Roster Database API Specifications

If the server is running locally, it can be accessed at `http://127.0.0.1:8080/`. The port number is configurable at the top of `roster-server.py`. If the server is running on another machine on the LAN, you will need to know that machine's IP address as well.

Request Format

In all cases, the server sends and expects to recieve JSON data. The "content-type" header of all requests that require sending data should be `application/json`. The database tables can be interfaced with at `/api/TABLE_NAME` by sending various HTTP requests.

For example, the members table can be accessed via `http://127.0.0.1:8080/api/members`.

Tables

The database exposes two tables, `members` and `attendance`. Because of the way SQL works, the `attendance` table is actually two tables under the hood, but for our purposes it behaves as if it were one.

The `members` table stores `id: int, name: str, email: str, role: int, note: str`. The ID is the primary key, so no two rows may have the same ID. If you try to POST a new row with an ID that already exists in another row, that will result in an error. By default, the name and email fields are 50 chars, and the note field is 500 chars. This is also configurable at the top of `roster-server.py`.

The `attendance` table stores `date: str, ids: int[]`. Each date must be unique, and each ID must be unique within each of the date's ids list. Note that you aren't intended to manually write a date string. Sending a POST request to this table will automatically create a new row with date set to the current time. Any subsequent POSTs and DELETEs should match that automatically created date string exactly.

GET Requests

A GET request is used to read the database. All other requests will return in this format as well. No data is required to be sent. The server will simply return the table in JSON format:

```
{"members": [{"id": 0, "name": "Julian Williams", "email": "jaw6642@truman.edu",
"role": 0, "note": ""}, {"id": 1, "name": "Justin Caringal",
"email": "jac5566@truman.edu", "role": 1, "note": ""}]}
```

POST Requests

A POST request is used to create a new row in the database. POST requests can fail in multiple ways: - If applicable, the content-type must be application/json (400 Bad Request) - The JSON must contain all expected variables with the appropriate datatypes (400 Bad Request) - The new row's primary key must not conflict with an existing row (409 Conflict)

Example data sent along with a POST request: `{"id": 2, "name": "Andrew Ruff", "email": "abr8115@truman.edu", "role": 1, "note": "Hello, world!"}`

Note that the date is optional when POSTing to the `attendance` table. The database will automatically create a new row and set the `date` field to be the current time if no date is provided. The `ids` list starts empty.

Dates should always be of the format `YYYY-MM-DD HH:mm:ss`. For example `2024-04-08 14:00:22`.

PUT Requests

A PUT request is used to update an existing row in the database. PUT requests can fail in similar ways to POST: - The content-type must be application/json (400 Bad Request) - The JSON must contain all expected variables with the appropriate datatypes (400 Bad Request) - The row to be updated must already exist in the database (409 Conflict)

The data sent and recieved with a PUT request looks exactly the same as a POST request. When sending a PUT to the `members` table, the ID must match an existing member, but all other columns may be different. Those columns will be updated. When sending a PUT to the `attendance` table, an already existing `date` and member `id` must be included, then an attendance record will be created for that member on that date.

DELETE Requests

A DELETE request is used to remove a row from the database. DELETE requests can fail in the following ways: - The content-type must be application/json (400 Bad Request) - The primary key must be included in the request with the correct datatype, other fields are not required (400 Bad Request)

When deleting from the `attendance` table, if only the `date` is specified then all records of that date will be deleted from the database. Similarly, if only the `id` is specified then all attendance records for that member will be deleted. If both the `date` and `id` are sent together, then only that one entry will be removed if it exists.

Note that a DELETE request on a row that doesn't exist has no effect.

Example data for a DELETE request: `{"id": 1}`