**Event Scheduler System**

A simple Python Flask application designed to help users manage their events efficiently. This system allows for basic Create, Read, Update, and Delete (CRUD) operations on events, with data persistence and a real-time reminder feature.

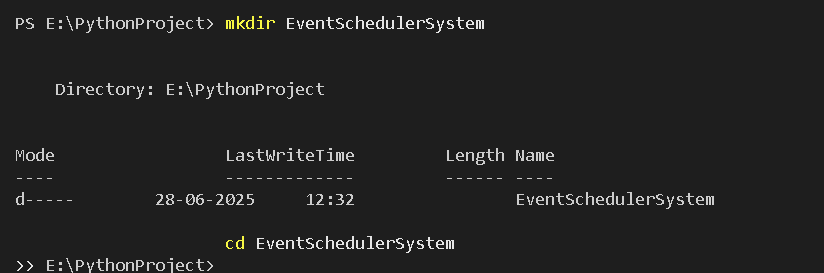
**Features**

* **Event Creation:** Easily add new events, specifying a title, description, start time, and end time.
* **Event Listing:** Retrieve all scheduled events, neatly sorted by their start time (earliest first).
* **Event Updating:** Modify any detail of an existing event using its unique ID. Partial updates are supported.
* **Event Deletion:** Remove events from your schedule using their unique ID.
* **Data Persistence:** All your events are automatically saved to a local JSON file (events.json) and loaded when the application starts, ensuring no data loss between sessions.
* **Real-time Reminders:** The system includes a background process that checks every minute for events starting within the next hour. Timely reminders are printed directly to the console where the Flask application is running.

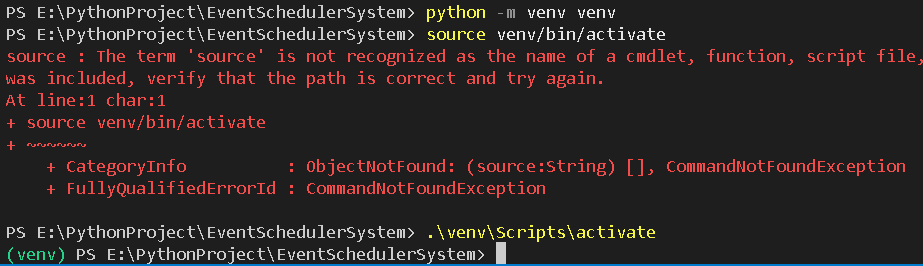
**Technologies Used**

* **Python 3.x:** The core programming language.
* **Flask:** A lightweight web framework for building the RESTful API.
* **Flask-APScheduler:** Used for scheduling the background task that checks for event reminders.
* **json module:** For handling event data persistence to and from events.json.
* **datetime module:** For robust handling and validation of event times.
* **uuid module:** For generating unique identifiers for each event.

Creating Project Folder



Using a virtual environment to manage project dependencies and activating Virtual Environment.



