**Time Capsule API — Testing Guide**

**Manual testing**

This section contains the full list of endpoints with sample requests, headers, and expected behaviours for testing via API clients like **Thunder Client**, or **Postman**.

# **Authentication**

1. **Register User**

* **Method:** POST
* **URL:** /auth/register

**Headers:**

Content-Type: application/json

**Body (raw JSON):**

{

"email": "user@example.com",

"password": "yourpassword"

}

**Expected Response:**

* 201 Created if successful
* 400 if user already exists

**Manual Testing Output:**

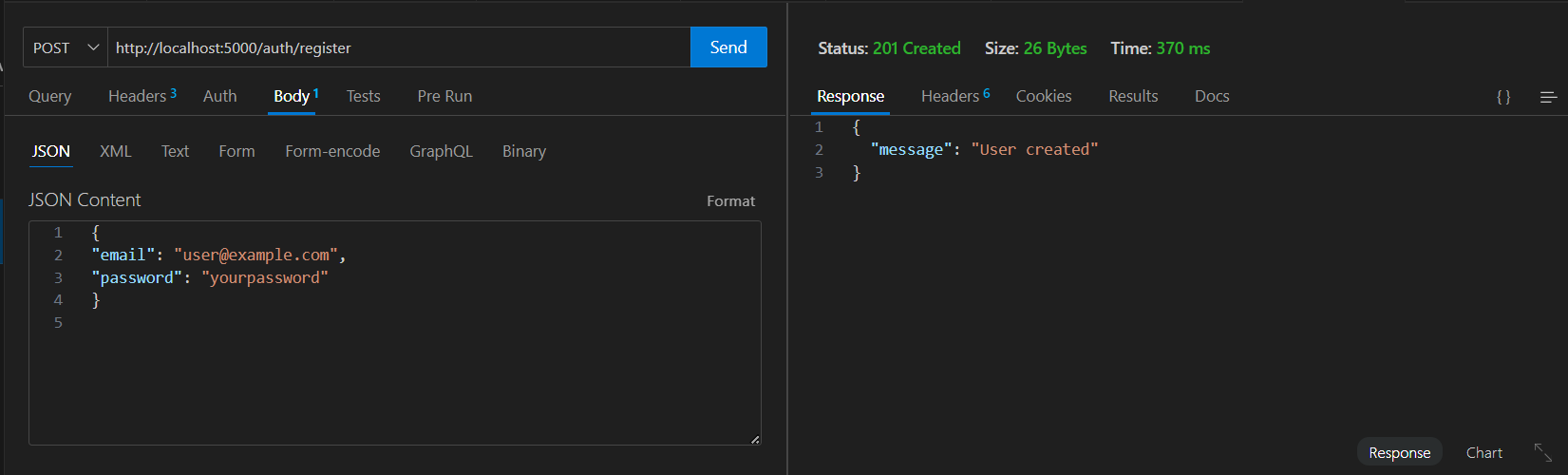


Figure 1: Successful user registration

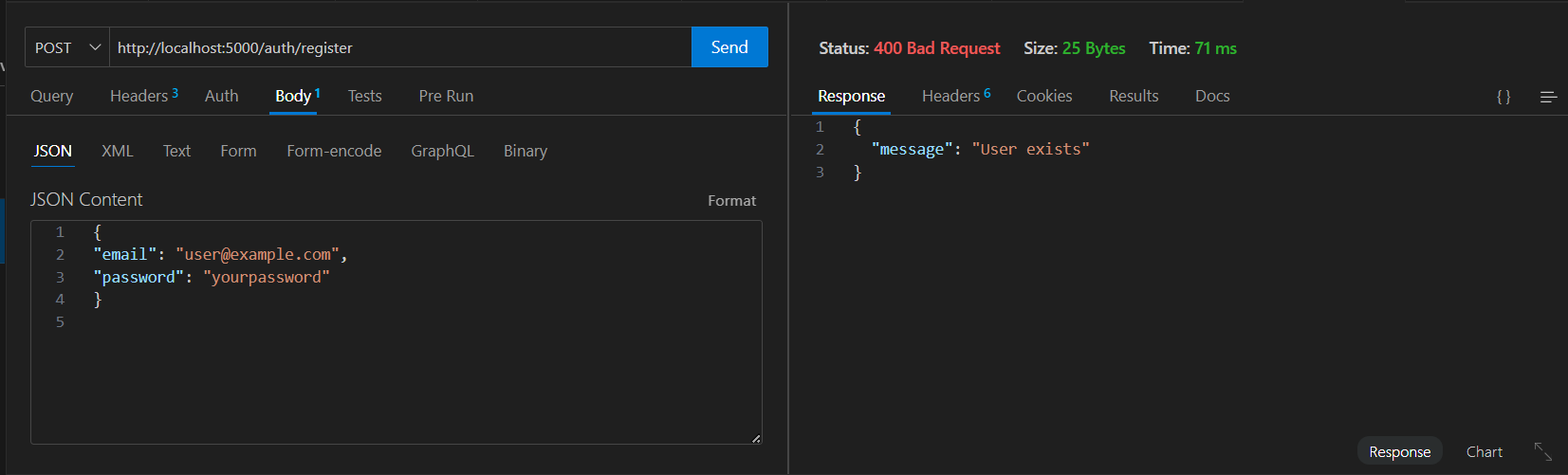
****

Figure 2: 400 error; user already exists

1. **Login User**

* **Method:** POST
* **URL:** /auth/login

**Body:**

{

"email": "user@example.com",

"password": "yourpassword"

}

**Expected Response:**

{

"token": "<JWT\_TOKEN>"

}

**Manual Testing Output:**

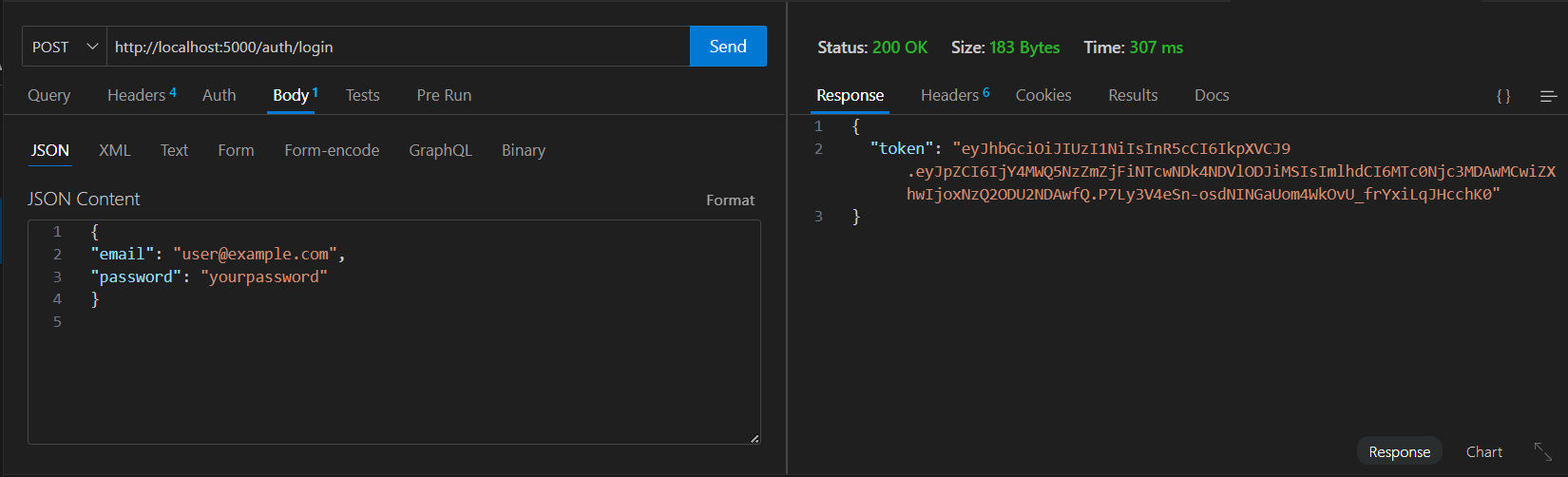
****

Figure 3: JWT Token is visible

# **Capsule Endpoints**

All capsule endpoints require the header:

Authorization: Bearer <JWT\_TOKEN>

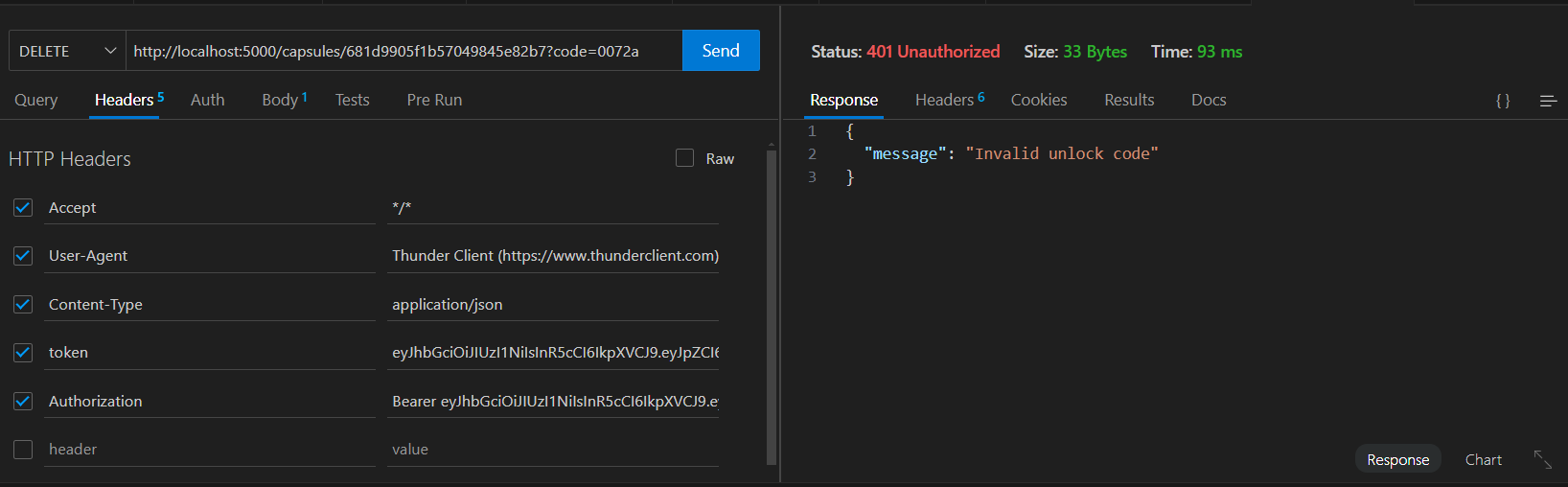


Figure 4: Require Headers

# **Create Capsule**

* **Method:** POST
* **URL:** /capsules

**Body:**

{

"message": "Hello future!",

"unlock\_at": "2025-12-31T23:59:59Z"

}

**Expected Response:**

{

"id": "<CAPSULE\_ID>",

"unlock\_code": "abc12345"

}

**Manual Testing Output:**

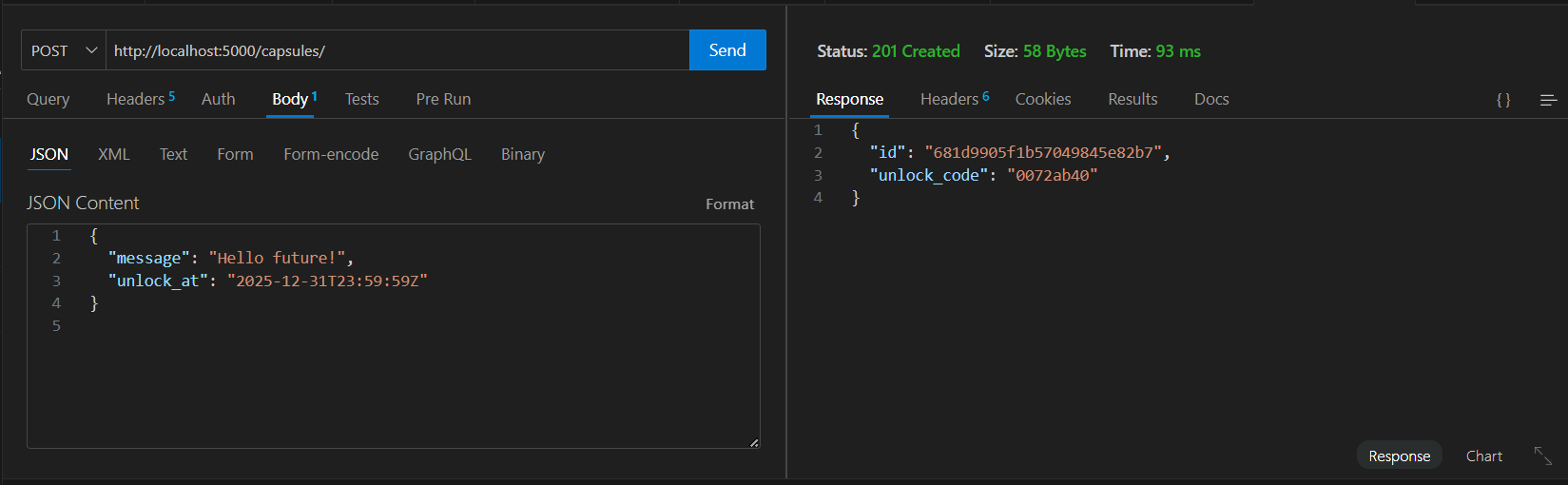
****

Figure 5: Capsule Created Successfully

# **Get Capsule (By ID & Unlock Code)**

* **Method:** GET
* **URL:** /capsules/:id?code=<UNLOCK\_CODE>

**Expected Responses:**

* 200 OK → Returns message if unlocked
* 401 Unauthorized → Wrong/missing code
* 403 Forbidden → Capsule still locked
* 410 Gone → Capsule expired

**Manual Testing Output:**

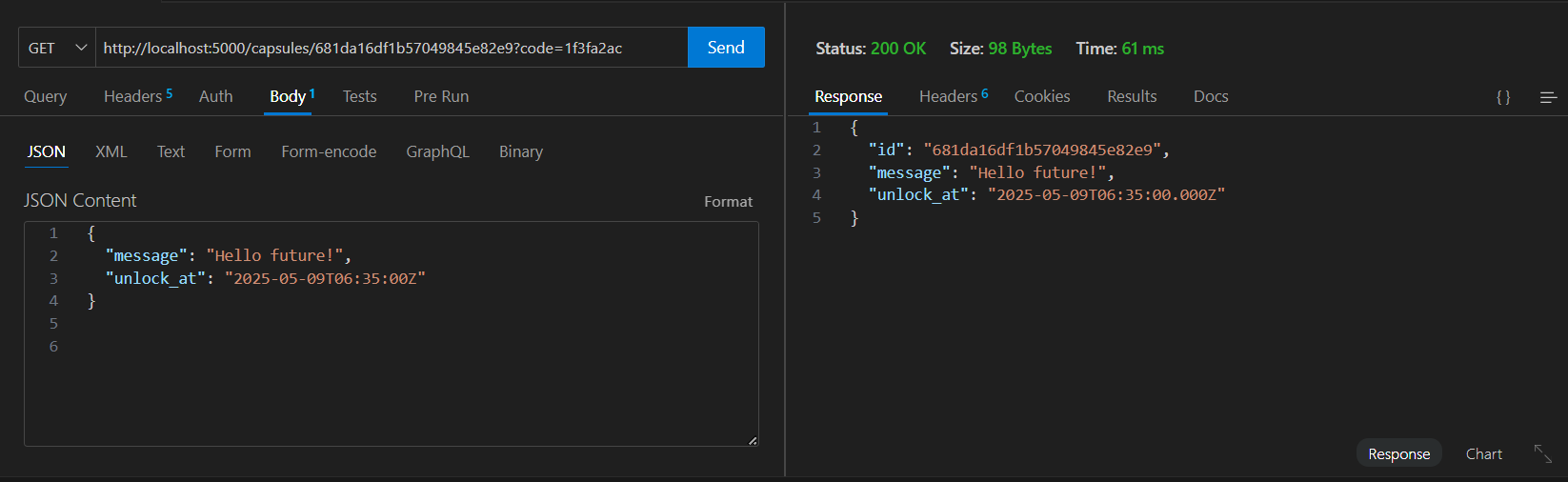
****

Figure 6:200 OK; Unlocked Capsule

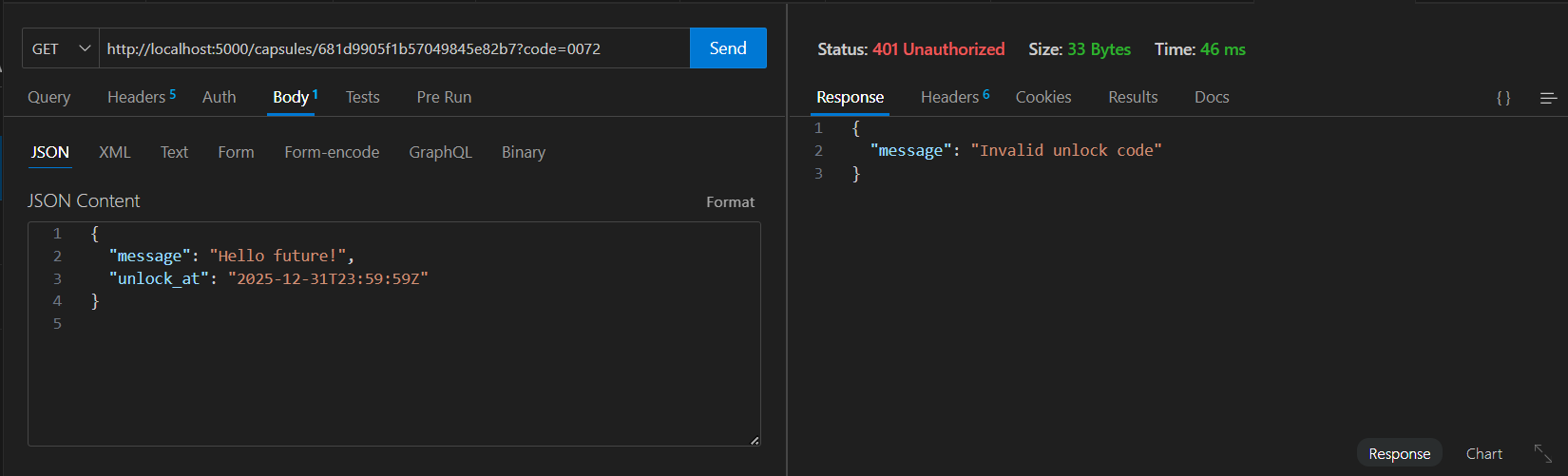
****

Figure 7: 401 Unauthorized; Wrong Code

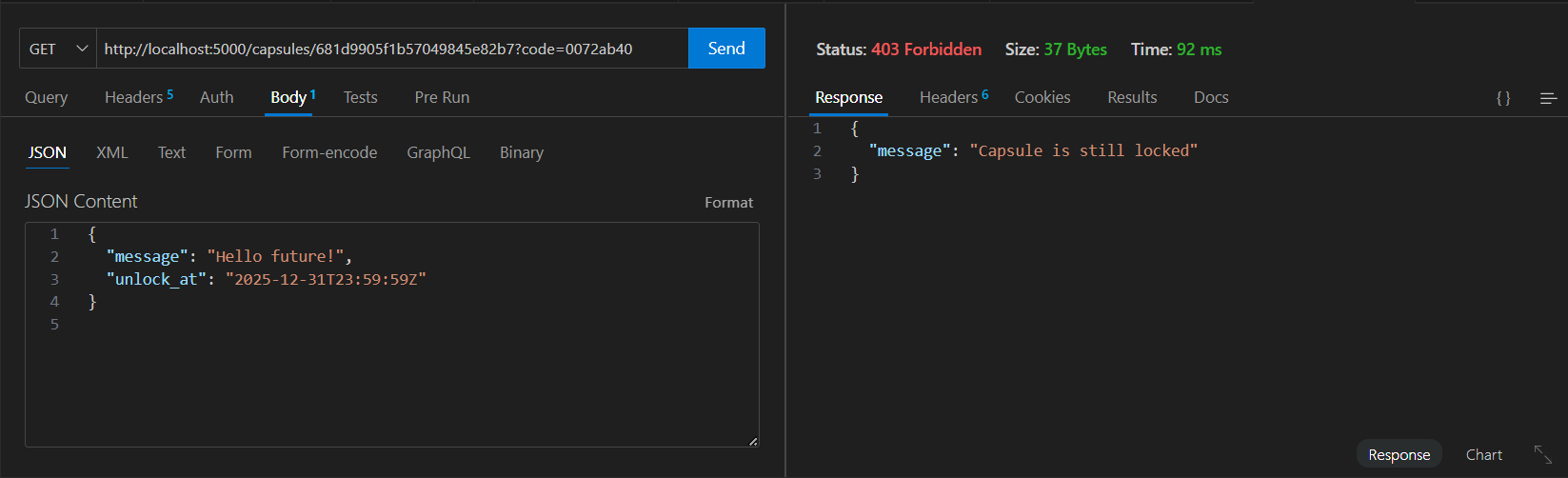
****

Figure 8: 403 Forbidden; Capsule is still locked

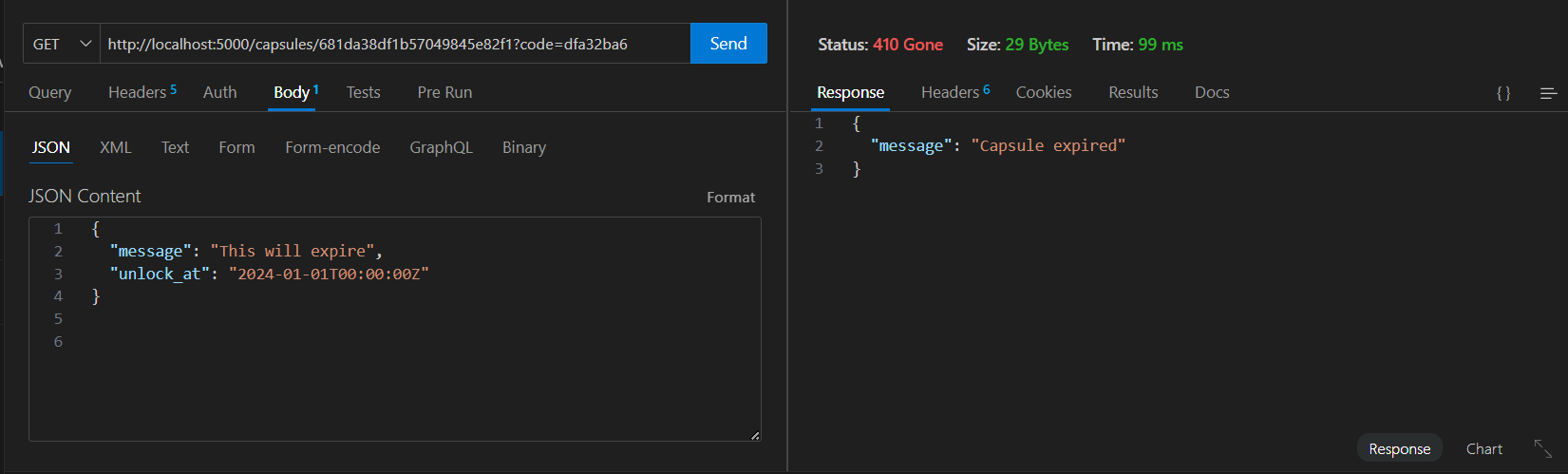
****

Figure 9: 401 Gone; Capsule is expired

# **List Capsules (Paginated)**

* **Method:** GET
* **URL:** /capsules?page=1&limit=5

**Expected Response:**

{

"page": 1,

"limit": 5,

"capsules": [

{

"id": "...",

"unlock\_at": "...",

"created\_at": "...",

"message": "..." // Only if unlocked

}

]

}

**Manual Testing Output:**

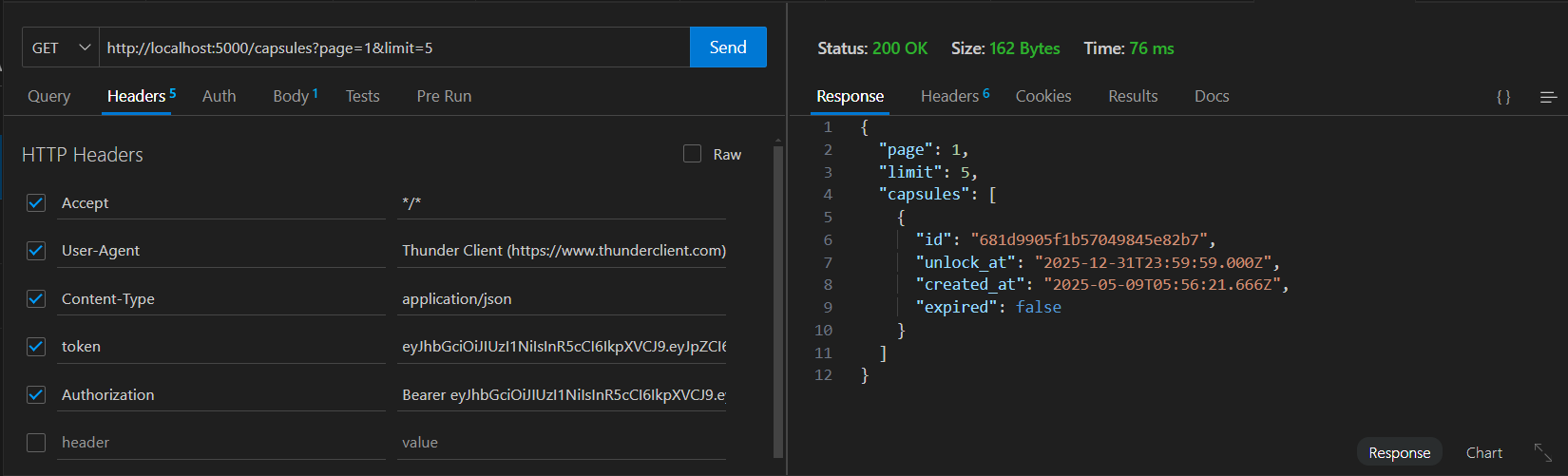
****

Figure 10: List of capsules

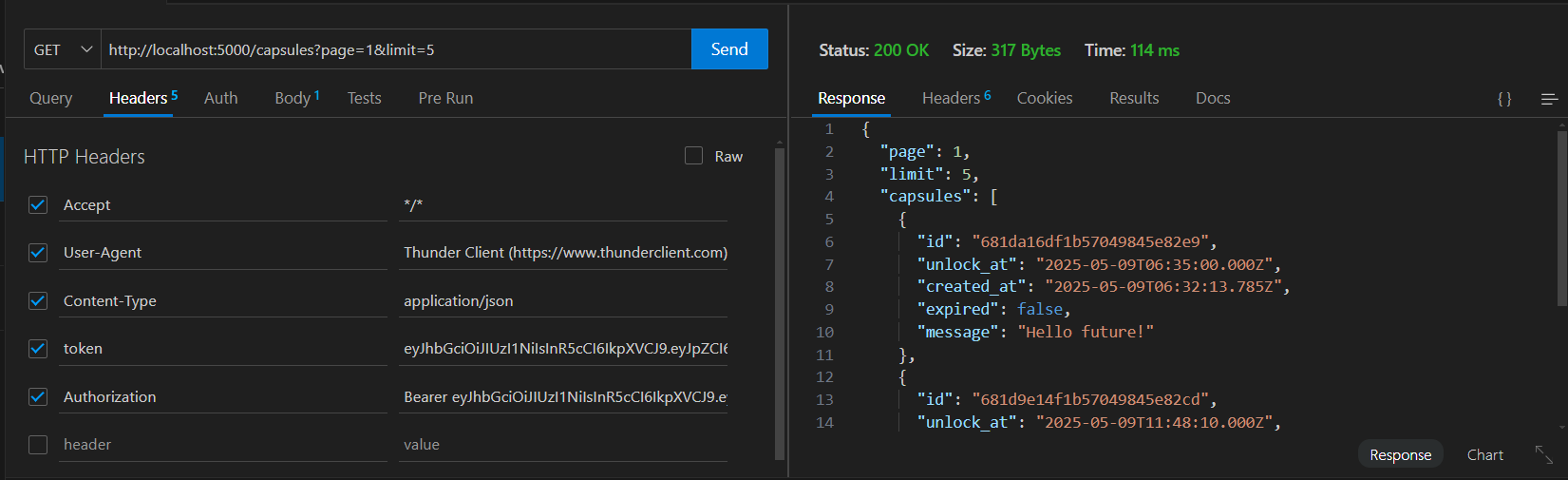
****

Figure 11: List of capsules and an unlocked capsule with the message

# **Update Capsule**

* **Method:** PUT
* **URL:** /capsules/:id?code=<UNLOCK\_CODE>

**Body:**

{

"message": "Updated message",

"unlock\_at": "2026-01-01T00:00:00Z"

