

# 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:

The screenshot shows the Thunder Client interface. In the top left, there's a dropdown menu set to 'POST' and a URL input field containing 'http://localhost:5000/auth/register'. To the right of the URL is a blue 'Send' button. Below the URL, there are tabs for 'Query', 'Headers', 'Auth', 'Body' (which is currently selected), 'Tests', and 'Pre Run'. Under the 'Body' tab, there are buttons for 'JSON', 'XML', 'Text', 'Form', 'Form-encode', 'GraphQL', and 'Binary'. The 'JSON Content' area contains the following JSON payload:

```
1 {  
2   "email": "user@example.com",  
3   "password": "yourpassword"  
4 }  
5
```

To the right of the body editor, there's a 'Format' button. The top right of the interface shows 'Status: 201 Created', 'Size: 26 Bytes', and 'Time: 370 ms'. Below this, there are tabs for 'Response', 'Headers', 'Cookies', 'Results', and 'Docs'. The 'Response' tab is active and displays the following JSON response:

```
1 {  
2   "message": "User created"  
3 }
```

At the bottom right, there are buttons for 'Response', 'Chart', and a refresh icon.

Figure 1: Successful user registration

The screenshot shows a POST request to `http://localhost:5000/auth/register`. The request body is JSON with fields `"email": "user@example.com"` and `"password": "yourpassword"`. The response status is **400 Bad Request**, size is **25 Bytes**, and time is **71 ms**. The response body is a JSON object with a single key `"message": "User exists"`.

```

Status: 400 Bad Request  Size: 25 Bytes  Time: 71 ms
Response Headers 6 Cookies Results Docs
1 {
2   "message": "User exists"
3 }

```

Figure 2: 400 error; user already exists

## 2. Login User

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

**Body:**

```
{
  "email": "user@example.com",
  "password": "yourpassword"
}
```

**Expected Response:**

```
{
  "token": "<JWT_TOKEN>"
}
```

**Manual Testing Output:**

The screenshot shows a POST request to `http://localhost:5000/auth/login`. The request body is JSON with fields `"email": "user@example.com"` and `"password": "yourpassword"`. The response status is **200 OK**, size is **183 Bytes**, and time is **307 ms**. The response body contains a JSON object with a key `"token"` containing a long string of characters.

```

Status: 200 OK  Size: 183 Bytes  Time: 307 ms
Response Headers 6 Cookies Results Docs
1 {
2   "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9
3     .eyJpZC16IjY4MmQ5NzZmZjF1NTcwNDk4NDVlODJiMSI6Imhdc16MTc0NjczMDAwMCwiZX
4     hwIjoaNzQ2ODU2NDAwFQ.P7Ly3V4eSn-osdNTNgauom4lk0vU_fryxiLqJHcchK0"
5 }

```

Figure 3: JWT Token is visible

## Capsule Endpoints

All capsule endpoints require the header:

Authorization: Bearer <JWT\_TOKEN>

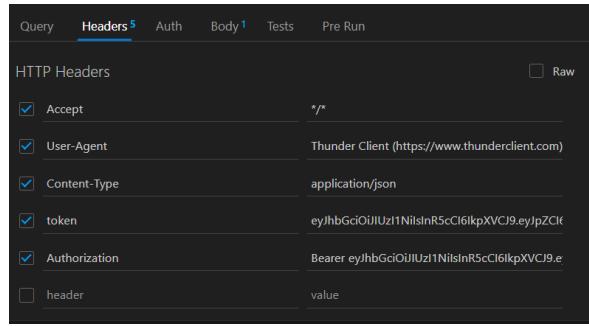


Figure 4: Require Headers

## 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:

The screenshot shows the results of a POST request to http://localhost:5000/capsules/. The response status is 201 Created, size is 58 Bytes, and time is 93 ms.

Response	Headers	Cookies	Results	Docs
<pre> 1  { 2    "id": "681d9905f1b57049845e82b7", 3    "unlock_code": "0072ab40" 4  } </pre>				

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:

The screenshot shows a POST request to `http://localhost:5000/capsules/681da16df1b57049845e82e9?code=1f3fa2ac`. The response status is 200 OK, size is 98 Bytes, and time is 61 ms. The response body is:

```

1 {
2   "id": "681da16df1b57049845e82e9",
3   "message": "Hello future!",
4   "unlock_at": "2025-05-09T06:35:00Z"
5 }

```

Figure 6: 200 OK; Unlocked Capsule

The screenshot shows a POST request to `http://localhost:5000/capsules/681d9905f1b57049845e82b7?code=0072`. The response status is 401 Unauthorized, size is 33 Bytes, and time is 46 ms. The response body is:

```

1 {
2   "message": "Invalid unlock code"
3 }

```

Figure 7: 401 Unauthorized; Wrong Code

The screenshot shows a POST request to `http://localhost:5000/capsules/681d9905f1b57049845e82b7?code=0072ab40`. The response status is 403 Forbidden, size is 37 Bytes, and time is 92 ms. The response body is:

```

1 {
2   "message": "Capsule is still locked"
3 }

```

Figure 8: 403 Forbidden; Capsule is still locked

The screenshot shows a POST request to `http://localhost:5000/capsules/681da38df1b57049845e82f1?code=dfa32ba6`. The response status is 410 Gone, size is 29 Bytes, and time is 99 ms. The response body is:

```

1 {
2   "message": "Capsule expired"
3 }

```

Figure 9: 410 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 shows the manual testing output for listing capsules. The request is a GET to `http://localhost:5000/capsules?page=1&limit=5`. The response is a 200 OK with a size of 162 bytes and a time of 76 ms. The response body is a JSON object with `page: 1`, `limit: 5`, and `capsules: [{}]`. The capsule object has fields `id`, `unlock_at`, `created_at`, and `message`.

Figure 10: List of capsules

Figure 11 shows the manual testing output for listing capsules. The request is a GET to `http://localhost:5000/capsules?page=1&limit=5`. The response is a 200 OK with a size of 317 bytes and a time of 114 ms. The response body is a JSON object with `page: 1`, `limit: 5`, and `capsules: [{}]`. The capsule object has fields `id`, `unlock_at`, `created_at`, and `message`, with `message: "Hello future!"`.

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:

The screenshot shows a POST request to `http://localhost:5000/capsules/681d9905f1b57049845e82b7?code=0072ab40`. The response status is 200 OK, size is 42 Bytes, and time is 305 ms. The response body is:

```

{
    "message": "Capsule updated successfully"
}

```

Figure 12: 200 OK; Capsule updated successfully

The screenshot shows a POST request to `http://localhost:5000/capsules/681d9905f1b57049845e82b7?code=0072a`. The response status is 401 Unauthorized, size is 33 Bytes, and time is 40 ms. The response body is:

```

{
    "message": "Invalid unlock code"
}

```

Figure 13: 401 Unauthorized; Wrong unlock code

The screenshot shows a POST request to `http://localhost:5000/capsules/681da16df1b57049845e82e9?code=1f3fa2ac`. The response status is 403 Forbidden, size is 54 Bytes, and time is 185 ms. The response body is:

```

{
    "message": "Capsule already unlocked. Cannot update."
}

```

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:

DELETE http://localhost:5000/capsules/681d9905f1b57049845e82b7?code=0072ab40

**Headers**

- Accept: \*/\*
- User-Agent: Thunder Client (https://www.thunderclient.com)
- Content-Type: application/json
- token: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZClt...
- Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.e...

**Response**

```

1  {
2      "message": "Capsule deleted successfully"
3  }

```

Figure 15: 200 OK; Capsule deleted successfully

DELETE http://localhost:5000/capsules/681d9905f1b57049845e82b7?code=0072a

**Headers**

- Accept: \*/\*
- User-Agent: Thunder Client (https://www.thunderclient.com)
- Content-Type: application/json
- token: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZClt...
- Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.e...

**Response**

```

1  {
2      "message": "Invalid unlock code"
3  }

```

Figure 16: 401 Unauthorized; Wrong unlock code

DELETE http://localhost:5000/capsules/681da16df1b57049845e82e9?code=1f3fa2ac

**Body**

**JSON**

```

1  {
2      "message": "Hello future, Welcome!",
3      "unlock_at": "2025-05-09T06:35:00Z"
4  }

```

**Response**

```

1  {
2      "message": "Capsule already unlocked. Cannot delete."
3  }

```

Figure 17: 403 Forbidden; Capsule already unlocked

The screenshot shows a Thunder Client interface. In the top bar, there is a 'DELETE' button, a dropdown menu, the URL 'http://localhost:5000/capsules/681d9905f1b57049845e82b7?code=0072ab40', and a 'Send' button. Below this, there are tabs for 'Query', 'Headers 5', 'Auth', 'Body 1', 'Tests', and 'Pre Run'. The 'Headers' tab is selected. It contains the following fields:

- Accept**: \*/\*
- User-Agent**: Thunder Client (https://www.thunderclient.com)
- Content-Type**: application/json
- token**: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6...
- Authorization**: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.e...
- header**: value

In the main area, the status bar shows 'Status: 404 Not Found Size: 31 Bytes Time: 100 ms'. The 'Response' tab is selected, displaying the following JSON response:

```

1 {
2   "message": "Capsule not found"
3 }

```

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:

The screenshot shows a Thunder Client interface. In the top bar, there is a 'GET' button, a dropdown menu, the URL 'http://localhost:5000/capsules/681da30df1b57049845e82f1?code=dfa32ba6', and a 'Send' button. Below this, there are tabs for 'Query', 'Headers 5', 'Auth', 'Body 1', 'Tests', and 'Pre Run'. The 'Body' tab is selected. It contains the following JSON content:

```

1 {
2   "message": "This will expire",
3   "unlock_at": "2024-01-01T00:00:00Z"
4 }
5
6

```

In the main area, the status bar shows 'Status: 410 Gone Size: 29 Bytes Time: 99 ms'. The 'Response' tab is selected, displaying the following JSON response:

```

1 {
2   "message": "Capsule expired"
3 }

```

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"

#### **2. 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.

#### **2. Capsule Creation**

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

### **3. Capsule Retrieval (GET)**

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

**Tested Scenarios:**

Condition	Expected Status	Behavior
<b>Valid, still locked</b>	403	Capsule is locked
<b>Valid, wrong unlock code</b>	401	Unauthorized
<b>Capsule expired</b>	410	Marked as expired

### **4. Capsule Update (PUT)**

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

**Tested Scenarios:**

Condition	Expected Status	Behavior
<b>Valid code, still locked</b>	200	Capsule updated
<b>Wrong unlock code</b>	401	Invalid unlock code
<b>After unlock time</b>	403	Forbidden to update

### **5. Capsule Delete (DELETE)**

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

**Tested Scenarios:**

Condition	Expected Status	Behavior
<b>Valid, still locked</b>	200	Capsule deleted
<b>Already deleted</b>	404	Capsule not found

### **6. Capsule Listing (Pagination)**

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

**Tested Scenarios:**

Condition	Expected Status	Behavior

<b>Page 1 request</b>	200	Returns up to 5 capsules
<b>Page 2 request</b>	200	Returns next batch

## 7. 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:

```
> time-capsule@1.0.0 test
> jest --runInBand

PASS  src/tests/auth.test.js (5.65 s)
● Console

  console.log
    MongoDB connected
    at log (src/config/db.js:6:13)

PASS  src/tests/capsule.test.js
● Console

  console.log
    MongoDB connected
    at log (src/config/db.js:6:13)

Test Suites: 2 passed, 2 total
Tests:       12 passed, 12 total
Snapshots:  0 total
Time:        8.684 s
Ran all test suites.
Jest did not exit one second after the test run has completed.

This usually means that there are asynchronous operations that weren't stopped in your tests. Consider running Jest with `--detectOpenHandles` to troubleshoot this issue.
|
```

Figure 20: Test cases passed for all the functionality

```

PS E:\Internship\Guidespoint\Time Capsule> npm test -- --verbose

> time-capsule@1.0.0 test
> jest --runInBand --verbose

console.log
  MongoDB connected

    at log (src/config/db.js:6:13)

PASS  src/tests/auth.test.js
Auth Tests
  ✓ should register a new user (1512 ms)
  ✓ should login and return a token (409 ms)

console.log
  MongoDB connected

    at log (src/config/db.js:6:13)

PASS  src/tests/capsule.test.js
Capsule Creation
  ✓ should create a new capsule (51 ms)
Capsule Retrieval with Unlock Logic
  ✓ should return 403 if unlock time has not passed (60 ms)
  ✓ should return 401 for invalid unlock code (35 ms)
Capsule Pagination
  ✓ should return first page of capsules (41 ms)
  ✓ should return second page of capsules (56 ms)
Capsule Update
  ✓ should update the capsule before unlock time (102 ms)
  ✓ should return 401 with wrong unlock code (52 ms)
Capsule Delete
  ✓ should delete the capsule before unlock time (84 ms)
  ✓ should return 404 when trying to delete again (49 ms)
Expired Capsule Retrieval
  ✓ should return 410 for expired capsule (41 ms)

Test Suites: 2 passed, 2 total
Tests:       12 passed, 12 total
Snapshots:  0 total
Time:        6.194 s, estimated 9 s
Ran all test suites.
Jest did not exit one second after the test run has completed.

'This usually means that there are asynchronous operations that were

```

Figure 21: Detailed report of all the passed test cases