**Planned API Calls**

Reference for HTTP Status Codes: https://www.restapitutorial.com/httpstatuscodes.html

Will return 200 on successful call.

Will return 400, 401, 404, or 500 depending on error. If we decide to use more codes, this document will be updated.

**note:** Since we plan to do all of this locally,

url = http://localhost:PortNumber

MySQL default PortNumber = 3306

**Problems:** Currently our DB design doesn’t have unique usernames. I think we decided to check if a username already exists at frontend during registration. We could pass id instead of username or we could use JWT to store id. The current inputs and outputs are temporary and will be updated to reflect how we decide to implement this.

We may still want to have an admin table or view to make some of these API Calls easier and less confusing.

|  |  |
| --- | --- |
| **url/registerUser** | Register a new user |
| HTTP Request | POST |
| Required Inputs | Username and password (should we require email?) |
| Optional Inputs | Name, email, and city |
| Successful Output | Username and message |
| Error Output | message |

|  |  |
| --- | --- |
| **url/loginUser** | Log in an existing user |
| HTTP Request | POST |
| Required Inputs | Username and password |
| Optional Inputs |  |
| Successful Output | token, username, name, email, city, level, and message |
| Error Output | message |

|  |  |
| --- | --- |
| **url/editUser** | Edit existing user data |
| HTTP Request | PUT |
| Required Inputs | Username and password |
| Optional Inputs | NewPassword, name, email, and city |
| Successful Output | Any edited fields (aside from password) and message |
| Error Output | message |

|  |  |
| --- | --- |
| **url/deleteUser** | Delete an existing user from the database |
| HTTP Request | DELETE or POST |
| Required Inputs | Username and password |
| Optional Inputs |  |
| Successful Output | Username and message |
| Error Output | message |

|  |  |
| --- | --- |
| **url/listAdmins** | SuperAdmin lists all admins |
| HTTP Request | POST |
| Required Inputs | Id (id of superadmin) |
| Optional Inputs |  |
| Successful Output | All usernames (corresponding to organizers) and message |
| Error Output | message |

|  |  |
| --- | --- |
| **url/adminEvents** | SuperAdmin lists all events organized by a particular admin |
| HTTP Request | POST |
| Required Inputs | Id (of superadmin) and username (of admin) |
| Optional Inputs |  |
| Successful Output | All event fields and message |
| Error Output | message |

|  |  |
| --- | --- |
| **url/listUsers** | SuperAdmin lists all users |
| HTTP Request | POST |
| Required Inputs | Id (of superadmin) |
| Optional Inputs |  |
| Successful Output | All usernames and message |
| Error Output | message |

|  |  |
| --- | --- |
| **url/userEvents** | SuperAdmin lists titles of all events this user has attended |
| HTTP Request | POST |
| Required Inputs | Id (of superadmin) and username |
| Optional Inputs |  |
| Successful Output | Titles (of events username has participated in) and message |
| Error Output | message |

|  |  |
| --- | --- |
| **url/approveEvent** | SuperAdmin approves an event created by an Admin |
| HTTP Request | PUT |
| Required Inputs | Id (of superadmin) |
| Optional Inputs |  |
| Successful Output | message |
| Error Output | message |

|  |  |
| --- | --- |
| **url/createEvent** | User creates an event and becomes an admin |
| HTTP Request | POST |
| Required Inputs | Username |
| Optional Inputs |  |
| Successful Output | Eventid, title, and message |
| Error Output | message |

|  |  |
| --- | --- |
| **url/editEvent** | Admin edits an event |
| HTTP Request | PUT |
| Required Inputs | Id (of admin) and eventid |
| Optional Inputs |  |
| Successful Output | Any edited fields and message |
| Error Output | message |

|  |  |
| --- | --- |
| **url/listSelfEvents** | Admin lists all events this admin has created (can also be used to filter only active events) |
| HTTP Request | POST |
| Required Inputs | Id (of admin) |
| Optional Inputs |  |
| Successful Output | Titles, urls, active (of events associated with admin Id), and message |
| Error Output | message |

|  |  |
| --- | --- |
| **url/eventsByDate** | List all events within startdate and enddate |
| HTTP Request | GET |
| Required Inputs | Startdate and enddate |
| Optional Inputs |  |
| Successful Output | Titles, urls, and message |
| Error Output | message |

|  |  |
| --- | --- |
| **url/eventsByCity** | List all active events by city |
| HTTP Request | GET |
| Required Inputs | city |
| Optional Inputs |  |
| Successful Output | Titles, urls, and message |
| Error Output | message |

|  |  |
| --- | --- |
| **url/joinEvent** | User joins an event |
| HTTP Request | POST |
| Required Inputs |  |
| Optional Inputs |  |
| Successful Output | message |
| Error Output | message |

|  |  |
| --- | --- |
| **url/leaveEvent** | User leaves an event |
| HTTP Request | DELETE |
| Required Inputs |  |
| Optional Inputs |  |
| Successful Output | message |
| Error Output | message |