|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Feature/**  **Interaction** | **Purpose/**  **Description** | **Request Data Source** | **Request send to server** | **Processing needed/response data source** | **Response sent to client** | **Request Tye & Path** | **Other Notes** |
| Filter | Chooses what events the user will see displayed. | User input: Various options such as branch, and RSVP status can be inputted. | JavaScript to hide events/show the filtered ones. Nothing for server. | When receiving the events array from the database, the user needs to be logged in.  See figure 1 for the SQL command.  After the events are collected, the RSVP status must also be collected. Figure 2 shows the SQL required. Depending on the result is whether the RSVP status is set to true, or false. | A JSON array of all events will be loaded using routes and AJAX when the page loads. See figure 4 for an example. | Does not need server for the request, as this is just changing what events are visible to the user. | No extra notes. |
| RSVP Button | A user can RSVP to an event. | Button click on the particular event the user wants to RSVP to. | AJAX request to post that the user has RSVP’d. | Verify that the user is logged in. Then, see figure 3 for the SQL command. | A 200 status if the RSVP change worked, else a 404-error code. | POST request to  /userid/events/eventid | No extra notes. |

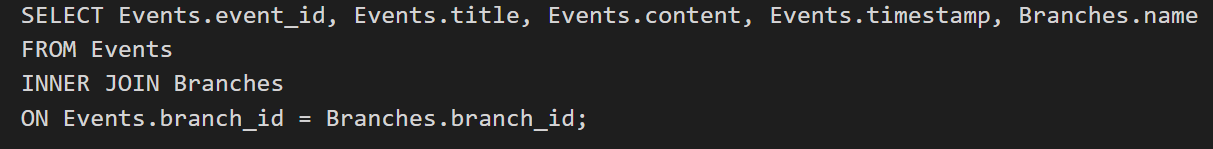


Figure 1

A black background with white text

Description automatically generated

Figure 2 – The user\_id will be from the user (as they need to be logged in). A loop will be used, so the event\_id is known.

A black background with white text

Description automatically generated

Figure 3 – The user\_id is known (as the user needs to be logged in). The event\_id will be found easily when the RSVP button is selected.

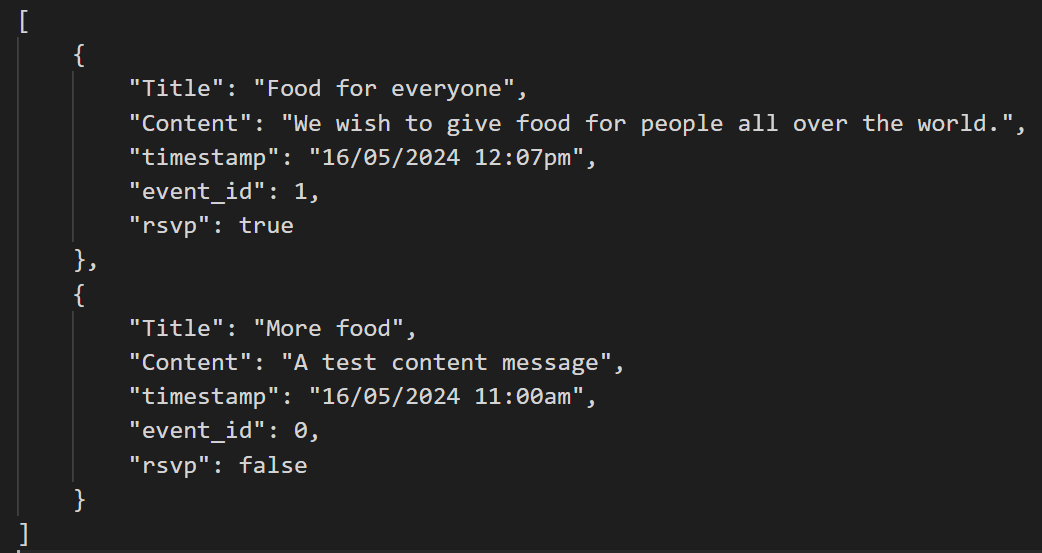


Figure 4 – An example response of events when they are collected from the server.