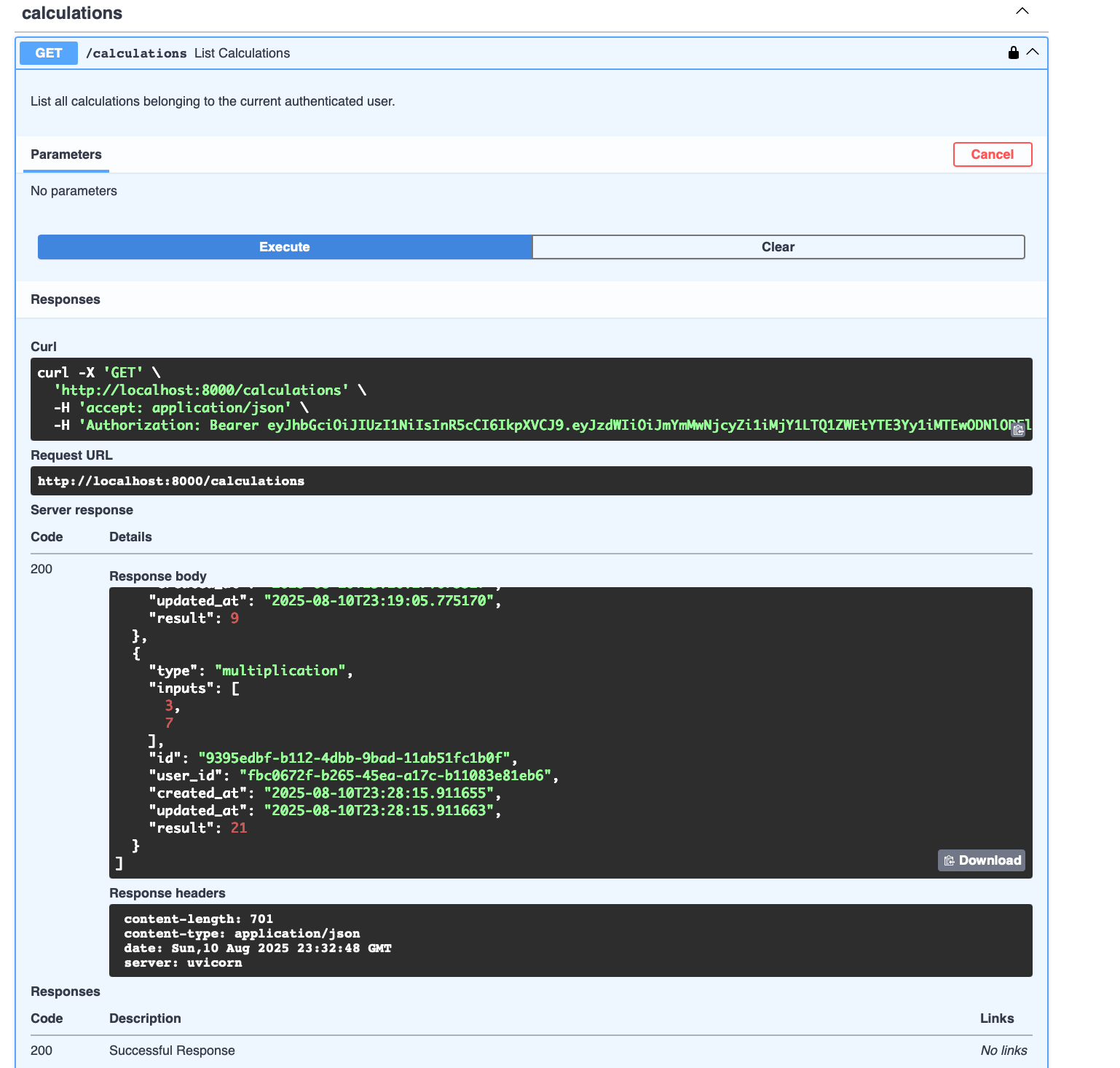
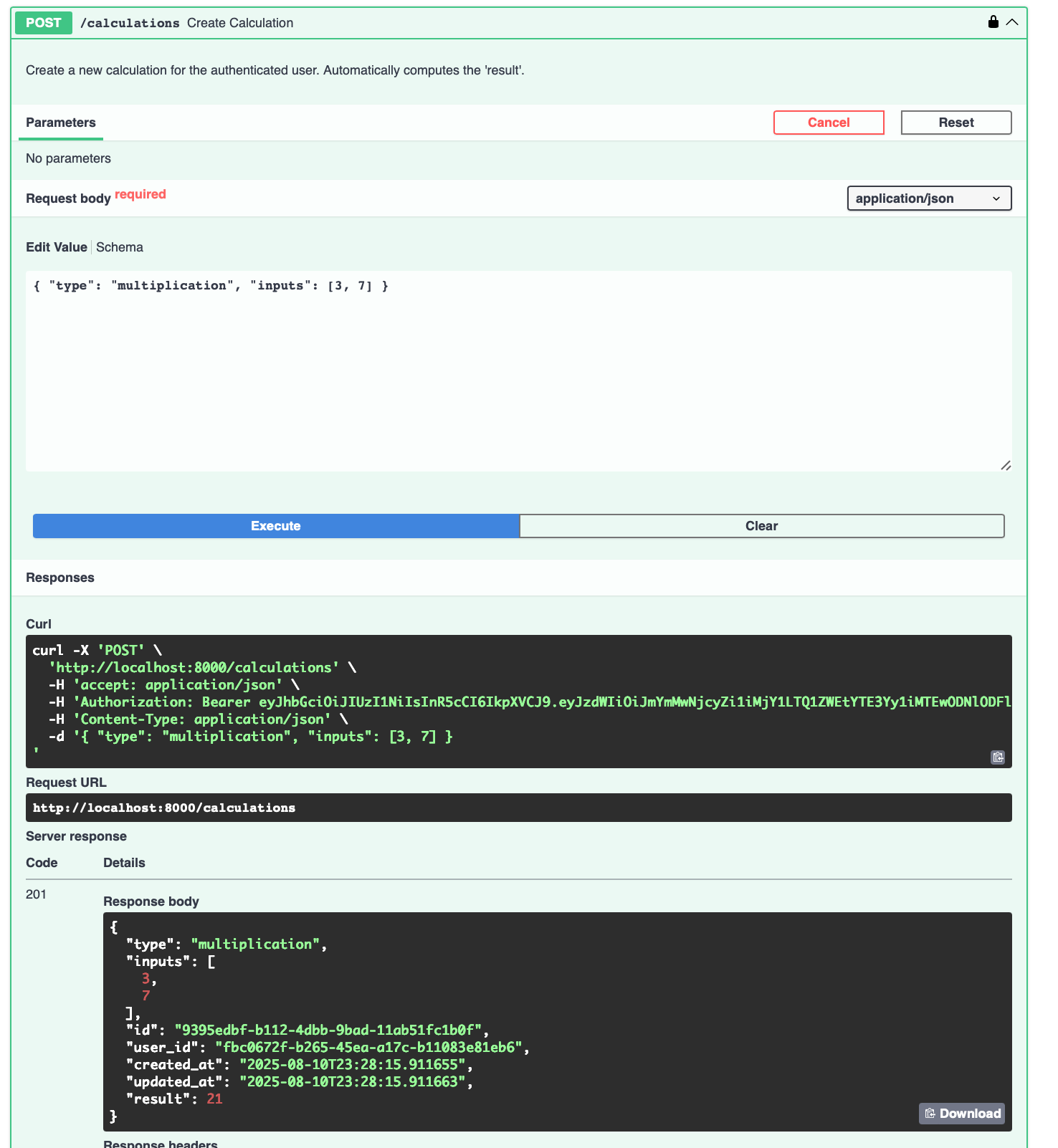
### **Screenshot 1 — Browse (GET /calculations)**