Installing required libraries: pip install Flask

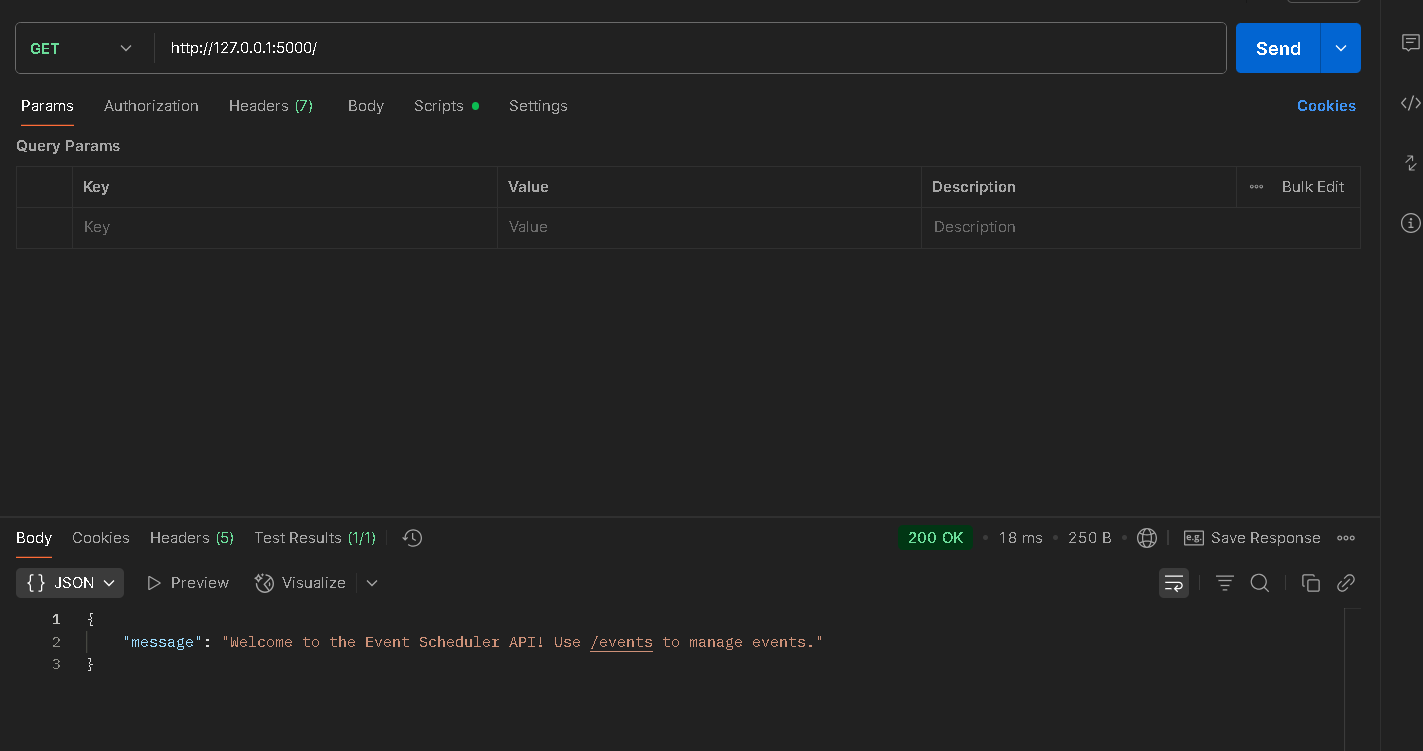
Creating the app.py for events and code writing.

