Exposee Documentation

Webhook User Notification System

The webhook user notification service allows you to subscribe to events carrying information about user registrations and messages sent between users in the fictional notification system to which the webhook service is connected.

When a new event occurs, the webhook service will send a POST request to the subscribed URL with information about the selected event.

The server is hosted on a demand basis using a local tunnel. If you want to use and test the service, you will need to contact the owner at the following email: marc303i@stud.kea.dk to receive the server's base URL.

Alternatively, you can clone or fetch the project from GitHub at:
https://github.com/Marcus-K-Thorsen/12a._Expose_and_integrate_with_a_webhook_system.git

You can interact with the service using either **Postman** or **Swagger**:

- **Postman:** Import the API endpoints into Postman to send requests and test the service.
- **Swagger:** Visit the Swagger documentation at `base_url/docs` to explore and test the endpoints directly in your browser.

Prerequisites:

- A publicly accessible URL (e.g., using a local tunnel service like ngrok) that can be used to subscribe to the webhook service.
- Basic knowledge of sending HTTP requests and working with JSON payloads.
- Access to the server's base URL, which can be obtained by contacting the owner or by fetching the project from GitHub and running it yourself.

Valid events:

The table below provides an overview of the events the webhook user notification service supports:

Name	Description
user_registered	An event that occurs when a new user is registered in the system.
user_send_message	An event that occurs when a user sends a message to another user.

Registration:

To register a webhook, send a **`POST`** request to **`/webhook`** with a JSON body specifying the URL that should receive updates when a specific event occurs and the event name you want to subscribe to.
See example below:

```
{
  "event": "user_registered",
  “url”: "http://your-webhook-receiver-url/webhook"
}
```

Explanation:

- **event:** The name of the event you want to subscribe to. For example, "user_registered" or "user_send_message".
- **url:** The publicly accessible URL where the webhook notifications should be sent. For example, "http://your-webhook-receiver-url/webhook".

This request will register the specified URL to receive updates whenever the specified event occurs in the system.

CURL Request:

The following CURL request shows an example of the registration request:

```
curl -X POST -H "Content-Type: application/json" -d '{"url":"YOUR_URL",
"event":"EVENT_NAME"}' SERVER_URL/webhook
```

- Replace **`YOUR_URL`** with the publicly accessible URL that should receive information on events (e.g., `http://your-webhook-receiver-url/webhook`).

- Replace **`EVENT_NAME`** with the event you want to subscribe to (e.g., `user_registered` or `user_send_message`).

- Replace **`SERVER_URL`** with the server address provided by the webhook registration owner (e.g., `http://127.0.0.1:8000` or the base URL provided by the owner).

Example:

```
curl -X POST -H "Content-Type: application/json" -d '{"url":"http://example.com/webhook",
"event":"user_registered"}' http://127.0.0.1:8000/webhook
```

This request registers the URL `http://example.com/webhook` to receive notifications for the `user_registered` event.

Unregistration:

To unregister a webhook, send a **`DELETE`** request to **`/webhook`** with a JSON body specifying the URL that should no longer receive updates when a specific event occurs and the event name you want to unsubscribe from. See example below:

```
{
  "event": "user_registered",
  “url”: "http://your-webhook-receiver-url/webhook"
}
```

CURL Request:

The following CURL request shows an example of the unregistration request:

```
curl -X DELETE -H "Content-Type: application/json" -d '{"url":"YOUR_URL",
"event":"EVENT_NAME"}' SERVER_URL/webhook
```

- Replace **`YOUR_URL`** with the URL that you subscribed to during the registration process (e.g., `http://example.com/webhook`).
- Replace **`EVENT_NAME`** with the event you want to unsubscribe from (e.g., `user_registered` or `user_send_message`).
- Replace **`SERVER_URL`** with the server address provided by the webhook service registration owner or if you are running the server locally: `http://127.0.0.1:8000`.

Example:

```
curl -X DELETE -H "Content-Type: application/json" -d '{"url":"http://example.com/webhook",
"event":"user_registered"}' http://127.0.0.1:8000/webhook
```

This request removes the URL `http://example.com/webhook` from receiving notifications for the `user_registered` event.

Retrieve Registered Webhooks:

To retrieve all registered webhooks grouped by event, send a `GET` request to `/webhooks`. You can also filter the results by specifying an event name as a query parameter.

Why Use This Endpoint?

- To get an overview of all registered webhooks and the events they are subscribed to.
- To check which URLs are subscribed to a specific event.

Example Request: Retrieve All Registered Webhooks:

```
curl -X GET "SERVER_URL/webhooks"
```

Example Response: All Registered Webhooks:

```
{
  "webhooks": [
    {
      "event": "user_registered",
      "urls": ["http://example.com/webhook1", "http://example.com/webhook2"]
    },
    {
      "event": "user_send_message",
      "urls": ["http://example.com/webhook3"]
    }
  ]
}
```

Example Request: Retrieve Webhooks for a Specific Event:

```
curl -X GET "SERVER_URL/webhooks?event_filter=user_registered"
```

Example Response: Webhooks for a Specific Event:

```
{
  "webhooks": [
    {
      "event": "user_registered",
      "urls": ["http://example.com/webhook1", "http://example.com/webhook2"]
    }
  ]
}
```

Explanation

- **Without Query Parameter**: The endpoint returns all registered webhooks grouped by event.
- **With Query Parameter** (`event_filter`): The endpoint filters the results to show only the webhooks for the specified event.

This endpoint is useful for debugging, monitoring, or verifying which webhooks are currently registered in the system.

Test Your Registered Webhooks:

You can test if your registered webhooks are working correctly by sending a `**POST**` request to `**/ping**`. This endpoint allows you to send a test payload to all registered webhook URLs or to the URLs registered for a specific event. If no payload is provided, a default payload will be used.

What Happens When You Use /ping?

1. The system retrieves all registered webhook URLs from the storage.
2. If an event filter is provided, only the URLs registered for that specific event will be pinged.
3. A `**POST**` request is sent to each registered URL with the provided payload (or the default payload if none is provided).
4. The system collects the results of the pings, including:
    - The number of successful and failed webhook calls.
    - Details about each success or failure.

Example Use Case

Imagine you have registered a webhook URL for the **user_registered** event. You want to verify that your webhook receiver is working correctly. You can use the **/ping** endpoint to send a test payload to your registered URL and check the response.

The following CURL request shows an example of the registration request without a payload:

```
curl -X POST -H "Content-Type: application/json" SERVER_URL/ping
```

And this is a CURL request shows an example of the registration request with a payload:

```
curl -X POST -H "Content-Type: application/json" -d '{"arbitrary_key":"test", "arbitrary_key2":"test2"}' SERVER_URL/ping
```

Example Response

```
{
   "message": "Pinged all registered webhooks successfully.",
   "successful_webhooks_count": 2,
   "successful_webhooks": [
      {
         "event": "user_registered",
         "url": "http://example.com/webhook1",
         "payload_response": {"message": "Webhook received successfully"},
         "status_code": 200
      },
      {
         "event": "user_send_message",
         "url": "http://example.com/webhook2",
         "payload_response": {},
         "status_code": 200
      }
   ],
   "failed_webhooks_count": 1,
   "failed_webhooks": [
      {
         "event": "user_registered",
         "url": "http://example.com/webhook3",
         "error": "Connection refused",
         "status_code": null
      }
   ]
}
```