* **What I did:** Called the list endpoint as the logged-in user.
* **Status:** 200 OK
* **Proof:** Response body shows my calculations array (e.g., two items), including IDs and fields (type, inputs, result, user\_id).
* **Why it matters:** Confirms the **Browse** part of BREAD returns only the current user’s data.



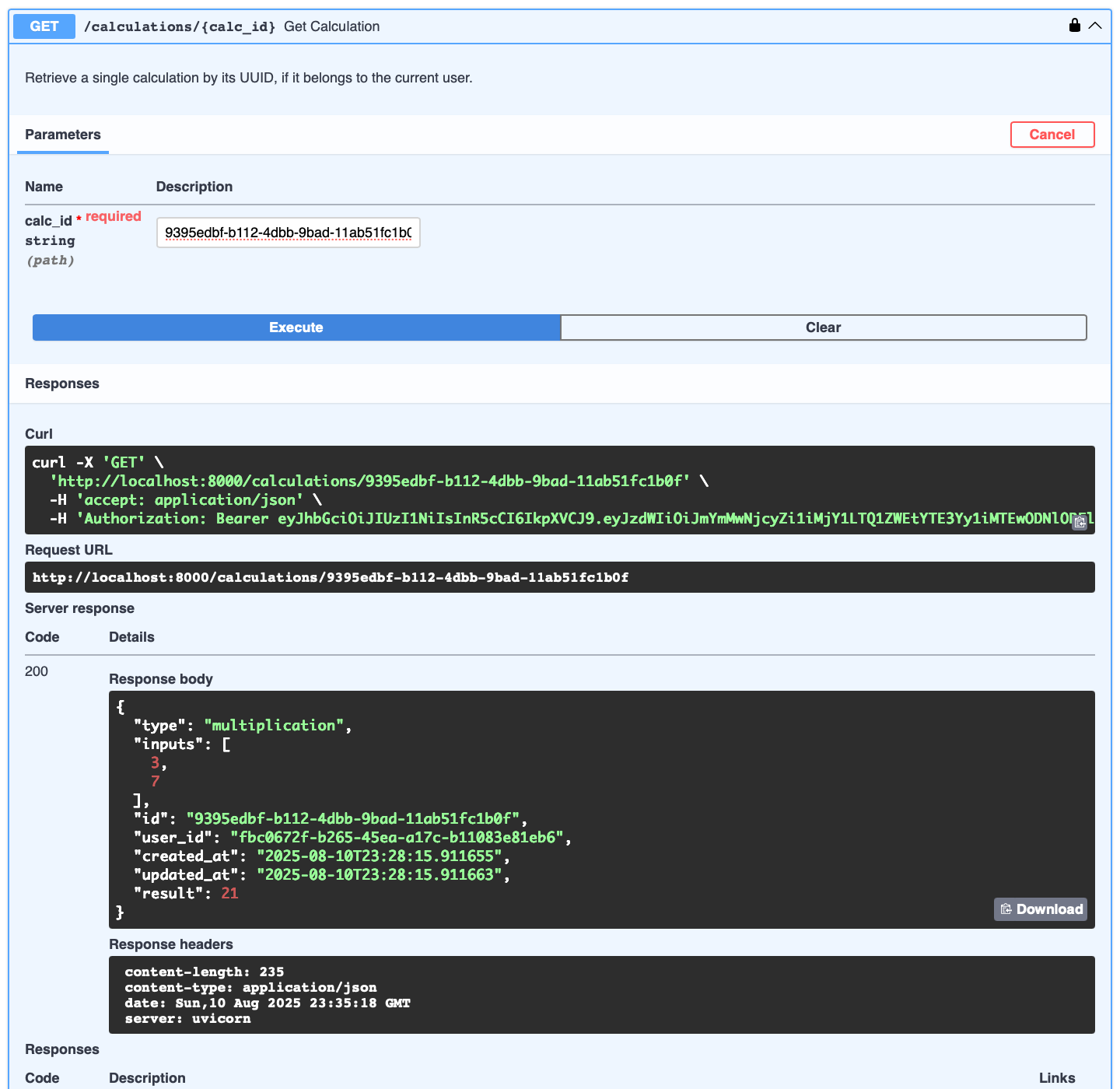
### **Screenshot 2 — Add / Create (POST /calculations)**

* **What I did:** Created a new calculation with body {"type":"multiplication","inputs":[3,7]}.
* **Status:** 201 Created
* **Proof:** Response contains the new record with ID **9395edbf-b112-4dbb-9bad-11ab51fc1b0f** and result: 21.
* **Why it matters:** Demonstrates the **Add** operation works and server computes/returns the result.



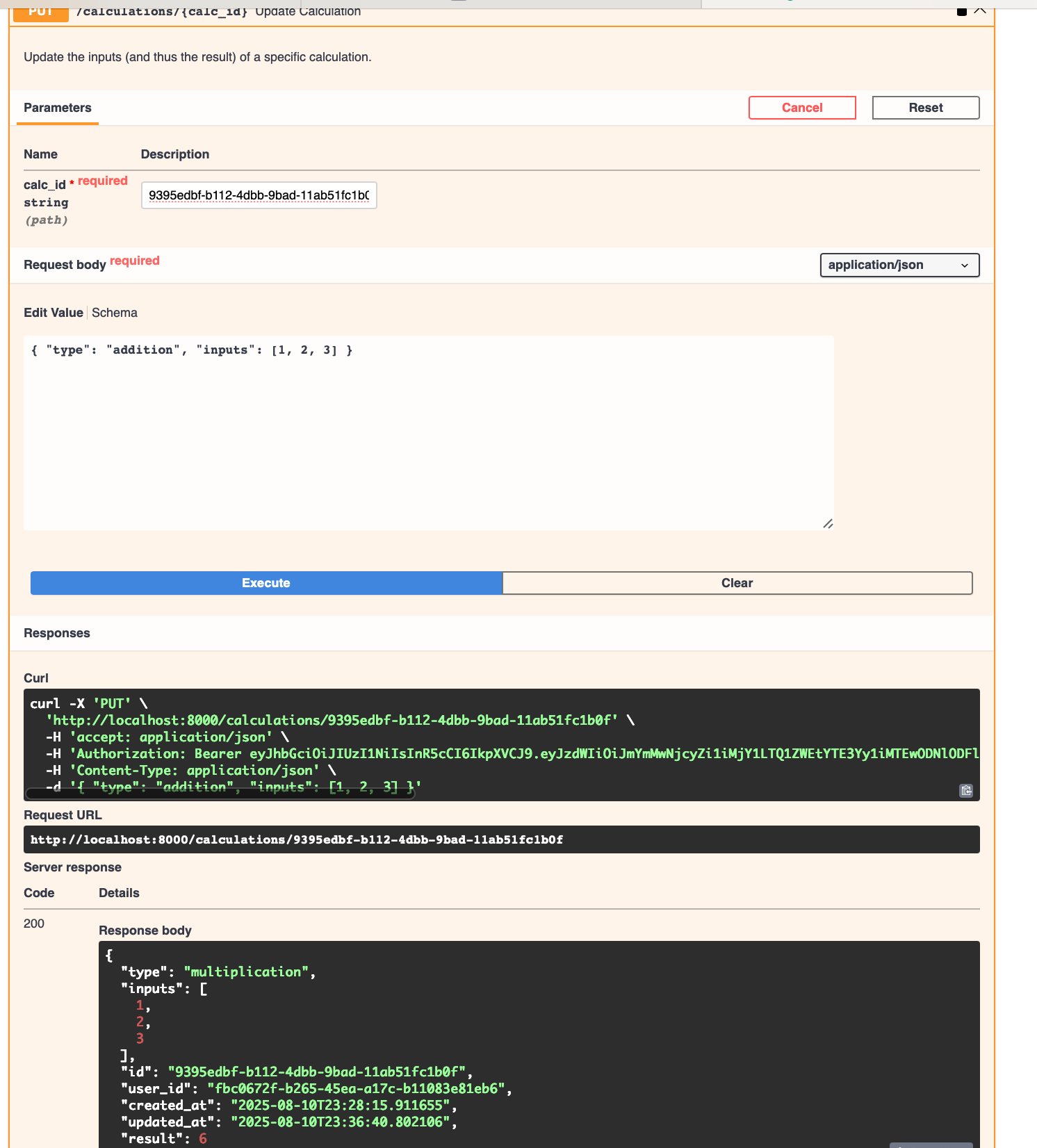
### **Screenshot 3 — Read (by id) (GET /calculations/{calc\_id})**

* **What I did:** Fetched the item I just created using its ID.
* **Status:** 200 OK
* **Proof:** Response shows type: "multiplication", inputs: [3, 7], result: 21, and the same ID **9395edbf-b112-4dbb-9bad-11ab51fc1b0f**.
* **Why it matters:** Confirms the **Read** part of BREAD returns the exact item by UUID.