Running app.py code: python app.py

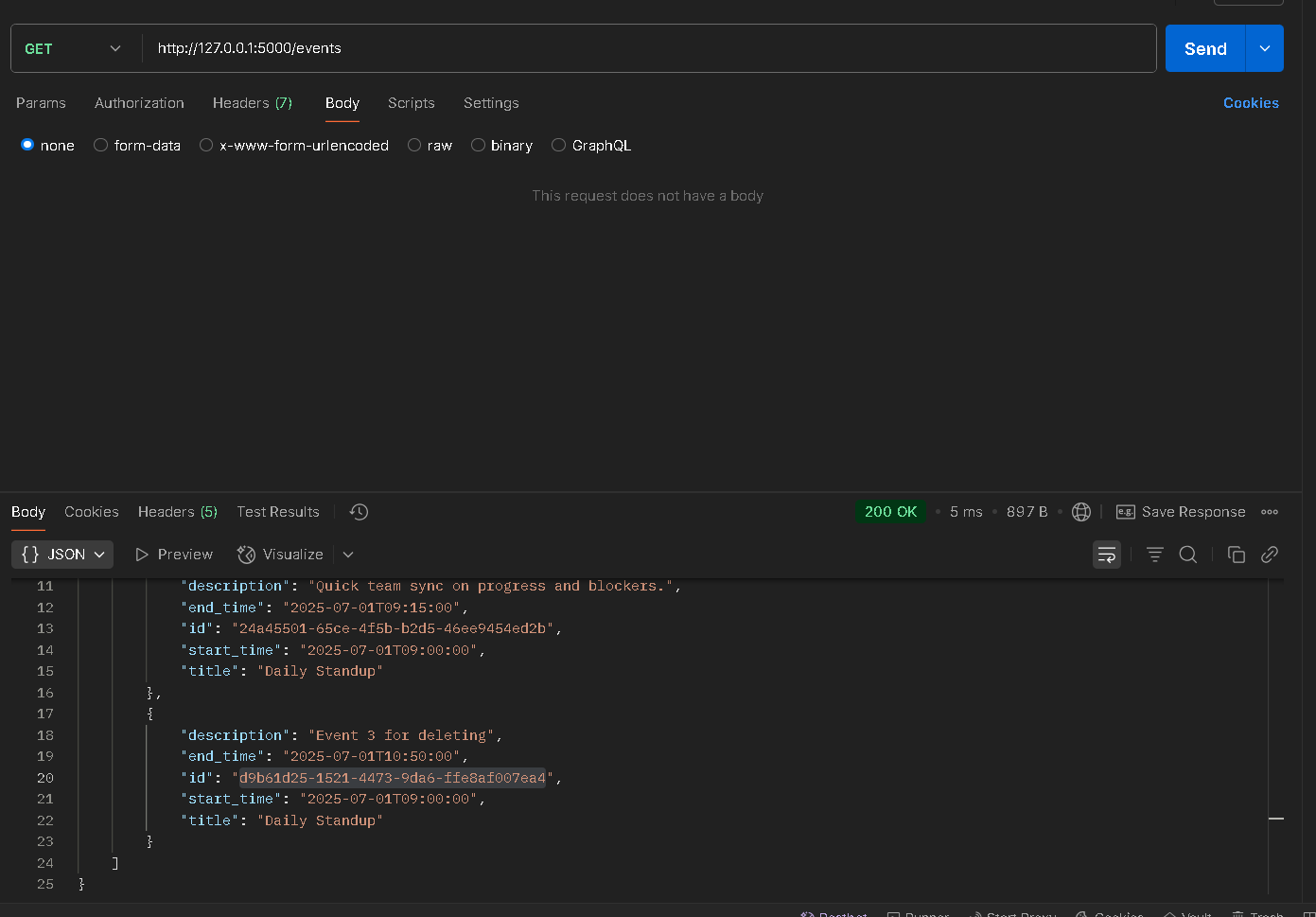
<http://127.0.0.1:5000>

Testing in Postman:

Get method: <http://127.0.0.1:5000/>

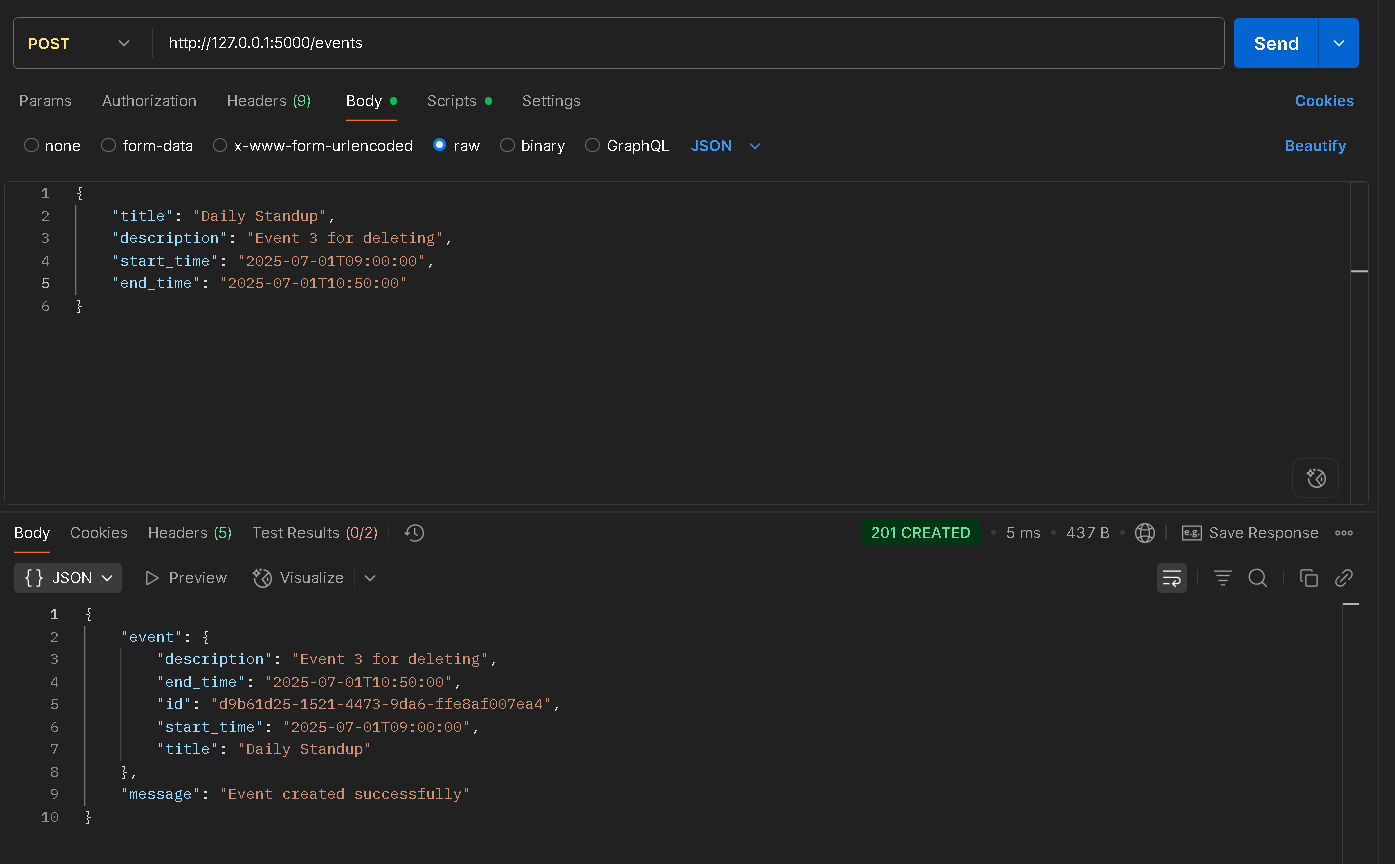


**Testing Viewing Events:** http://127.0.0.1:5000/events



**Expected Response:** 200 OK with a JSON array of all events, sorted by start\_time

**Testing Adding Events:** http://127.0.0.1:5000/events



**Body (raw, JSON):**

{

"title": "Daily Standup",

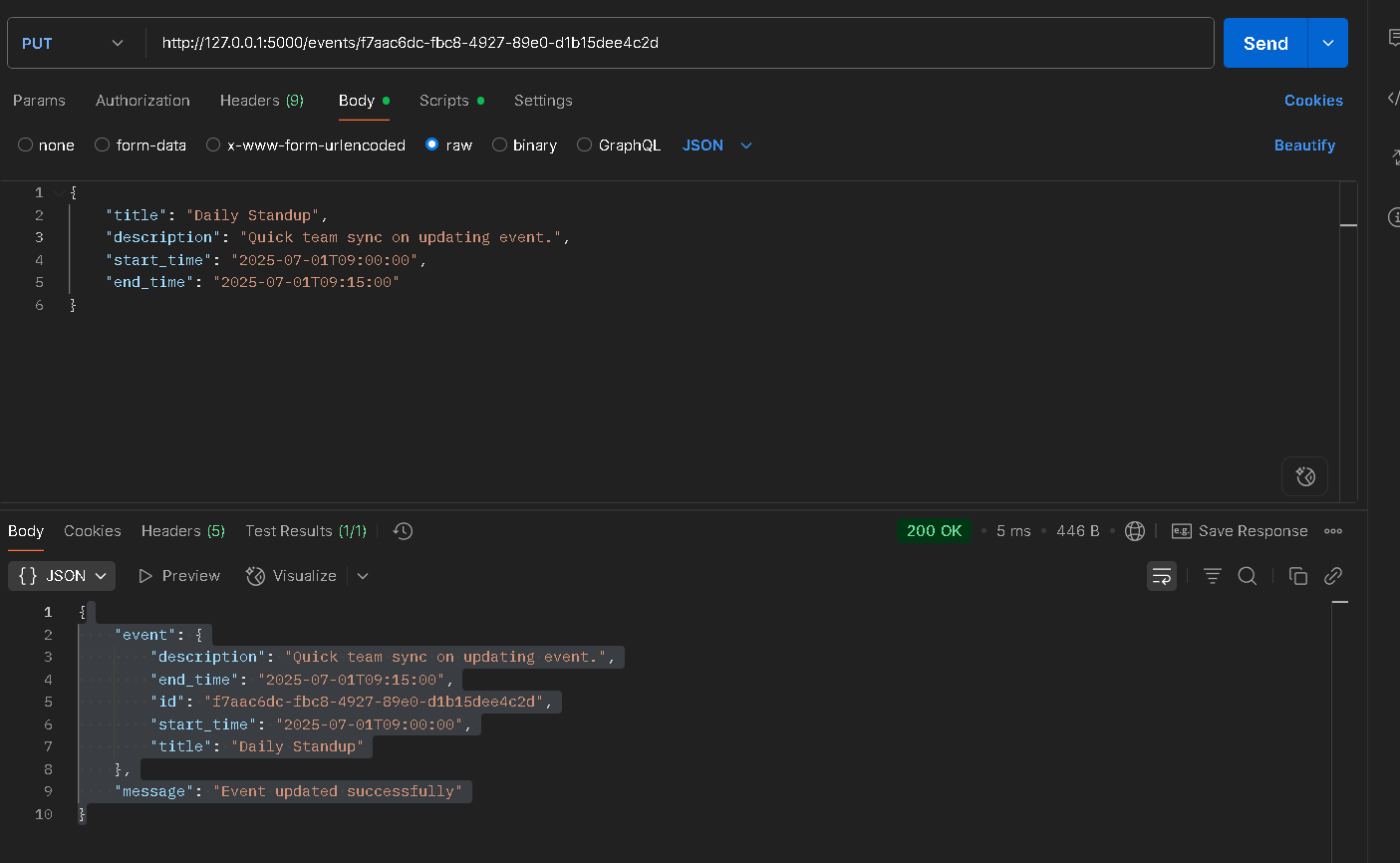
"description": "Quick team sync on progress and blockers.",

"start\_time": "2025-07-01T09:00:00",

"end\_time": "2025-07-01T09:15:00"

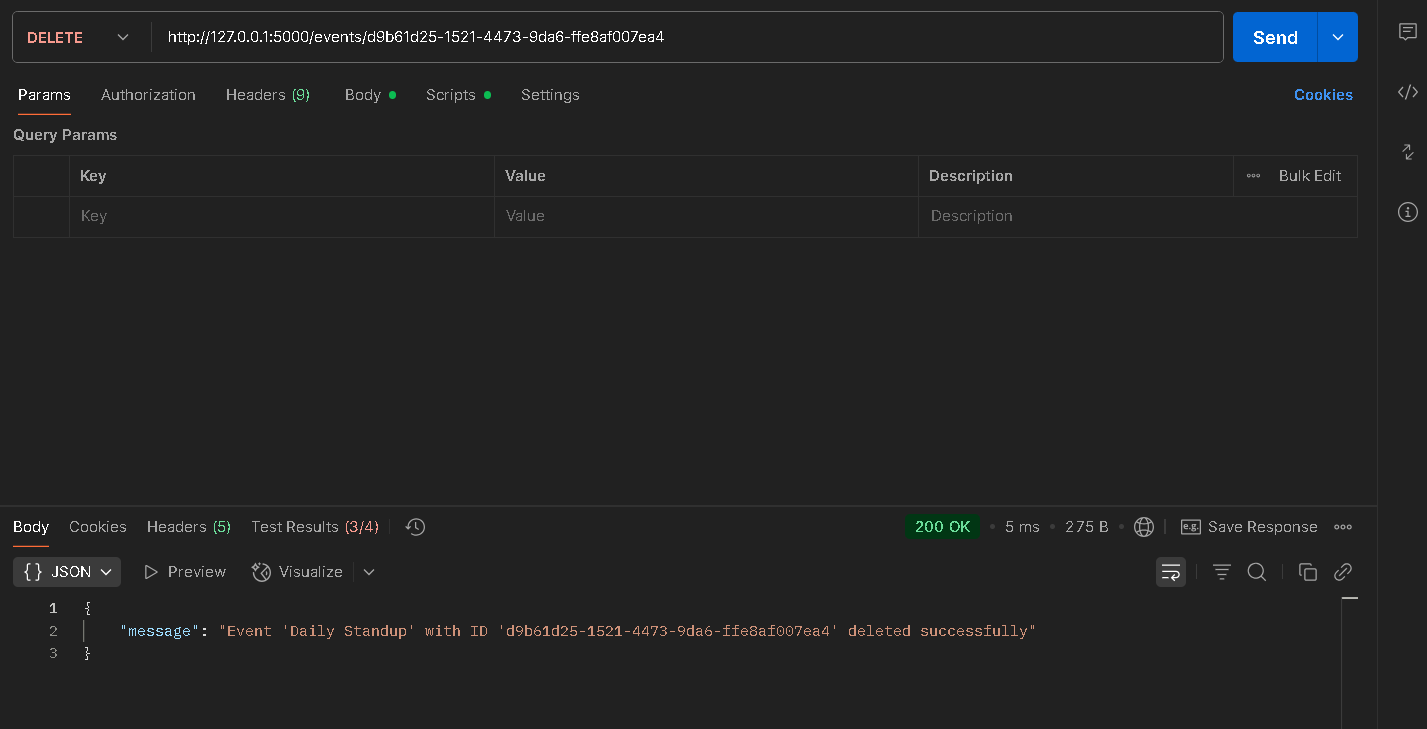
}

**Testing Update an Event:** http://127.0.0.1:5000/events/YOUR\_EVENT\_ID\_HERE

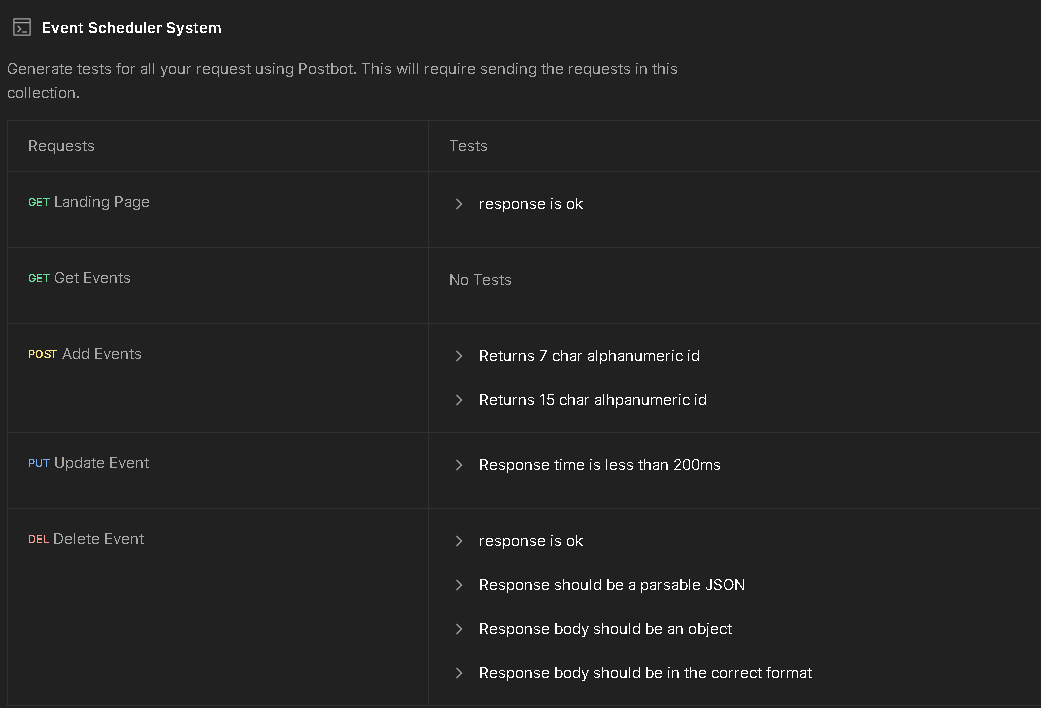


**Expected Response:** 200 OK with the updated event details.

Testing Delete Event: http://127.0.0.1:5000/events/YOUR\_EVENT\_ID\_HERE

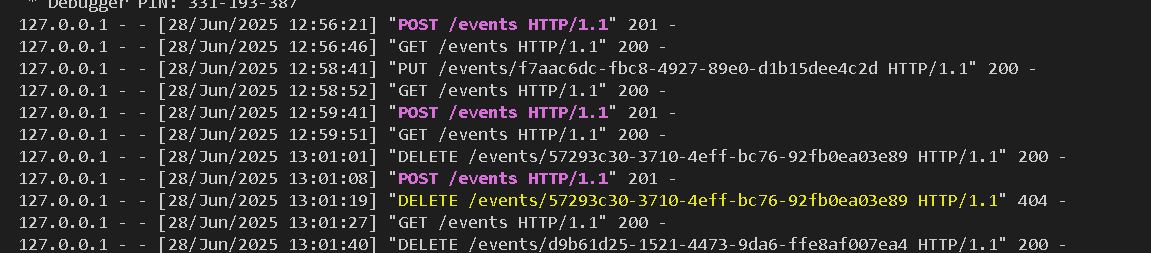


**Expected Response:** 200 OK with a confirmation message that the event was deleted.



Postman collection link: <https://www.postman.com/material-architect-44568090/event-handling-project/collection/zok11o0/event-scheduler-system?action=share&creator=0>

**Console Outputs:**



**API Endpoints:**

The API uses JSON. All event times (start\_time, end\_time) must be YYYY-MM-DDTHH:MM:SS.

* **GET /**
  + **Purpose:** Check API status.
  + **Response:** {"message": "Welcome to the Event Scheduler API! Use /events to manage events."}
* **POST /events**
  + **Purpose:** Create a new event.
  + **Request Body:** {"title": "string", "description": "string", "start\_time": "YYYY-MM-DDTHH:MM:SS", "end\_time": "YYYY-MM-DDTHH:MM:SS"}
  + **Success (201 Created):** Returns the new event with its id.
  + **Error (400 Bad Request):** Data is missing or invalid.
* **GET /events**
  + **Purpose:** Get all events.
  + **Response (200 OK):** {"events": [...]}. Events are sorted by start\_time.
* **PUT /events/<event\_id>**
  + **Purpose:** Update an event.
  + **URL Parameter:** event\_id (the event's unique ID).
  + **Request Body:** {"title": "string", "description": "string", "start\_time": "YYYY-MM-DDTHH:MM:SS", "end\_time": "YYYY-MM-DDTHH:MM:SS"} (partial updates work).
  + **Success (200 OK):** Returns updated event details.
  + **Error (400 Bad Request):** Invalid data.
  + **Error (404 Not Found):** event\_id not found.
* **DELETE /events/<event\_id>**
  + **Purpose:** Delete an event.
  + **URL Parameter:** event\_id (the event's unique ID).
  + **Success (200 OK):** Confirmation message.
  + **Error (404 Not Found):** event\_id not found.

**Testing with Postman**

Use Postman to test. A collection (EventScheduler.postman\_collection.json) might be in the project. Otherwise, make requests manually.

**Example Test Flow:**

1. **Create Event (POST):**
   * Send POST to http://127.0.0.1:5000/events with valid JSON.
   * **Copy the id** from the 201 Created response.
2. **View All Events (GET):**
   * Send GET to http://127.0.0.1:5000/events.
   * Confirm your new event appears.
3. **Update Event (PUT):**
   * Send PUT to http://127.0.0.1:5000/events/YOUR\_COPIED\_ID\_HERE. Replace YOUR\_COPIED\_ID\_HERE with your id.
   * Include JSON with changes (e.g., {"title": "Revised Project Kickoff"}).
   * Verify 200 OK. Check with GET /events again.
4. **Delete Event (DELETE):**
   * Send DELETE to http://127.0.0.1:5000/events/YOUR\_COPIED\_ID\_HERE.
   * Verify 200 OK. Confirm deletion with GET /events.