}

**Expected Responses:**

* 200 OK → Success
* 401 → Invalid unlock code
* 403 → Capsule already unlocked

**Manual Testing Output:**

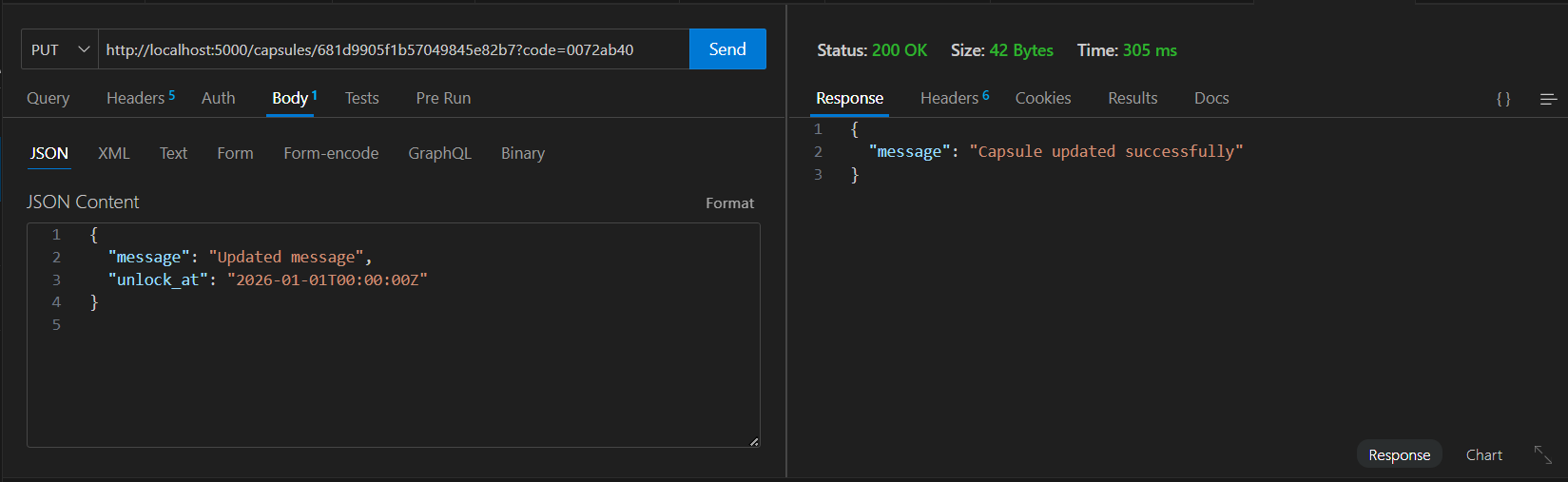
****

Figure 12: 200 OK; Capsule updated successfully

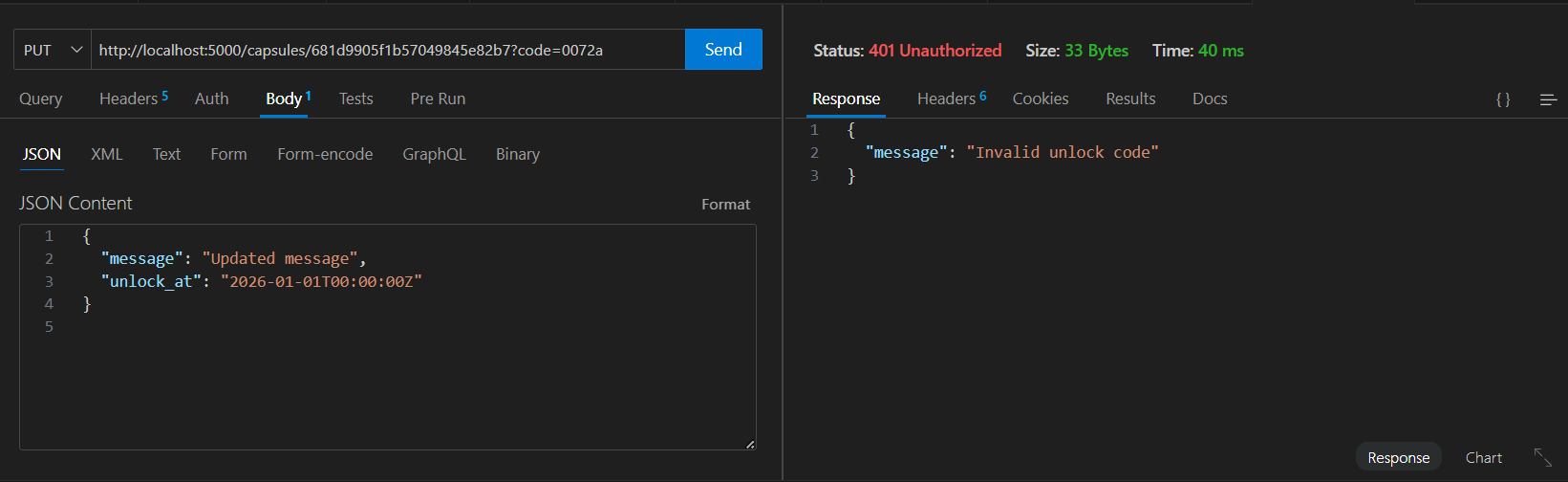
****

Figure 13; 401 Unauthorized; Wrong unlock code

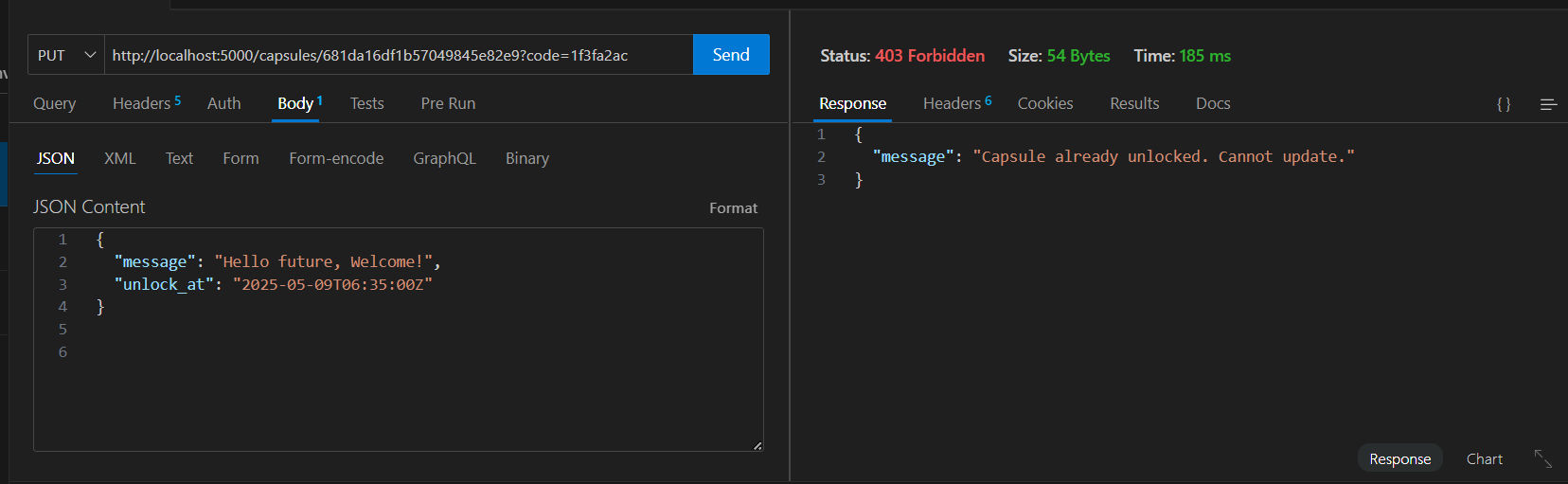
****

Figure 14: 403 Forbidden; Capsule already unlocked, thus cannot be updated

# **Delete Capsule**

* **Method:** DELETE
* **URL:** /capsules/:id?code=<UNLOCK\_CODE>

**Expected Responses:**

* 200 OK → Capsule deleted
* 401 → Wrong code
* 403 → Unlock time has passed
* 404 → Already deleted or wrong ID

**Manual Testing Output:**

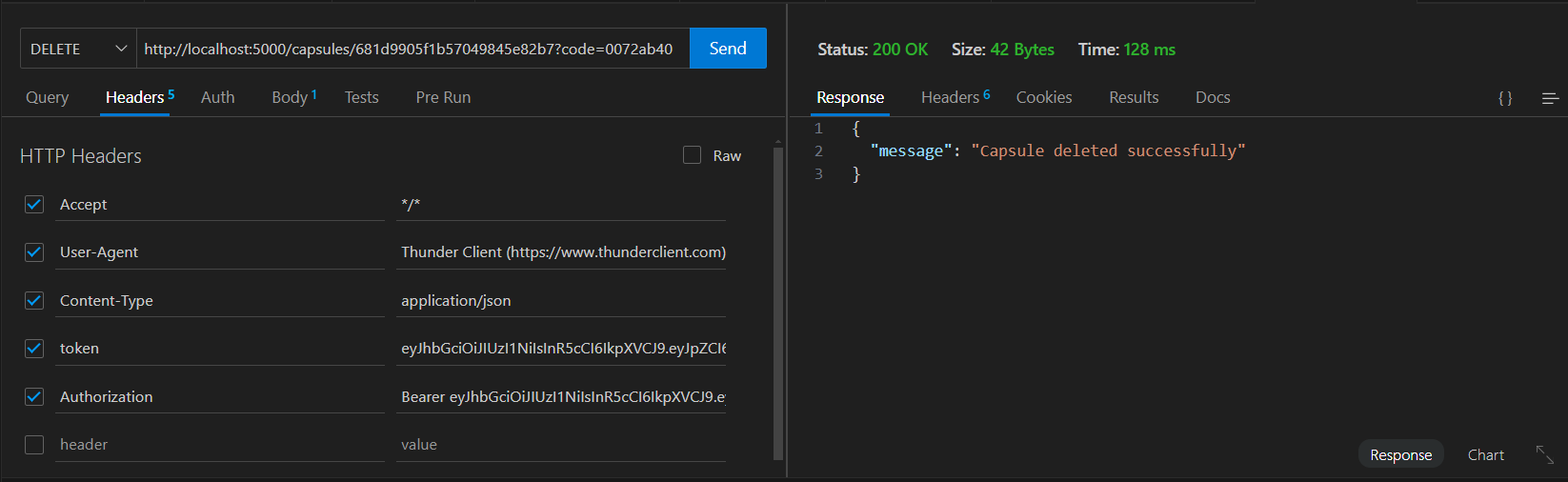
****

Figure 15: 200 OK; Capsule deleted successfully

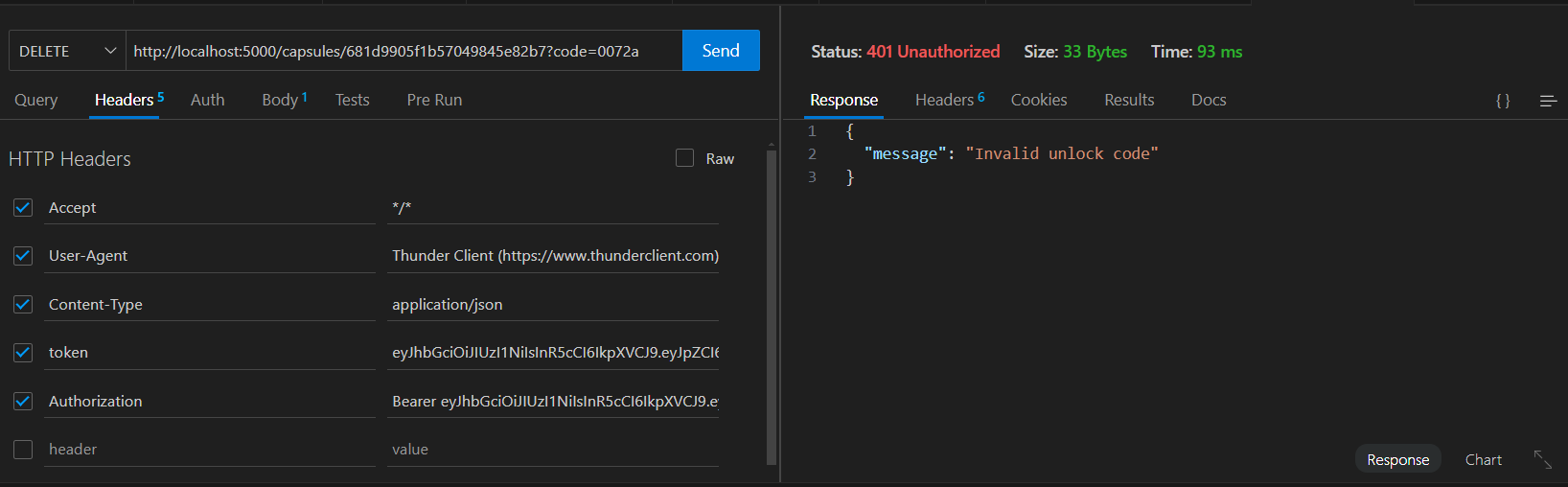
****

Figure 16: 401 Unauthorized; Wrong unlock code

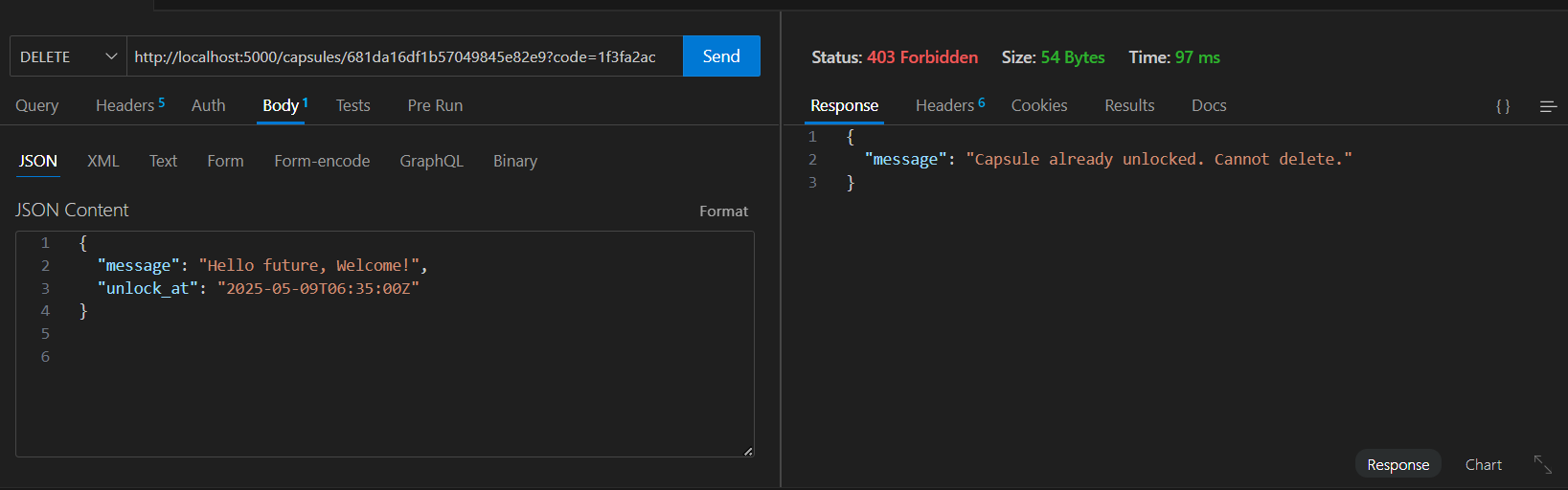
****

Figure 17:403 Forbidden; Capsule already unlocked

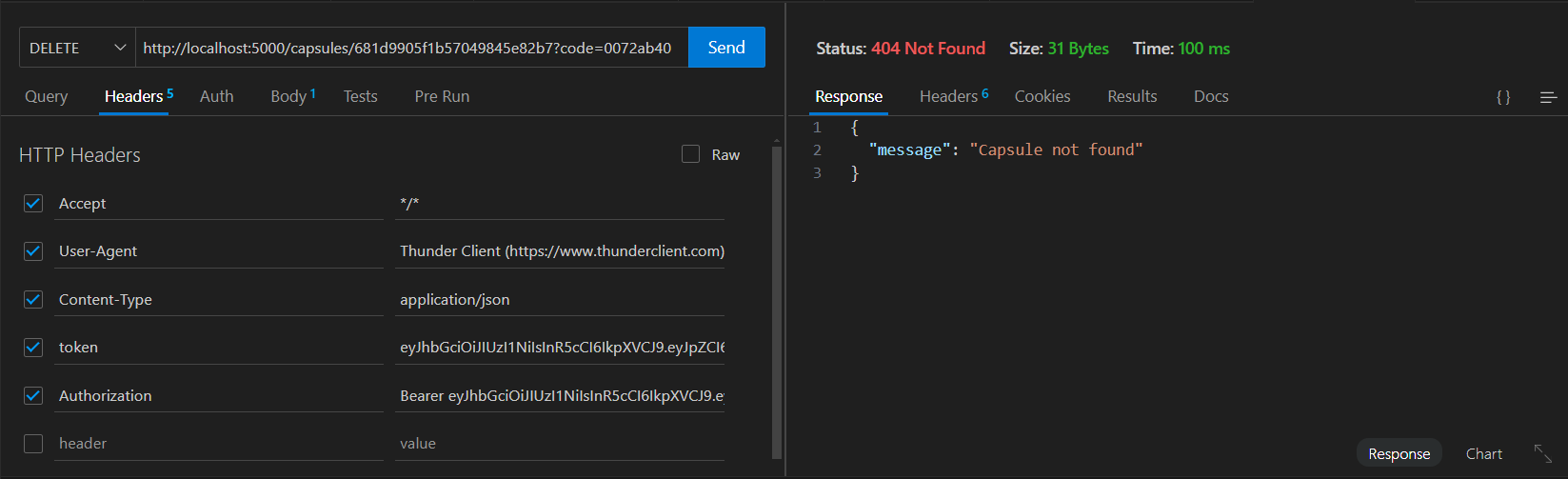
****

Figure 18: 404 Not found; Capsule already deleted

# **Expiration Behaviour**

* Capsules **expire automatically** 30 days after unlock\_at.
* If GET /capsules/:id is called for an expired capsule, response is:

{

"message": "Capsule expired"

}

**Manual Testing Output:**

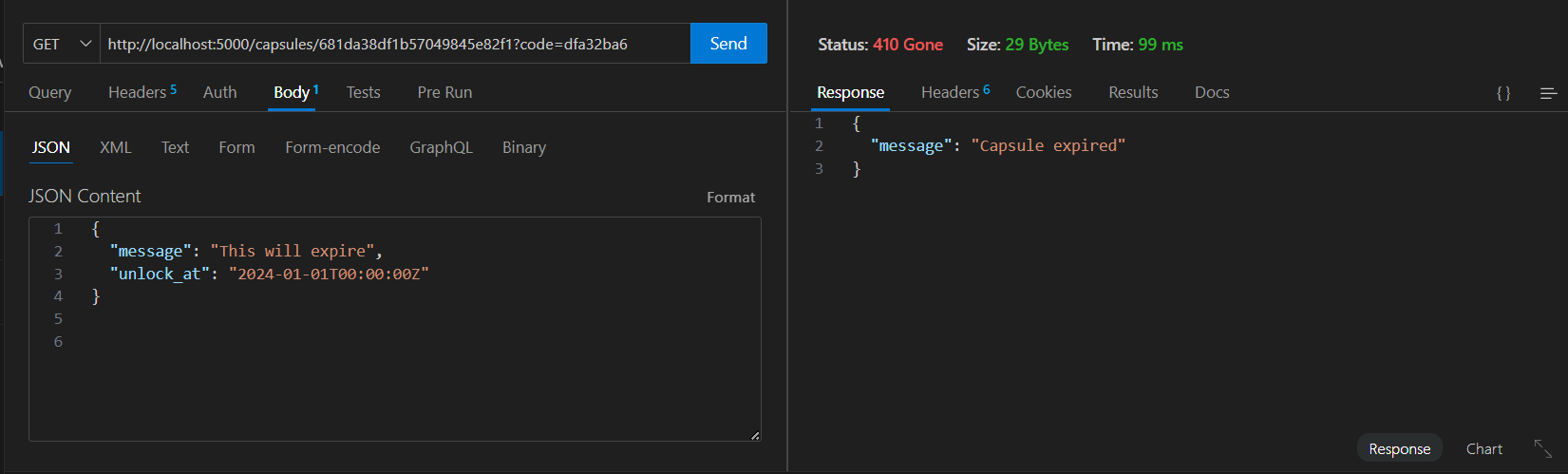
****

Figure 19: 410 Gone; Capsule Expired

**Jest Testing**

# **Setup Instructions**

PowerShell command:

* 1. Install Dev Dependencies: npm install --save-dev jest supertest
  2. To run the tests: npm test

# **auth.test.js**

1. **User Registration**

* Registers a new user with a unique email.
* Expects:
  + 201 Created
  + message: "User created"

1. **User Login**

* Logs in with registered credentials.
* Expects:
  + 200 OK
  + A valid JWT token in response.

# **capsule.test.js**

1. **Setup**

* Creates a test user and logs in.
* Reuses the token in all capsule operations.
* Clears relevant collections before/after tests.

1. **Capsule Creation**

* Endpoint: POST /capsules
* Creates a new capsule with message and unlock\_at.
* Expects:
  + 201 Created
  + Returns capsule id and unlock\_code.

1. **Capsule Retrieval (GET)**

* Endpoint: GET /capsules/:id?code=...

**Tested Scenarios:**

|  |  |  |
| --- | --- | --- |
| Condition | Expected Status | Behavior |
| Valid, still locked | 403 | Capsule is locked |
| Valid, wrong unlock code | 401 | Unauthorized |
| Capsule expired | 410 | Marked as expired |

1. **Capsule Update (PUT)**

* Endpoint: PUT /capsules/:id?code=...

**Tested Scenarios:**

|  |  |  |
| --- | --- | --- |
| Condition | Expected Status | Behavior |
| Valid code, still locked | 200 | Capsule updated |
| Wrong unlock code | 401 | Invalid unlock code |
| After unlock time | 403 | Forbidden to update |

1. **Capsule Delete (DELETE)**

* Endpoint: DELETE /capsules/:id?code=...

**Tested Scenarios:**

|  |  |  |
| --- | --- | --- |
| Condition | Expected Status | Behavior |
| Valid, still locked | 200 | Capsule deleted |
| Already deleted | 404 | Capsule not found |

1. **Capsule Listing (Pagination)**

* Endpoint: GET /capsules?page=1&limit=5

**Tested Scenarios:**

|  |  |  |
| --- | --- | --- |
| Condition | Expected Status | Behavior |
| Page 1 request | 200 | Returns up to 5 capsules |
| Page 2 request | 200 | Returns next batch |

1. **Expiration Logic**

* Manually marks a capsule as is\_expired: true in the DB.
* Requests the capsule and expects:
  + 410 Gone
  + "Capsule expired" message

**Testing Output:**

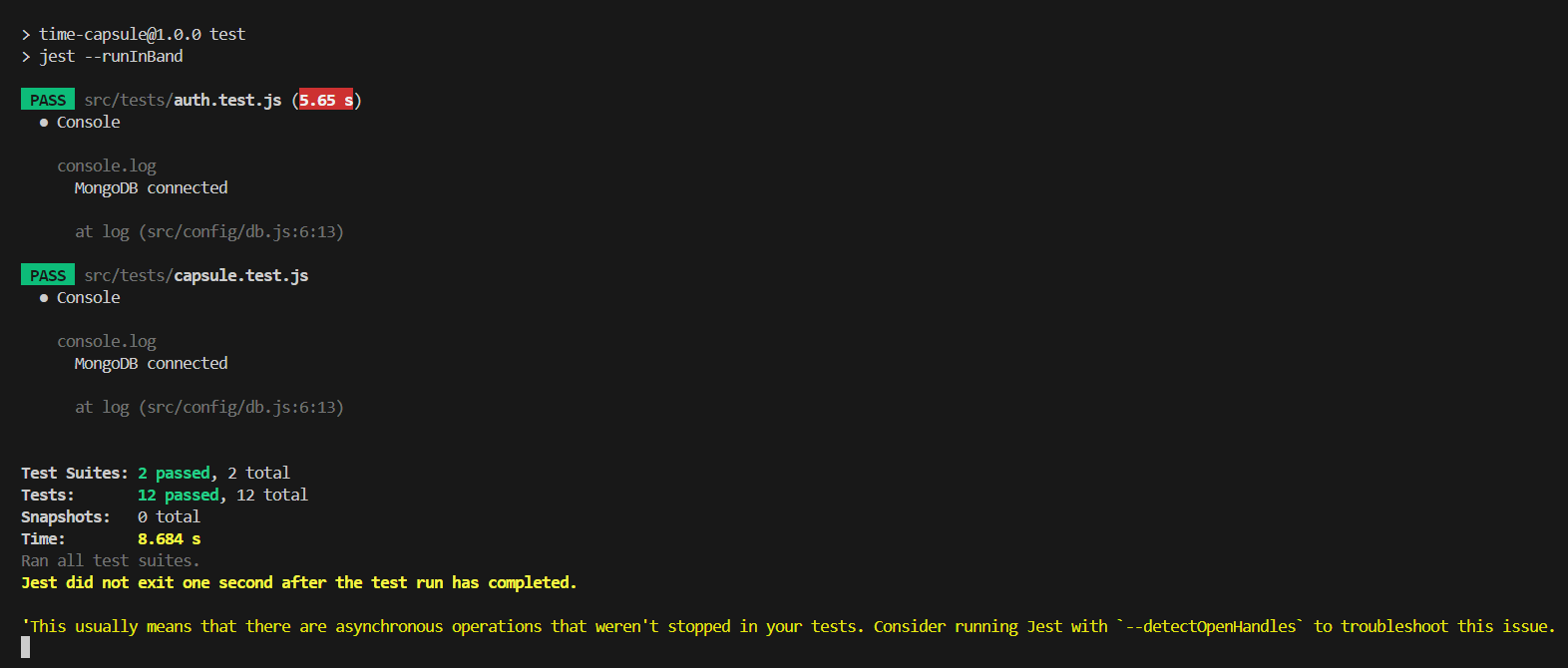
****

Figure 20: Test cases passed for all the functionality

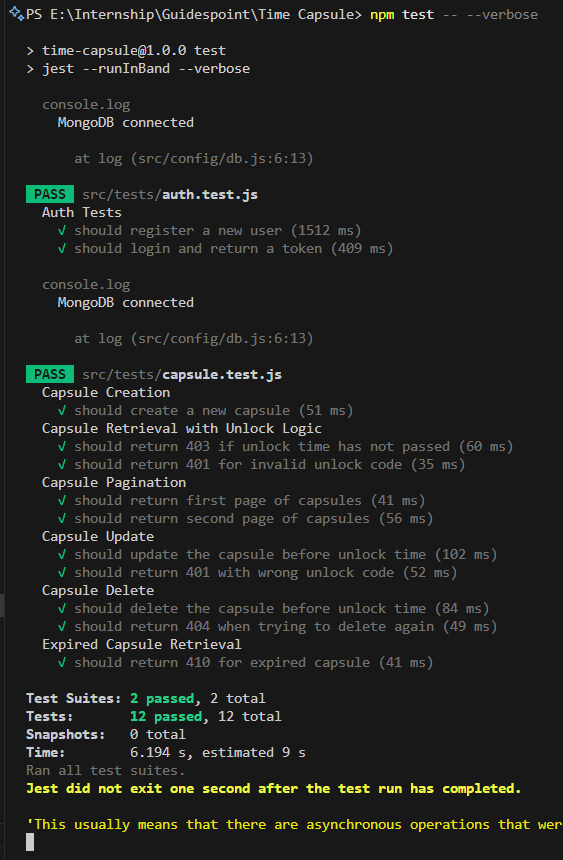


Figure 21: Detailed report of all the passed test cases