

# TEST

## 1.The digital time capsule

GET:<http://localhost:3000/>

The screenshot shows the Postman application interface. A GET request is made to <http://localhost:3000/>. The response status is 200 OK, with a response time of 68 ms and a size of 1.23 KB. The response body contains the following HTML code:

```
2 <html>
3   <head>
4     <title>Digital Time Capsule</title>
5   </head>
6 
7   <body>
8 
9     <h2>Digital Time Capsule</h2>
10 
11 
12 
13 
14     <form action="/add" method="POST">
15       <textarea name="message" placeholder="Enter message" required
16         style="width: 300px; height: 100px; padding: 10px; border-radius: 5px; border: 1px solid #ccc;"></textarea><br><br>
17       <input type="datetime-local" name="time" required style="padding: 5px;"><br><br>
18       <button
19         style="background-color: #28a745; color: white; padding: 10px 20px; border: none; border-radius: 5px; cursor: pointer;">Schedule
20           Message</button>

```

### Digital Time Capsule

Enter message

---

mm/dd/yyyy -- : -- : --

Schedule Message

#### All Messages

- [Message 1](#)

→ POST: <http://localhost:3000/add>

The screenshot shows the Postman application interface. A POST request is being made to `http://localhost:3000/add`. The request body contains the following JSON:

```
1 "message": "Test message",
2 "time": "2026-12-25T12:00"
```

The response status is 200 OK, with a response time of 49 ms and a size of 1.57 KB. The response body is displayed as HTML:

```
2 <html>
3 <head>
4 | <title>Digital Time Capsule</title>
5 </head>
6 <body>
7
8 <h2>Digital Time Capsule</h2>
9
10
11
12
```

## Digital Time Capsule

Test message

02/14/2026 11:51 AM

Schedule Message

### All Messages

- [Message 1](#)
- [Message 2](#) 🔒

GET: <http://localhost:3000/message/1771045872071>

The screenshot shows a Postman interface with a single request listed:

- Method:** GET
- URL:** <http://localhost:3000/message/1771045872071>
- Body:** Raw JSON response:

```
{ "Message": "hi", "Released at": "2/14/2026, 10:42:00 AM" }
```

The response details are as follows:

- Status:** 200 OK
- Time:** 98 ms
- Size:** 896 B

## Message Details

### Message:

hi

Released at: 2/14/2026, 10:42:00 AM

[Delete](#)

[Back](#)

POST: <http://localhost:3000/delete/1771045872071>

The screenshot shows the Postman application interface. A POST request is made to <http://localhost:3000/delete/1771045872071>. The response is a 200 OK status with a response time of 42 ms and a size of 1.4 KB. The response body contains a JSON object with a single key-value pair: "Message": "Message 1".

## Digital Time Capsule

Enter message

---

mm/dd/yyyy --:-- --

**Schedule Message**

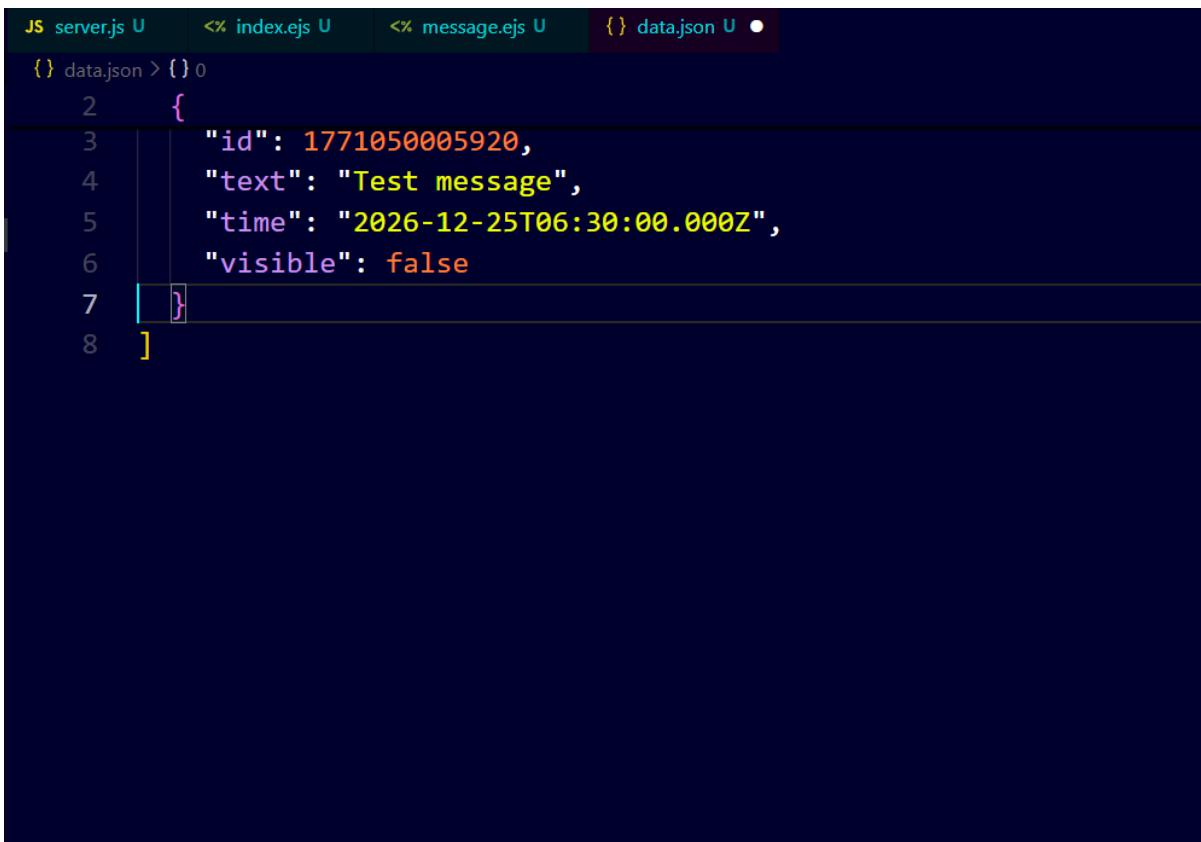
### All Messages

- [Message 1](#)

## Maaasge has been deleted

```
5 server.js U    <% index.ejs U    <% message.ejs U    {} data.json U •
{} data.json > {} 2
  2   {
  3     "id": 1771045872071,
  4     "text": "hi ",
  5     "time": "2026-02-14T05:12:00.000Z",
  6     "visible": true
  7   },
  8   {
  9     "id": 1771049982150,
 10     "text": "500",
 11     "time": "2026-02-14T06:20:00.000Z",
 12     "visible": true
 13   },
 14   {
 15     "id": 1771050005920,
 16     "text": "Test message",
 17     "time": "2026-12-25T06:30:00.000Z",
 18     "visible": false
 19   }
20 ]
```

This was before



The screenshot shows a code editor with a dark theme. At the top, there are tabs for 'server.js U', '<% index.ejs U', '<% message.ejs U', and '{ } data.json U'. The 'data.json' tab is active, indicated by a red dot. The code in the editor is:

```
{ } data.json > {} 0
2     {
3         "id": 1771050005920,
4         "text": "Test message",
5         "time": "2026-12-25T06:30:00.000Z",
6         "visible": false
7     }
8 ]
```

Check the id it has been removed