# MESGANTRY BACKEND API V1.3 DOCUMENTATION

**Last Updated: 2019/6/20 V1.3 Release**

## Types

All types used in the API are represented as JSON objects. If a field is not set by client, you cannot update it in PATCH requests either (nor supply it in POST requests). Only the fields being updated need to be supplied in PATCH requests. All fields are returned in GET requests unless otherwise specified. Being listed as required means that it has to be supplied in creation endpoints (ie POST requests). Times are in the format YYYY-MM-DDTHH:MM:SSZ (e.g. 2019-10-15T10:04:02Z). They are in UTC by default – see Endpoints > Times and Timezones for how to change this for both POST/PUT/PATCH requests and GET request response bodies.

### User

| Field | Type | Required (For creation) | Validation | Description |
|---|---|---|---|---|
| username | String | Yes | Not null, not empty string | The username used for logging in. Has to be unique. |
| name | String | Yes | Not null, not empty string | The name of the user |
| password | String | Yes | Not null, not empty string | The password, in plaintext, of the user. Not supplied from the server, only posted *to* the server. |
| createdAt | Time | Not set by client | Not null | The time at which the user was created |
| updatedAt | Time | Not set by client | Not Null | The time of the last update to this user’s details. Set to the creation time if never updated |
| lastLoggedIn | Time | Not set by client | Nullable | The last time the user logged in. Null if the user never logged in |

## Event

| Field | Type | Required (For creation) | Validation | Description |
|---|---|---|---|---|
| eventId | String (UUID) | Not set by client | Not null, not empty string | The V4 UUID of the event, guaranteed to be unique |
| name | String | Yes | Not null, not empty string, max length 64 bytes | The name of the event |
| url | String | No | Nullable, not empty string if not null, max length 128 bytes | The URL at which the event's website is hosted. Has to be unique. |
| triggers | Map of String to Time | No | Nullable, but will always be empty map if GET from the database (i.e. will be set to empty map if not supplied or supplied as null) | A map of trigger names to when they trigger. Stores time based events. For example, the "release" trigger can be set to 2019-06-01T10:30:00Z, and then an endpoint can be used to check if this trigger has been passed.<br><br>e.g.<br>{<br>"release": "2019-06-01T10:30:00Z"<br>"formrelease": "2019-06-01T12:00:00Z"<br>} |
| startDateTime | Time | No | Nullable | The time of the start of the event. |
| endDateTime | Time | No | Nullable | The time of the end of the event |

| Field | Type | Required (For creation) | Validation | Description |
|---|---|---|---|---|
| lat | Decimal | No | Nullable | The latitude of the event location (for geolocation based check in) |
| long | Decimal | No | Nullable | The longitude of the event (for geolocation based check in) |
| radius | Decimal | No | Nullable | The radius, in KM, of the circular geofence around the event longitude and latitude around which guests are able to check in (for geolocation based check in) |
| updatedAt | Time | Not set by client | Not null | The last time this object was updated – set to createdAt time if it has never been updated |
| createdAt | Time | Not set by client | Not null | The time this object was created, |

## Guest

| Field | Type | Required (For creation) | Validation | Description |
|---|---|---|---|---|
| nric | String | Yes | Not null | A **unique** identifier for the guest. Can technically be an empty string, for one guest. |
| name | String | Yes | Not null | The name of the guest. Can be an empty string. |
| Tags | Array of Strings | No | Can be null when supplied, | A list of tags (categories) |

| | | | | |
|---|---|---|---|---|
| | | | will be empty array when requested. | that the guest fits into. For example, “VIP” can be a tag. There are endpoints to filter guests by tags. |

# Endpoints

### How to read the table

An endpoint refers to the bit after the website name, where the back end is hosted. If the backend is hosted on mesgantry-backend.herokuapp.com, then in order to, say, log in as an administrator (the first row of the table), make a POST request (where POST is the method) to https://mesgantry-backend.herokuapp.com/api/v0/auth/admins/login, with the HTTPS request body being as specified in the Input/Output columns.

### Permissions

Permissions are handled using JSON Web Tokens. Upon logging in, a signed web token (a string) is given to the caller, with an expiry time (such as 24 hours from issuing; set by environment variable). In all future calls which have any permissions (see the permissions section), a valid web token is required, and contains the information required to check said permissions. The web token must be provided in the headers of the HTTP request, under the Authorization header (exact spelling), in the format of Bearer {{token}} (where {{token}} is replaced by the web token previously obtained). Note the space. Anything but this exact format (including a missing or blank Authorization header) will return a Status 400 Bad Request error message.

If an invalid (ie expired) web token is provided, a Status 401 Unauthorized is returned.

If a valid web token is provided, but it does not have authorization to access the given endpoint, a Status 403 Forbidden is returned.

**The Errors section in the tables ignores this possible 400/401/403, it is implicit in any permissions requirement**

### Times and Timezones

Any API endpoint that requires times to be supplied in the request body assumes that the time provided is in the format <YYYY-MM-DD>T<HH:MM:SS>Z (2019-03-02T18:04:00Z for example) and refers to UTC time. If the time is in another timezone, use the *loc* query string parameter with an IANA time zone database name to specify which. For example if posting an event with a release time in Singapore time (GMT +8), post to /api/v1-3/events?loc=Asia/Singapore. All times in the request body should be in the same time zone.

Any API endpoint that returns a time in the response body (say, an array of events, or a single user object) can also be supplied a *loc* query string parameter with an IANA time zone database name to change all times returned from UTC to whatever time zone is desired. If the timezone supplied can’t be found, a Status 400 Bad Request is returned.

**The Errors section in the tables ignores this possible 400**

**Struct Field Selection**

Any endpoint which returns structs or an array of structs will, by default, return all fields in the struct. Query string parameters called *field* can be specified to specify the exact fields of the struct that are desired. For example, GET /api/v0/events?field=name&field=username returns an array of all users, but only their name and username. Supplying non existent fields doesn’t throw an error.

## Authentication

| URL Endpoint | Method | Input/Output (Request Body/Response Body) | Permissions | Description |
|---|---|---|---|---|
| /api/v0/auth/admins/login | POST | **Input**<br>{“username”: “abc”, “password”: “abc”}<br><br>**Output (Successful)**<br>Status 200<br>{“accessToken”: “eyzbdshdbsyd256”}<br><br>**Errors**<br>Error 400 if JSON body not parseable<br>Error 401 if login details incorrect (username or password wrong)<br>Error 500 if any internal server issues | None | Logging in as an adminstrator. The accessToken needs to be saved and uses in all future requests which need permissions of any kind. Such requests need to have a header called **Authorization** with value<br>**Bearer {{accessToken}}**<br>Note the space between |
| /api/v0/auth/users/login | POST | **Input**<br>{“username”: “abc”, “password”: “abc”}<br><br>**Output (Successful)**<br>Status 200<br>{“accessToken”: “eyzbdshdbsyd256”} | None | Logging in as a user. The accessToken needs to be saved and uses in all future requests which need permissions of any kind. Such requests need to have a header called |

| | | | | |
|---|---|---|---|---|
| | | **Errors**<br>401 if login details incorrect (username or password wrong)<br>400 if JSON body not parseable<br>500 if any internal server issues | | **Authorization** with value<br>**Bearer {{accessToken}}**<br>Note the space between |
| /api/v1-3/auth/verify | POST | **Input**<br>None, reads from Authorization header instead<br><br>**Output**<br>“true” if token in Authorization header is valid (unexpired, properly formatted)<br>“false” if token in Authorization header is invalid<br><br>**Errors**<br>400 if Authorization header is not in Bearer {{accessToken}} format | None | Checks whether the http request, with an Authorization header of values Bearer {{accessToken}}, has a valid access token. |

## Users

| URL Endpoint | Method | Input/Output (Request Body/Response Body) | Permissions | Description |
|---|---|---|---|---|
| /api/v0/users | GET | **Input**<br>None<br><br>**Query String Arguments**<br>*loc* to specify the time zone of the times in the output<br>*field* to select only some fields of the struct to render.<br>See introduction of document for more information on these two arguments.<br><br>**Output (Successful)**<br>Status 200<br>An array of user objects | Valid access token<br>Administrator account | Returns a JSON array of all users |

| URL Endpoint | Method | Input/Output (Request Body/Response Body) | Permissions | Description |
|---|---|---|---|---|
| | | **Errors**<br>500 if any internal server issues | | |
| /api/v0/users | POST | **Input**<br>A user object, like:<br>{“name”: “Ashwin”, “username”: “Hackerboy”, “password”: “unsecured”}<br><br>Supplying any other fields will cause an error. Username, name and password cannot be empty strings.<br><br>**Output (Successful)**<br>Status 201<br>{"message":"Registration successful"}<br><br>**Errors**<br>400 if JSON body not parseable, unknown fields supplied, or blank username/password/name set<br>409 if user with that username already exists<br>500 if any internal server issue | Valid access token<br>Administrator account | Creates a new user, will automatically populate the createdAt and updatedAt fields to the time of creation. |
| /api/v0/users/{username} | GET | **Input**<br>None<br><br>**Query String Arguments**<br>*loc* to specify the time zone of the times in the output<br>*field* to select only some fields of the struct to render.<br>See introduction of document for more information on these two arguments.<br><br>**Output (Successful)**<br>Status 200<br>A user object with the given username | Valid access token<br>One of the following:<br>Administrator account<br>User with the given username (you can look at your own details) | Returns a single user JSON object based on the username in the URL.<br>The user with the {username} needs to exist |

| URL Endpoint | Method | Input/Output (Request Body/Response Body) | Permissions | Description |
|---|---|---|---|---|
| | | **Errors**<br>404 if the user doesn’t exist<br>500 if any internal server issues | | |
| /api/v0/users/{username} | PATCH | **Input**<br>Supply **only** the fields of the user object you wish to update, with the new values e.g.<br>{"username":"LCPSANIL",<br>"name":"Sanil","password":"AbetterPassword"}<br><br>Note that you can update the username. Cannot update fields not set by user.<br><br>**Output (Successful)**<br>Status 200<br>{"message":"User updated"}<br><br>**Errors**<br>400 if JSON body not parseable, unknown fields supplied or blank name/password/username set<br>400 if invalid field is specified<br>404 if the user does not exist<br>409 if newly specified username is already in username<br>500 if any internal server issues | Valid access token<br>One of the following:<br>Administrator account<br>User with the given username (you can update your own details) | Update a user’s details. You only need to give the fields that need changing. The user given by {username} needs to exist. |
| /api/v0/users/{username} | DELETE | **Input**<br>None<br><br>**Output (Successful)**<br>Status 200<br>{"message":"Successfully deleted user"} | Valid access token<br>One of the following:<br>Administrator account<br>User with the given username (you can delete yourself) | Deletes the user from the database. |

| URL Endpoint | Method | Input/Output (Request Body/Response Body) | Permissions | Description |
|---|---|---|---|---|
| | | **Errors**<br>404 if the user does not exist<br>500 if any internal server issues | | |

**Events**

| URL Endpoint | Method | Input/Output (Request Body/Response Body) | Permissions | Description |
|---|---|---|---|---|
| /api/v1-3/events | GET | **Input**<br>None<br><br>**Query String Arguments**<br>*loc* to specify the time zone of the times in the output<br>*field* to select only some fields of the struct to render. See introduction of document for more information on these two arguments.<br><br>**Output (Successful)**<br>An array of event objects<br><br>**Errors**<br>500 if any internal server issue | Valid access token | Returns all the events hosted by the **user making the request** (the user is identified using the web token), as a JSON array. |
| /api/v1-3/events | POST | **Input**<br>An event object<br><br>**Query String Arguments**<br>*loc* to specify the time zone of the times supplied in the input.<br><br>**Output (Successful)**<br>Status 201<br>“5b7e451d-249d-4545-9c44-fff028127d53” | Valid access token | Creates a new event with the given fields. Note that the URL of the event is either an external URL address or can meant to direct to an endpoint |

| URL Endpoint | Method | Input/Output (Request Body/Response Body) | Permissions | Description |
|---|---|---|---|---|
| | | The UUID of the newly generated event.<br><br>**Errors**<br>400 if invalid field values supplied, or if badly formatted time string, or request body could not be read as JSON.<br>409 if event with the given URL already exists<br>500 if any internal server issues<br>**As of v1.3, admin account trying to create an event will not be blocked by permissions, but will produce an error 500**. | | |
| /api/v0/events/takenurls/{eventURL} | GET | **Input**<br>None<br><br>**Output (Successful)**<br>Status 200<br>“true” or “false”<br>Note that a URL being taken (IE “true”) means that it is *not* available for use.<br><br>**Errors**<br>500 if any internal server issues. | Valid access token | Checks if an event already exists with that given **URL**, **(not event ID!)** |
| /api/v1-3/events/id/{eventURL} | GET | **Input**<br>None<br><br>**Output**<br>Status 200<br>“7bb38ef0-0aae-44d5-92a0-aa31606e5878”<br><br>**Errors**<br>404 if no event exists with that URL<br>500 if there is an internal server error. | None | Returns the ID of an event which has the given URL. |

| URL Endpoint | Method | Input/Output (Request Body/Response Body) | Permissions | Description |
|---|---|---|---|---|
| /api/v1-3/events/{eventID} | GET | **Input**<br>None<br><br>**Output (Successful)**<br>Status 200<br>The event object with that<br><br>**Errors**<br>404 if an event with that eventID doesn't exist<br>500 if any internal server issues | Valid access token<br>One of the following:<br>User account which is a host of that event<br>Administrator account | Returns the details of a specific event given its event ID. |
| /api/v1-3/events/{eventID} | PATCH | **Input**<br>Supply **only** the fields of the user you wish to update, with the new values e.g.<br>{"name": "abc", "startDateTime": "2019-01-08T14:00:00Z", "radius": 5}<br><br>**Output (Successful)**<br>Status 200<br>{"message":"Event updated"}<br><br>**Errors**<br>400 if body could not be read as JSON, or blank URL/name supplied, or invalid fields supplied, or badly formatted time fields.<br>404 if an event with that eventID does not exist.<br>409 if updated URL is already used by another event.<br>500 if any internal server issue | Valid access token<br>One of the following:<br>User account which is a host of that event<br>Administrator account | Updates the details of the event. You only need to give the fields that need changing. The event given by {eventID} needs to exist. |
| /api/v0/events/{eventID} | DELETE | **Input**<br>None<br><br>**Output (Successful)**<br>{"message":"Successfully deleted event"} | Valid access token<br>One of the following:<br>User account which is a host of that event<br>Administrator | Deletes the event with the given eventID. |

| URL Endpoint | Method | Input/Output (Request Body/Response Body) | Permissions | Description |
|---|---|---|---|---|
| | | **Errors**<br>404 if an event with that eventID does not exist.<br>500 if any internal server issue | account | |
| /api/v0/events/{eventID}/released<br><br>**DEPRECATED, use /api/v1-3/events/{eventID}/triggers/release/occurred if the event has a trigger called release; although this endpoint returns a 404 if no release is set.** | GET | **Input**<br>None<br><br>**Output (Successful)**<br>“true” if the event is open for check in, “false” if it is not.<br><br>**Errors**<br>404 if an event with that eventID does not exist.<br>500 if any internal server issue. | None | Checks if the current Singapore Time is past the release date/time of the event. If no release date/time is set, the event is assumed to always be open for check in, and therefore this endpoint will return true.<br>Release time is in UTC. |
| /api/v1-3/events/{eventID}/triggers/{triggername} | GET | **Input**<br>None<br><br>**Query String Arguments**<br>*loc* to specify the time zone of the time returned<br>**Output (Successful)**<br>Status 200<br>“2019-03-01T11:30:00Z”<br><br>**Errors**<br>404 if event with that ID does not exist or no trigger exists with that name<br>500 if internal server issue | None | Returns the time that a trigger with that given name is associated with in the event with that ID. |
| /api/v1-3/events/{eventID}/triggers/{triggername}/occurred | GET | **Input**<br>None<br><br>**Output (Successful)** | None | Returns true or false whether the given trigger of the given event, if it exists, has passed (based |

| URL Endpoint | Method | Input/Output (Request Body/Response Body) | Permissions | Description |
|---|---|---|---|---|
| | | Status 200<br>“true” or “false”<br><br>**Errors**<br>404 if event with that ID does not exist or no trigger exists with that name<br>500 if internal server issue | | on server time). |
| /api/v1-2/events/{eventID}/feedback | POST | **Input**<br>{"name":"Jimothy Bob","survey":<br>[{"question":"A","answer":"AA"},<br>{"question":"B","answer":"BB"},...]}<br><br>Survey cannot be null or an empty array, name can empty string (or omitted) for anonymous submissions. Forms can have different questions from each other.<br><br>**Output (Successful)**<br>Status 200<br><br>**Errors**<br>400 if null/empty survey or extra fields supplied<br>404 if that event does not exist<br>500 if internal server issue. | None | Submits the given feedback form for the given event. Unrestricted submission by the way, **this could be an issue with spam submissions**. |
| /api/v1-2/events/{eventID}/feedback/report | GET | **Input**<br>None<br><br>**Output (Successful)**<br>CSV file. The first row will have the cell “Name” followed by all the unique questions asked across all the feedback forms submitted. Then for each form submitted, the name of the submitted (possibly an empty string) will appear under name, and answers to questions they did | Valid access token<br>One of the following:<br>User account which is a host of that event<br>Administrator account | Generates a CSV file with all the feedback forms submitted. |

| URL Endpoint | Method | Input/Output (Request Body/Response Body) | Permissions | Description |
|---|---|---|---|---|
| | | answer will appear under the column for that question (empty string for unanswered questions/questions not asked).<br><br>**Errors**<br>404 if that event does not exist<br>500 if internal server issue | | |

## Guests

| URL Endpoint | Method | Input/Output (Request Body/Response Body) | Permissions | Description |
|---|---|---|---|---|
| /api/v0/events/{eventID}/guests | GET | **Input**<br>None<br><br>**Query String Arguments**<br>*checkedin*, which can be true or false (not case sensitive). Will return only checked in (or not checked in) guests.<br>*tag*, which can be any string. Will return only guests which have that tag. Multiple tags allowed, by reusing the tag form.<br>No query string arguments will just return all guests. Arguments are not case sensitive.<br>For example:<br>/api/v0/event/{eventID}/guests for all guests<br>/api/v0/event/{eventID}/guests?tag=VIP for guests tagged VIP<br>/api/v0/event/{eventID}/guests?tag=VIP&tag=ATTENDING for all guests who are VIP and ATTENDING<br>/api/v0/event/{eventID}/guests?checkedin=true for all guests who are checked in | Valid access token<br>One of the following:<br>User account which is a host of that event<br>Administrator account | Gets the list, as a JSON array, of names of all guests attending that particular event, subject to the tag and checkedin constraints that are provided. Event with that ID must exist. |

| URL Endpoint | Method | Input/Output (Request Body/Response Body) | Permissions | Description |
|---|---|---|---|---|
| | | /api/v0/event/{eventID}/guests?checkedin=false&tag=OFFICER for all guests who are not checked in and are tagged OFFICER.<br><br>**Output (Successful)**<br>Status 200<br>["PERSON MCPERSON","CHINESE ENGLISHNAME",…]<br><br>Note that if you ask for a tag that does not exist, you'll get successful output of []<br><br>**Errors**<br>400 if any of the query string arguments are badly formatted<br>404 if an event with that eventID does not exist.<br>500 if any internal server issue | | |
| /api/v0/events/{eventID}/guests | POST | **Input**<br>{"nric": "1234F", "name": "BOB MCBOB", "tags": ["VIP", "ATTENDING"]}<br><br>tags can be omitted, or empty array, for no tags. Name and tags cannot be longer than 64 bytes.<br><br>**Output (Successful)**<br>Status 201<br>{"message":"Registration successful"}<br><br>**Errors**<br>400 if body could not be read as JSON<br>404 if an event with that eventID does not exist.<br>500 if any internal server issue | Valid access token<br>One of the following:<br>User account which is a host of that event<br>Administrator account | Registers a guest as attending a given event. NRIC must be unique (for that particular event). Event with that ID must exist. Technically a blank NRIC and blank name is acceptable. |

| URL Endpoint | Method | Input/Output (Request Body/Response Body) | Permissions | Description |
|---|---|---|---|---|
| /api/v1-3/events/{eventID}/guests | POST | **Input**<br>An array of Guest objects<br>[{“nric”: “1234F”, “name”: “BOB MCBOB”, “tags”: [“VIP”, ”ATTENDING”]}, ...]<br><br>**Output (Successful)**<br>Status 201<br>{“message”: “Registration successful for all guests”}<br><br>**Errors**<br>400 if body could not be read as JSON, or empty array of guests is provided<br>404 if an event with that eventID does not exist.<br>500 if any internal server issue | Valid access token<br>One of the following:<br>User account which is a host of that event<br>Administrator account | Registers all of the guests supplied for a given event. NRIC must be unique (for that particular event). Event with that ID must exist.<br>Technically on guest with a blank NRIC is permitted, as are blank names.<br>If *any* of the guests fail to register, all will fail. |
| /api/v0/events/{eventID}/guests | DELETE | **Input**<br>{“nric”: “1234F”}<br><br>**Output (Successful)**<br>Status 200<br>{"message":"Successfully deleted guest"}<br><br>**Errors**<br>400 if body could not be read as JSON<br>404 if an event with that eventID does not exist.<br>500 if any internal server issue | Valid access token<br>One of the following:<br>User account which is a host of that event<br>Administrator account | Removes a guest (given their nric) from the guest list of a given event. Event with that ID must exist. |
| /api/v0/events/{eventID}/guests/checkedin<br><br>**DEPRECATED, use GET api/v0/events/{eventID}/guests?checkedin=true** | GET | **Input**<br>None<br><br>**Output (Successful)**<br>Status 200<br>["PERSON MCPERSON","CHINESE ENGLISHNAME",…] | Valid access token<br>One of the following:<br>User account which is a host of that event<br>Administrator account | Gets the list, as a JSON array, of names of guests who have checked in to (i.e. attended) a given event. Event with that ID must exist. |

| URL Endpoint | Method | Input/Output (Request Body/Response Body) | Permissions | