [Array(10)] (UI order did not match calculated alphabetic order).*

Project: qa-training-app

Bug ID: BUG009 (API_TC_36)

Date Reported: 2026-02-07

Reported by: 215131E

Assigned to: Developer

Status: New

Priority: Medium

Severity: Major

Component: API (Admin)

Summary: [API – Categories – Update] Updating a category name to null causes 500 Internal Server Error instead of 400 Bad Request

Description:

When an admin attempts to update a category using the PUT /api/categories/{id} endpoint with "name": null, the API should enforce the 'Not Null' constraint and return a 400 Bad Request. Instead, the API sometimes responds with a 500 Internal Server Error, indicating a server-side failure while committing the transaction.

Steps to reproduce:

1. Ensure an admin token is available.
2. Ensure a valid Category ID exists.
3. Send PUT /api/categories/{id} with the body:

   ```
   { "name": null,  "parentId": null }
   ```
4. Observe the API response.

Expected result:

1. The API should return 400 Bad Request with a message: "Name is required".

Actual result:

1. The API returned 500 Internal Server Error instead of 400 Bad Request.
2. The error message indicates a server-side failure while committing the transaction.

Logs if exist:

Test Log: Status 400 Bad Request; Error "Name is required" (expected)

Observed: 500 Internal Server Error

Project: qa-training-app

Bug ID: BUG010 (API_TC_37)

Date Reported: 2026-02-07

Reported by: 215131E

Assigned to: Developer

Status: New

Priority: Medium

Severity: Major

Component: API (Admin)

Summary: [API – Categories – Update] Updating a category name to an existing name allows duplicates instead of returning an error

Description:

When an admin attempts to update a category to a name that already exists (e.g., renaming Category "B" to "A"), the API should enforce unique name constraints and return a 400 Bad Request or 409 Conflict. Instead, the API successfully processes the request, allowing duplicate category names and returning 200 OK without any validation error.

Steps to reproduce:

1. Ensure categories "A" and "B" exist.
2. Send PUT /api/categories/{id_of_B} with the body:
   ```
   { "name": "A" }
   ```
3. Observe the API response.

Expected result:

1. The API should reject the update and return 400 Bad Request or 409 Conflict with a message like "Name already exists".

Actual result:

1. The API returned 200 OK and allowed the category name to be updated to a duplicate value.
2. No validation error message was displayed.

Logs if exist:

Test Log: Status 400 or 409 Conflict; Error "Name already exists" (expected)

Observed: Status 200 OK, category updated successfully

Project: qa-training-app

Bug ID: BUG011 (API_TC_40)

Date Reported: 2026-02-07

Reported by: 215131E

Assigned to: Developer

Status: New

Priority: Medium

Severity: Major

Component: API (User)

Summary: [API – Categories – Sorting] Sorting by Name in Ascending order returns Descending order

Description:

When a user requests the category list sorted by Name in Ascending order (A → Z), the API returns the list in Descending order (Z → A). The response status is 200 OK, but the data order does not match the requested sort direction.

Steps to reproduce:

1. Ensure a valid User Token is available.
2. Ensure multiple categories exist (A–Z).
3. Send GET /api/categories/page?sort=name,asc with the User Token.
1. Observe the order of the returned categories.

Expected result:

1. The API should return the list in Ascending order by Name (A → Z).

Actual result:

2. The API returned the list in Descending order (Z → A).

Logs if exist:

Test Log: Status 200 OK; List returned in Descending order despite requesting Ascending order

---

Project: qa-training-app

Bug ID: BUG012 (API_TC_41)

Date Reported: 2026-02-07

Reported by: 215131E

Assigned to: Developer

Status: New

Priority: Medium

Severity: Major

Component: API (User)

Summary: [API – Categories – Sorting] Sorting by Parent Category in Ascending order returns Descending order

Description:

When a user requests the category list sorted by Parent Category in Ascending order, the API returns the data in Descending order. Categories are not properly grouped or arranged by Parent ID/Name as expected.

Steps to reproduce:

1. Ensure a valid User Token is available.
2. Send GET /api/categories/page?sort=parent,asc with the User Token.
3. Observe the order and grouping of categories in the response.

Expected result:

1. Categories should be grouped and sorted in Ascending order by Parent ID/Name.

Actual result:

1. Categories were returned in Descending order, and the grouping did not match the requested ascending sort.

Logs if exist:

Test Log: Status 200 OK; List sorted in Descending order instead of Ascending

---

Project: qa-training-app

Bug ID: BUG013 (API_TC_42)

Date Reported: 2026-02-07

Reported by: 215131E

Assigned to: Developer

Status: New

Priority: Medium

Severity: Major

Component: API (User)

Summary: [API – Categories – Sorting] API does not reject invalid sort fields

Description:

When a user sends a GET request with an invalid sort field (e.g., invalid_col), the API should reject the request and return 400 Bad Request with an error message "Invalid sort field". Instead, the API processes the request successfully and defaults to sorting by Name (200 OK).

Steps to reproduce:

1. Ensure a valid User Token is available.
2. Send GET /api/categories/page?sort=invalid_col with the User Token.
3. Observe the API response.

Expected result:

1. The API should return 400 Bad Request with the message "Invalid sort field".

Actual result:

1. The API returned 200 OK and automatically sorted the data by Name, ignoring the invalid sort field.

Logs if exist:

Test Log: Status 200 OK; Data sorted by Name despite invalid sort parameter

---

Project: qa-training-app

Bug ID: BUG014 (API_TC_43)

Date Reported: 2026-02-07

Reported by: 215131E

Assigned to: Developer

Status: New

Priority: Medium

Severity: Major

Component: API (User)

Summary: [API – Categories – Sorting] API fails to handle multiple sort parameters correctly

Description:

When a user requests complex sorting (e.g., sort by Parent ascending, then Name descending), the API should respect all specified sort directions. However, the returned data does not follow the requested sort order, causing the Parent and Name hierarchy to be reversed.

Steps to reproduce:

1. Ensure a valid User Token is available.
2. Send GET /api/categories/page?sort=parent,asc&sort=name,desc with the User Token.
3. Observe the order of categories in the response.

Expected result:

1. Categories should be sorted first by Parent ascending, then by Name descending.

Actual result:

1. The API returned categories in an incorrect order, reversing the Parent and Name hierarchy.

Logs if exist:

Test Log: Status 200 OK; Data not sorted according to multiple parameters

---

2. Sales

Project: qa-training-app

Bug ID: BUG015 (UI_TC_21)

Date Reported: 2026-02-06

Reported by: 215023B

Assigned to: Developer

Status: New

Priority: Low

Severity: Minor

Component: UI / Sales Management

Summary: {UI – Sales - Sell Plant] Validation message mismatch; System uses native HTML5 error instead of required custom message.

Description:

The "Sell Plant" form correctly prevents the submission of quantities less than 1. However, the implementation uses the native HTML5 attribute min="1". This causes the browser to display a generic error message ("Value must be greater than or equal to 1") instead of the project-specific validation message defined in the requirements ("Quantity must be greater than 0").

Steps to reproduce:

1. Navigate to the "Sell Plant" page.
2. Select a plant and enter "0" or "-1" in the Quantity field.
3. Observe the validation message triggered upon clicking "Sell".

Expected result:

1. System prevents form submission for invalid quantities
2. System displays "Quantity must be greater than 0" error message in red below the field.

Actual result:

1. System prevents form submission for invalid quantities

2. When submitting the form with quantity -1, System displays "Value must be greater than or equal to 1" error message in red below the field.

Technical Note:

The input field is currently implemented with <input type="number" min="1">. To meet the specific requirements, a custom validation handler (e.g., setCustomValidity or a framework-level validation) should be used to override the default browser text.

Logs if exist:

"Error

No validation error message found on the page "

Project: qa-training-app

Bug ID: BUG016 (API_TC_14)

Date Reported: 2026-02-06

Reported by: 215023B

Assigned to: Developer

Status: New

Priority: Medium

Severity: Major (Security/Stability Risk)

Component: API / Sales Management

Summary: POST /api/sales/plant returns 500 Internal Server Error for non-numeric quantities.

Description:

When sending a request to the "Sell Plant" endpoint with a string value (e.g., "five") provided for the quantity parameter, the API crashes with a 500 error. The system should perform type validation and return a 400 Bad Request to indicate a client-side data type mismatch.

Steps to reproduce:

1. Send a POST request to /api/sales/plant/{plantId}.

2. Input a valid integer for plantId.
3. Input a string (e.g., "five") for the quantity query parameter.

Expected result:

1. API endpoint is correctly targeted.
2. Plant ID is accepted as a valid path variable.
3. System returns status code 400 Bad Request due to data type mismatch.

Actual result:

1. API endpoint is correctly targeted.
2. Plant ID is accepted as a valid path variable.
3. The system returned 500 Internal Server Error instead of the expected 400 Bad Request.

Logs if exist:

Response Body:

```
{"status":500,"error":"INTERNAL_SERVER_ERROR","message":"Failed to convert value of type 'java.lang.String' to required type 'int'; For input string: \"five\"","timestamp":"2026-02-06T22:55:00.7699166"}
```

Project: qa-training-app

Bug ID: BUG017 (API_TC_15)

Date Reported: 2026-02-03

Reported by: 215023B

Assigned to: Developer

Status: New

Priority: High

Severity: Major (Security/Stability Risk)

Component: API / Sales Management

Summary: API returns 500 Internal Server Error for missing path variables and query parameters.