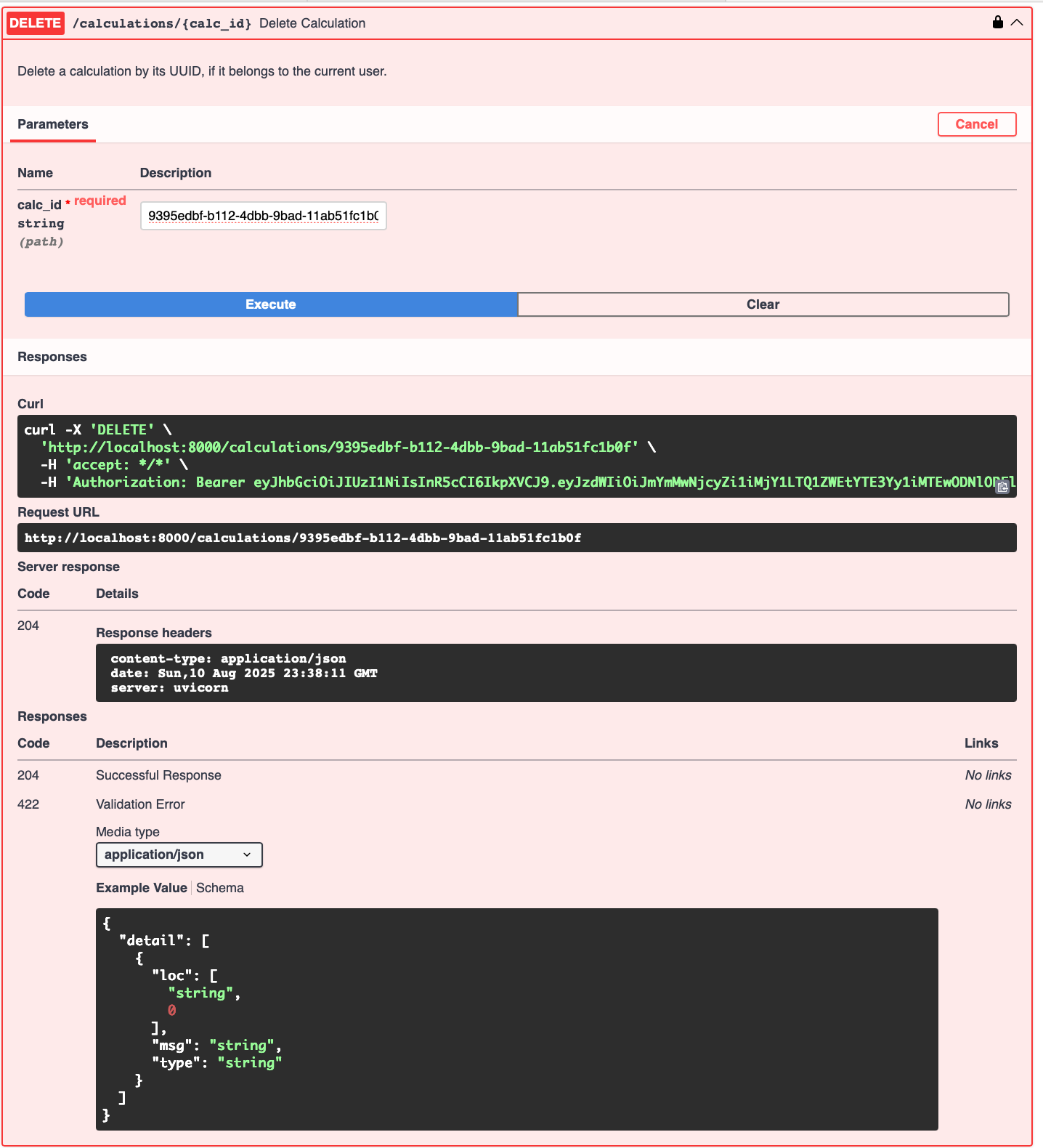
### **Screenshot 4 — Edit / Update (PUT /calculations/{calc\_id})**

* **What I did:** Updated the same calc with body {"type":"addition","inputs":[1,2,3]}.
* **Status:** 200 OK
* **Proof:** Response shows updated fields and result: 6 with a new updated\_at timestamp.
* **Why it matters:** Demonstrates the **Edit** operation saves changes and re-computes the result.



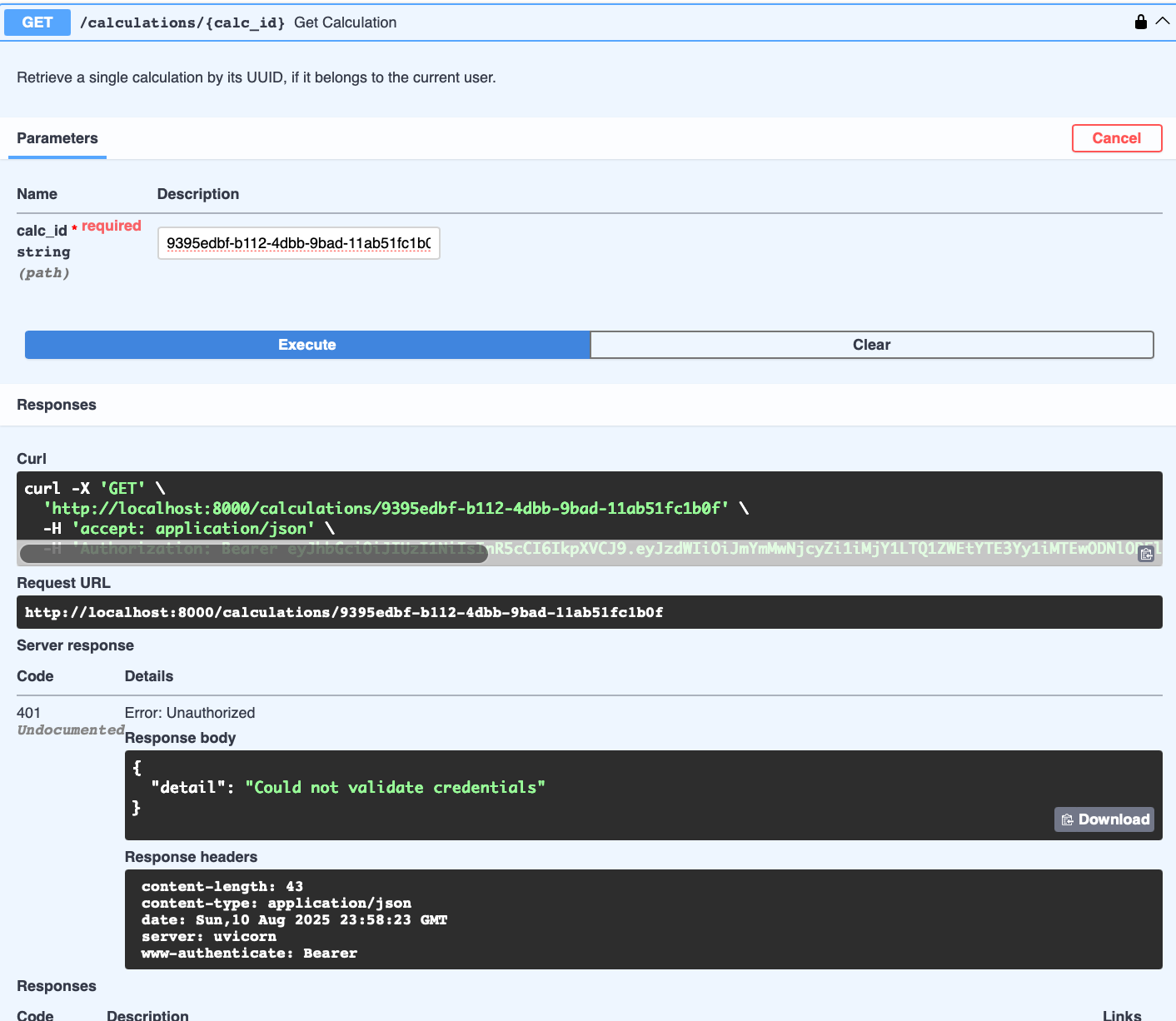
### **Screenshot 5 — Delete (DELETE /calculations/{calc\_id})**