Description:

The POST /api/sales/plant/ endpoint fails to utilize standard HTTP status codes for client-side errors. When mandatory fields are missing, the server undergoes an internal crash (500) and exposes raw Java exception details in the response body.

Steps to reproduce:

1. Send a POST request to /api/sales/plant/{plantId}.
2. Leave the quantity query parameter empty or null.
3. Leave the plantid path variable empty or null.

Expected result:

1. Send a POST request to /api/sales/plant/{plantId}.
2. Leave the quantity query parameter empty or null.
3. System returns status code 400 Bad Request.
4. Leave the plantid path variable empty or null.
5. System returns status code 404 Not Found

Actual result:

1. Send a POST request to /api/sales/plant/{plantId}.
2. Leave the quantity query parameter empty or null.
3. System returns status code 500 Internal Server Error instead of the expected 400 Bad Request.
4. Leave the plantId path variable empty or null.
5. System returns status code 500 Internal Server Error instead of the expected 404 Not Found.

Logs if exist:

logTesting: PlantId=missing, Quantity=missing

logURL: http://localhost:8080/api/sales/plant/requestPOST 500 /api/sales/plant/

logResponse: 500 - {"status":500,"error":"INTERNAL_SERVER_ERROR","message":"No static resource api/sales/plant.","timestamp":"2026-02-07T11:50:54.6380826"}

Project: qa-training-app

Bug ID: BUG018 (API_TC_16)

Date Reported: 2026-02-03

Reported by: 215023B

Assigned to: Developer

Status: New

Priority: Medium

Severity: Major

Component: API / Sales Management

Summary:

API returns 500 Error when the plantId path variable is provided as a string ("null").

Description:

The system fails to gracefully handle data type mismatches for path variables. When a string value like "null" is passed into the plantId segment of the URI, the application throws an unhandled MethodArgumentTypeMismatchException (reported as a 500 error) instead of returning a standard 400 Bad Request.

Steps to reproduce:

1. Send a POST request to /api/sales/plant/{plantId}.
2. Input a floating-point number (e.g., 2.5) for the quantity parameter.

Expected result:

1. POST request reaches the API gateway.
2. System returns status code 400 Bad Request.
3. Response message indicates a data type mismatch or invalid parameter format.

Actual result:

1. Send a POST request to /api/sales/plant/{plantId}.
2. Leave the quantity query parameter empty or null.

3. System returns status code 500 Internal Server Error instead of the expected 400 Bad Request.
4. Leave the plantId path variable empty or null.
5. System returns status code 500 Internal Server Error instead of the expected 404 Not Found.

Logs if exist:

```
logResponse: 500 - {"status":500,"error":"INTERNAL_SERVER_ERROR","message":"Failed to convert value of type 'java.lang.String' to required type 'java.lang.Long'; For input string: "null"","timestamp":"2026-02-06T22:55:14.5942229"}
```

Project: qa-training-app

Bug ID: BUG019 (API_TC_20)

Date Reported: 2026-02-03

Reported by: 215023B

Assigned to: Developer

Status: New

Priority: High

Severity: Critical (Security Vulnerability)

Component: API / Sales Controller / Security

Summary: Unauthorized Access: Normal "User" roles can successfully create plant sales.

Description:

The POST /api/sales/plant/{plantId} endpoint lacks proper role-based access control (RBAC). A user with a standard ROLE_USER token is permitted to create sales records and modify inventory, which should be a restricted operation reserved for ROLE_ADMIN.

Steps to reproduce:

1. Obtain a **User Auth Token** (standard user role).
2. Identify a valid plantId.
3. Send a **POST** request to: /api/sales/plant/{plantId}?quantity=1.

4. Observe the HTTP status code and verify if a record was created.

Expected result:

- **Status Code:** 403 Forbidden.
- **System Action:** No sale record should be created; inventory stock must remain unchanged.

Actual result:

- **Status Code:** 201 Created.
- **System Action:** Sale ID 256 was created, and inventory was updated successfully.

Logs: API Response: {"id":256, "plant":{...}, "quantity":1, "totalPrice":100, "soldAt":"2026-02-07T10:15:49..."}AssertionError: expected 201 to equal 403

Project: qa-training-app

Bug ID: BUG020 (API_TC_21)

Date Reported: 2026-02-03

Reported by: 215023B

Assigned to: Developer

Status: New

Priority: High

Severity: Critical (Security Vulnerability)

Component: API / Sales Controller / Security

Summary: Broken Access Control: Standard "User" roles can delete sales records.

Description:

The DELETE /api/sales/{id} endpoint does not properly enforce role-based access control. Standard users are able to delete transaction records, leading to unauthorized data loss and potential audit trail manipulation. This operation must be restricted to ROLE_ADMIN.

Steps to reproduce:

1. Log in as a **Standard User** and obtain an Auth Token.
2. Identify an existing sales record ID (e.g., 257).

3. Send a **DELETE** request to: /api/sales/257.
4. Observe the HTTP status code and verify if the record still exists.

Expected result:

- **Status Code:** 403 Forbidden.
- **Database State:** The sales record must remain in the database.

Actual result:

- **Status Code:** 204 No Content.
- **Database State:** Sales record ID 257 was permanently deleted.

Logs: request DELETE 204 /api/sales/257

AssertionError: expected 204 to equal 403

Project: qa-training-app

Bug ID: BUG021 (API_TC_22)

Date Reported: 2026-02-07

Reported by: 215023B

Assigned to: Backend Developer

Status: New

Priority: Medium

Severity: Major (Information Exposure)

Component: API / Sales Controller

Summary: GET /api/sales/{id} returns 500 error instead of 400 when a string ID is provided.

Description:

The sales retrieval endpoint does not gracefully handle invalid data types for the {id} path variable. Providing an alphanumeric string (e.g., "ABC") causes an unhandled conversion exception, leading to a 500 Internal Server Error and the leakage of internal stack trace information.

Steps to reproduce:

1. Obtain a valid User Auth Token.
2. Send a **GET** request to: /api/sales/ABC.
3. Observe the HTTP status code and response body.

Expected result:

1. Connection to the endpoint is initiated with an invalid data type.
2. System returns status code 400 Bad Request due to invalid request format.
3. API error is handled as a standard validation alert.

Actual result:

1. A GET request was successfully sent to the endpoint with invalid data type.
2. The system returned 500 Internal Server Error instead of the expected 400 Bad Request.
3. The API exposed raw backend exception details (Failed to convert value of type 'java.lang.String'...) rather than a standard validation alert.

Logs: request GET 500 /api/sales/ABC

AssertionError: expected 500 to equal 400

Project: qa-training-app

Bug ID: BUG022 (API_TC_23)

Date Reported: 2026-02-07

Reported by: 215023B

Assigned to: Backend Developer

Status: New

Priority: Medium

Severity: Major (Information Exposure)

Component: API / Sales Controller

Summary: DELETE /api/sales/{id} returns 500 error instead of 400 when an alphanumeric string ID is used.

Description:

The delete endpoint does not properly validate the data type of the {id} path variable. Sending a non-numeric string (e.g., "XYZ") triggers an unhandled MethodArgumentTypeMismatchException at the server level, resulting in a 500 status code and the disclosure of backend implementation details.

Steps to reproduce:

1. Obtain a valid User Auth Token.
2. Send a **DELETE** request to: /api/sales/XYZ.
3. Observe the HTTP status code and response body.

Expected result:

1. Status Code: 400 Bad Request.
2. System Logic: The request should be rejected during validation, prior to any permission checks or database queries.

Actual result:

1. Status Code: 500 Internal Server Error.
2. Response Body: {"status":500, "error":"INTERNAL_SERVER_ERROR", "message":"Failed to convert value of type 'java.lang.String' to required type 'java.lang.Long'..."}

Logs: request DELETE 500 /api/sales/XYZ

AssertionError: expected 500 to equal 400

---

3. Plants

Project: qa-training-app

Bug ID: BUG022 (API_TC_25)

Date Reported: 2026-02-05

Reported by: 215107L

Assigned to: Developer

Status: New

Priority: Medium

Severity: Major

Component: API / Plant Management

Summary: API returns an incorrect validation message when plant name is missing during plant creation

Description: According to the defined validation rules, the **Plant Name** field has the following requirement:

- **Field:** Name
- **Rule:** Required
- **Expected Message:** *Plant name is required*

However, when a plant is created with a missing or empty name field, the API sometimes returns the **length validation message** instead of the required-field message.

This indicates that the validation logic is incorrectly triggering the **size constraint** before the **required constraint**, resulting in an inaccurate error message.

Steps to Reproduce

1. Authenticate as Admin and obtain a valid token.
2. Ensure a valid sub-category exists.
3. Send a POST request to /api/plants/category/{categoryId} with an empty or missing name field.
4. (Optional) Include "id": 0 in the request body and repeat the request.

Expected Results

1. API returns **400 Bad Request**

2. Validation message is **always**:

   “Plant name is required”

3. The required-field rule takes precedence over length validation

Actual Results

1. API returns **400 Bad Request**
2. Validation message returned in some cases is:

   “Plant name must be between 3 and 25 characters”

3. This message is incorrect for a missing/empty name field and contradicts the defined validation rule

Logs if exist