* **What I did:** Deleted the item by its ID.
* **Status:** 204 No Content
* **Proof:** Empty body with 204; request URL includes the same item ID.
* **Why it matters:** Confirms the **Delete** operation succeeds.



### **Screenshot 6 — Read after Delete (GET /calculations/{calc\_id})**

* **What I did:** Tried to fetch the deleted item.
* **Status:** 404 Not Found
* **Proof:** Response body shows {"detail":"Calculation not found."}.
* **Why it matters:** Verifies the item is actually gone.



UI integration

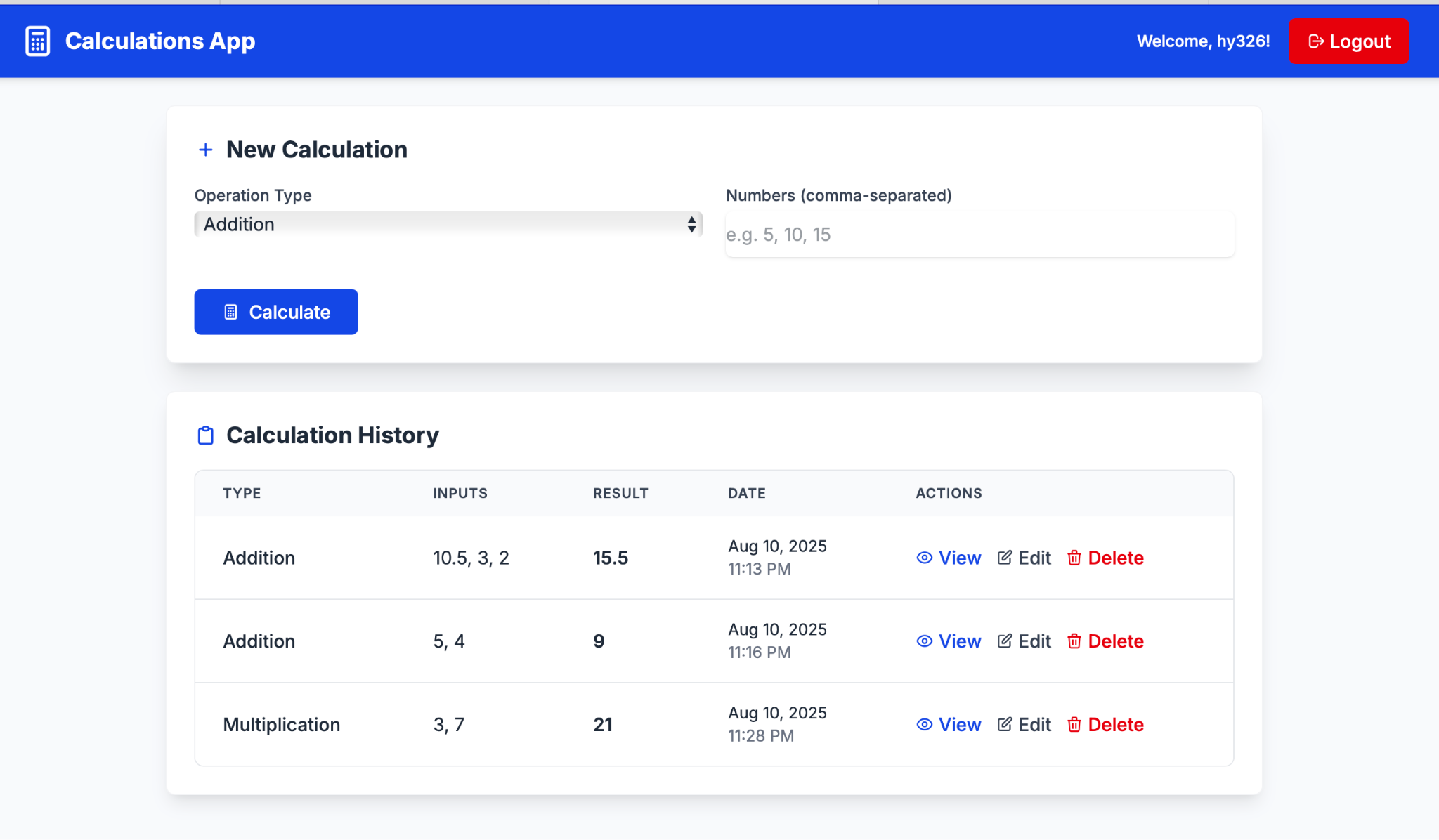
# **1) Logged in successfully (Loged in Succefully.png)**

**What this proves**

* Authenticated session is active (username shown + Logout button).
* All BREAD UI controls are gated behind login.
* The front-end is carrying a valid bearer token to use with API calls.

**Why it matters**

* Confirms the app is using the API’s auth flow and is ready to perform protected BREAD operations.



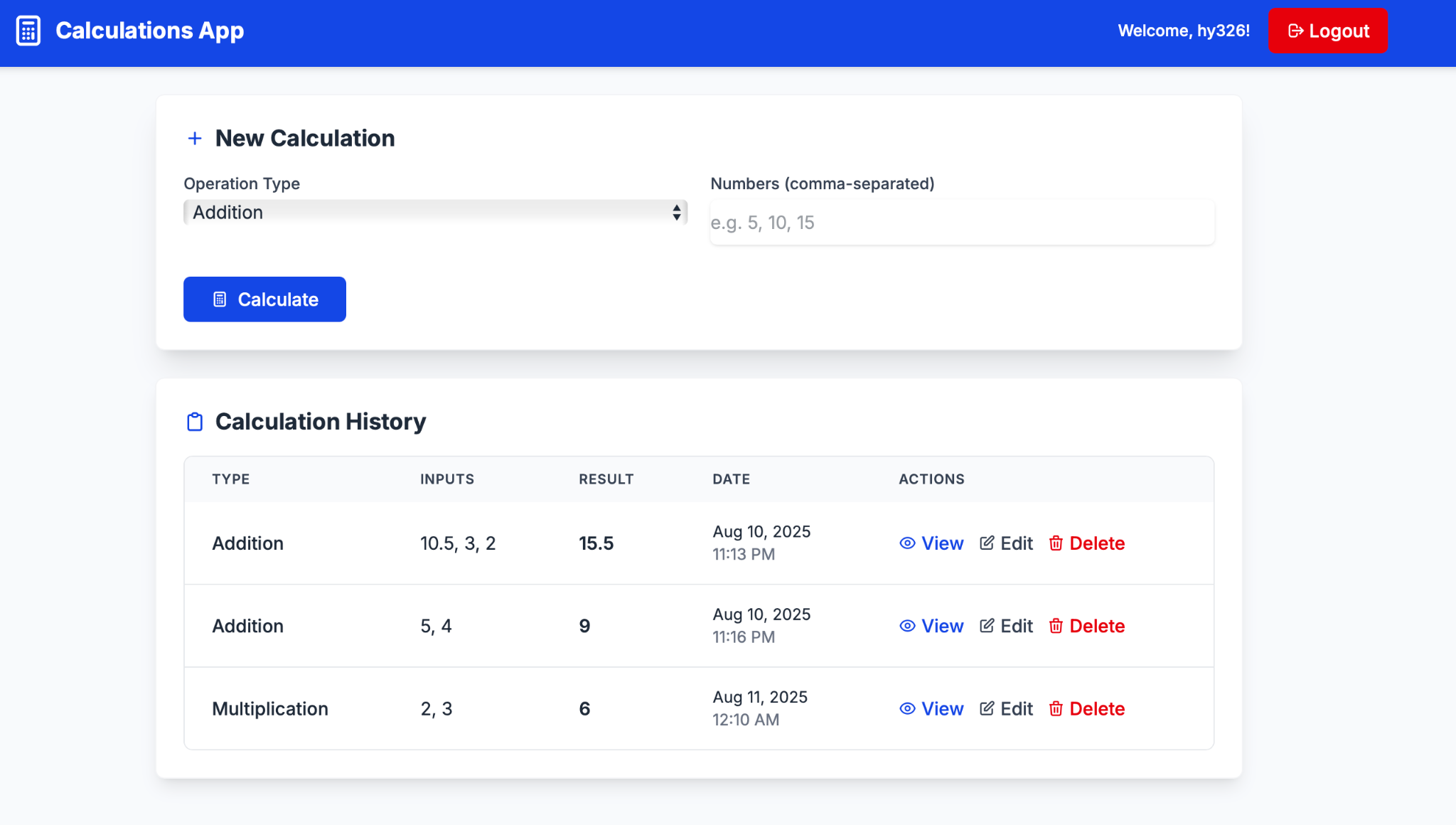
# **2) Browse / List view (list of transactions \_ History.png)**

**What this proves**

* **Browse (List)** is implemented: the “Calculation History” table shows all records for the current user.
* Each row shows **type**, **inputs**, **result**, **timestamps**, and action links (**View**, **Edit**, **Delete**).
* The **New Calculation** form is visible at the top (used for **Create**).

**Behind the scenes**

* Front-end sends GET /calculations with the bearer token and renders the returned JSON.



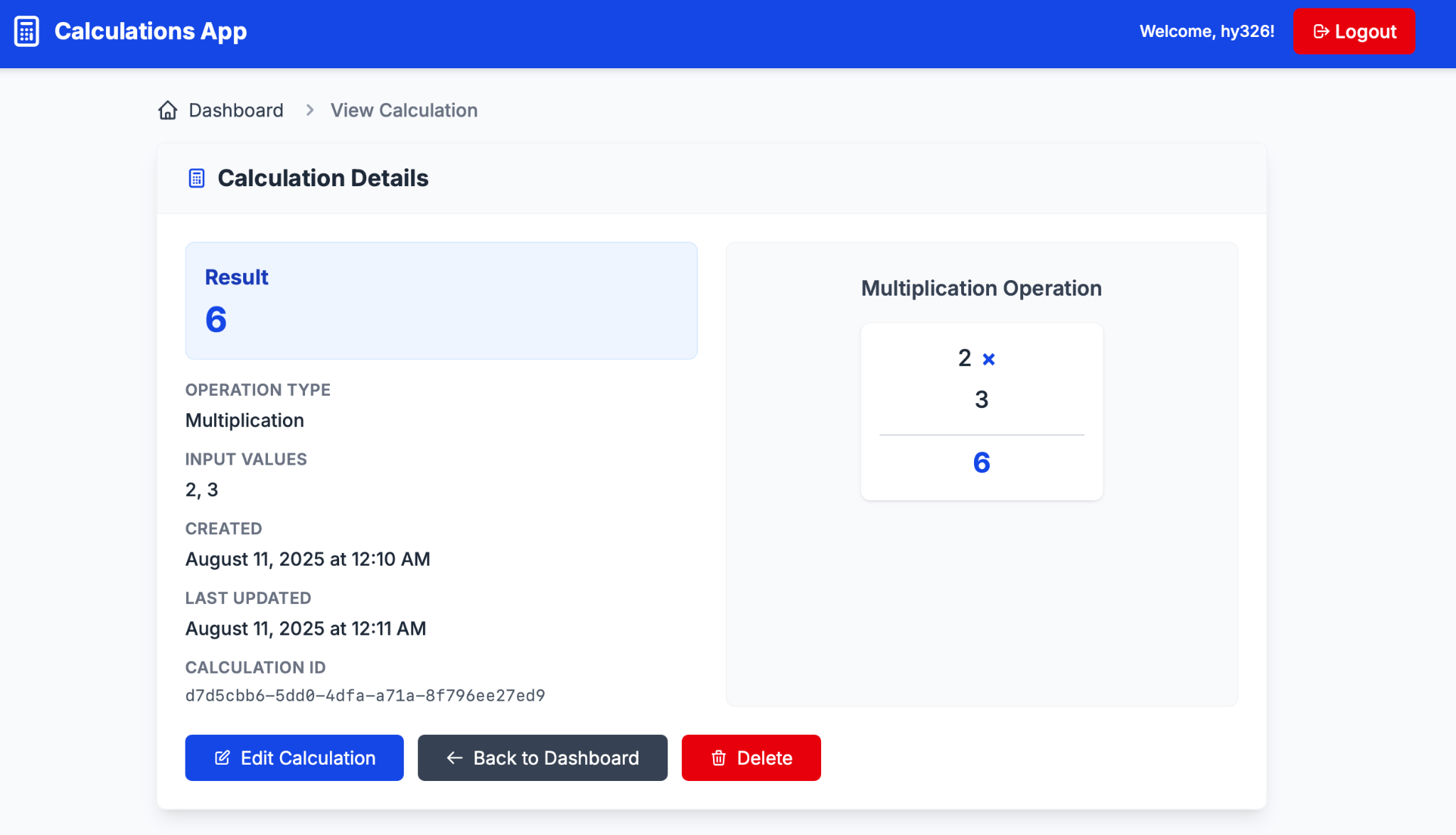
# **3) Read / Detail view (Browse\_works.png)**

**What this proves**

* **Read (View one)** is implemented: a dedicated “Calculation Details” page displays the record.
* Shows the result, operation type, input values, created/updated dates, and the calculation **ID**.
* Includes action buttons: **Edit Calculation**, **Back**, **Delete**.

**Behind the scenes**

* Front-end calls GET /calculations/{id} and renders the full record.
* Breadcrumbs and routing confirm deep-linking to a single resource.



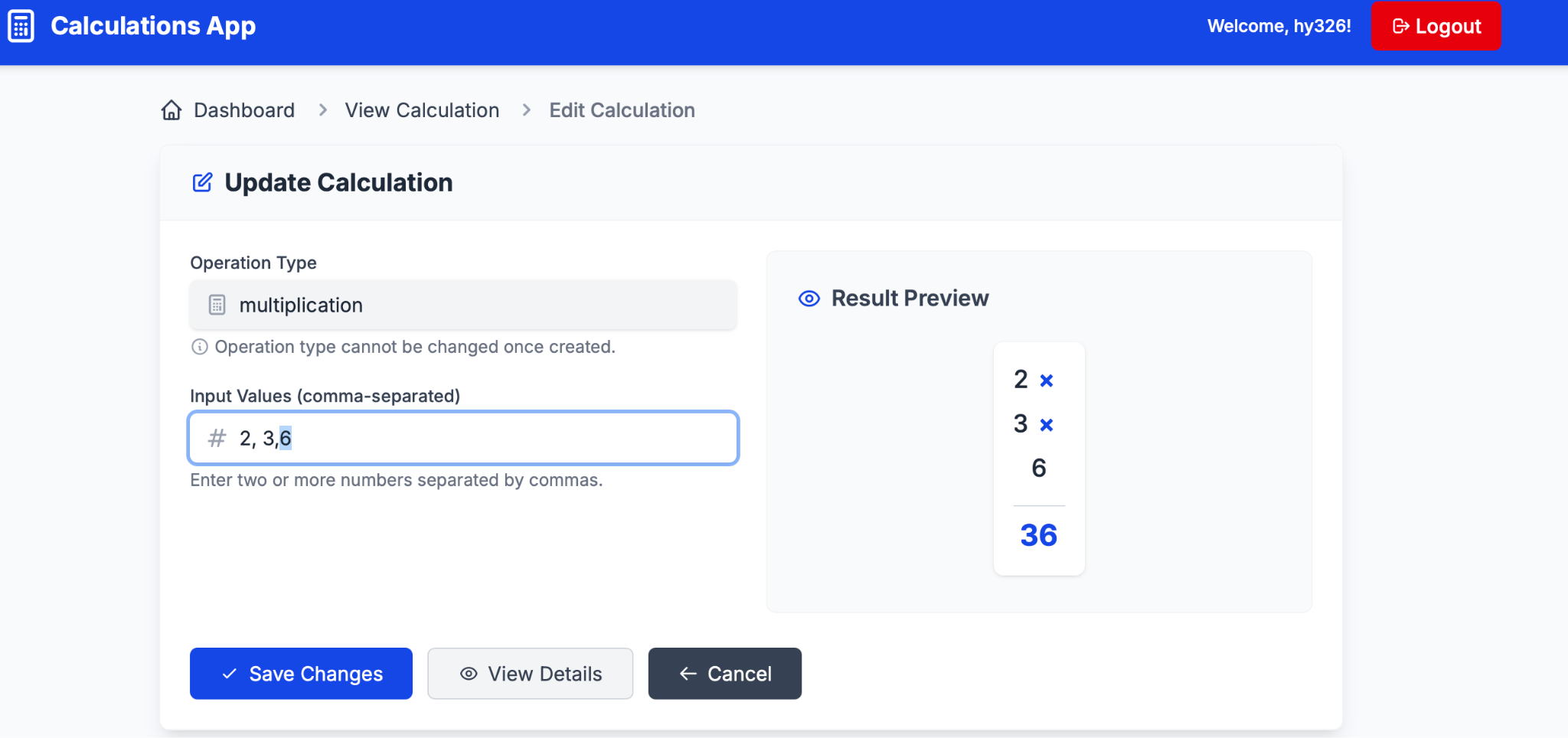
# **4) Edit form with validation & preview (Edit\_Works.png)**

**What this proves**

* **Edit (Update)** UI is implemented:
  + Operation type is locked (cannot change once created) — matches business rules.
  + Inputs field enforces **comma-separated numeric values** (client-side validation).
  + Live **Result Preview** updates before saving, confirming front-end logic matches API math.

**Behind the scenes**

* Front-end prepares a PUT /calculations/{id} payload (type stays fixed; inputs are updated).
* Validation prevents bad requests (non-numeric / empty input).



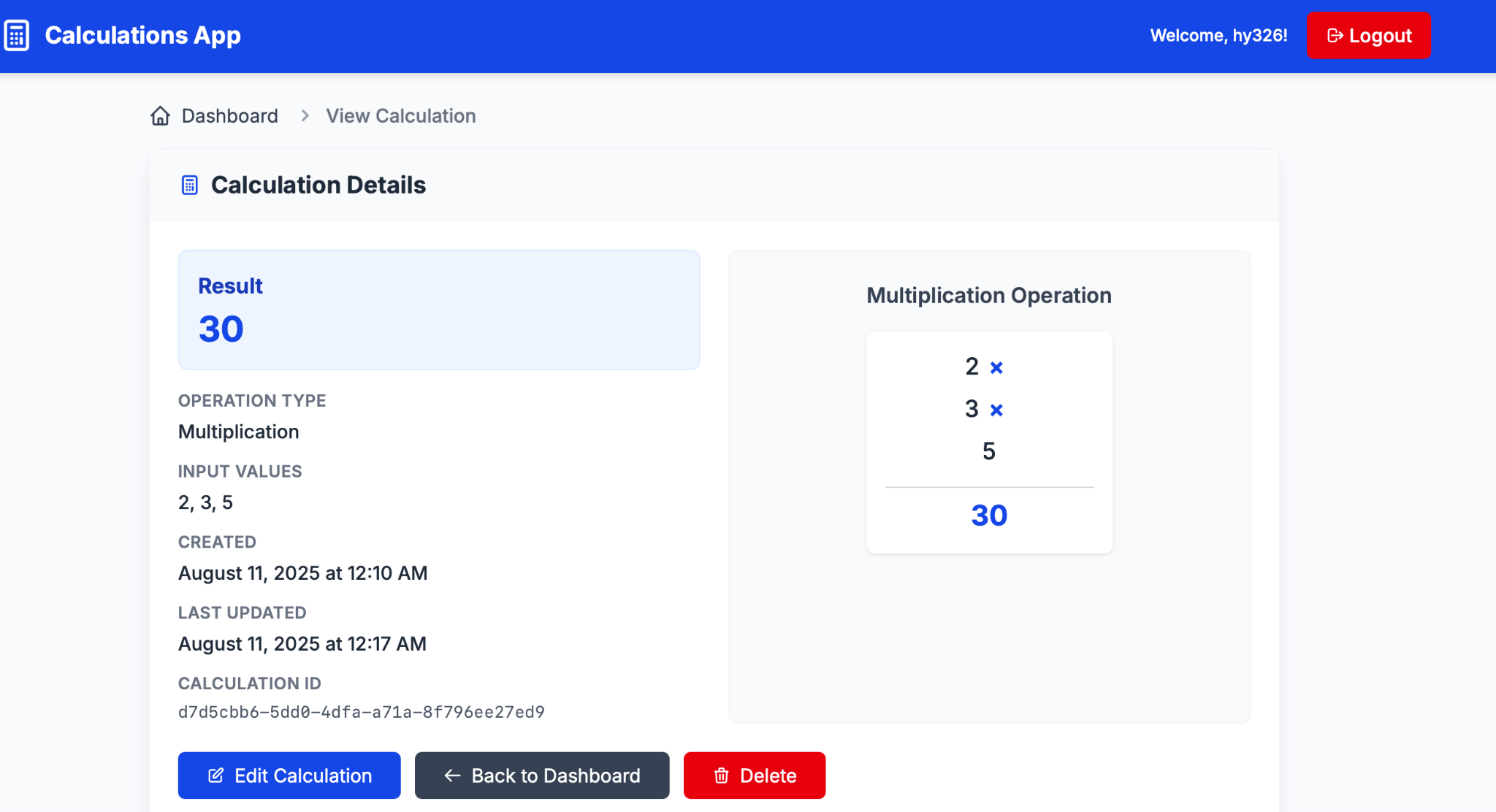
# **5) Edit saved (record updated) (Edit\_saved.png)**

**What this proves**

* **Update** round-trip succeeded:
  + Detail page reflects the new inputs and recomputed result.
  + “Last Updated” timestamp changed.
* Confirms the UI refreshes state after a successful PUT.

**Behind the scenes**

* Front-end sent PUT /calculations/{id}, then re-fetched or updated local state to show the new values.



# **6) Delete removed from list (Deleted Transaction\_removed.png)**

**What this proves**

* **Delete** is implemented:
  + After clicking **Delete** on the detail page, returning to the dashboard shows the item is gone.
  + The list now contains only the remaining records.

**Behind the scenes**

* Front-end sent DELETE /calculations/{id}, then refreshed GET /calculations (or pruned local state) to reflect the removal.

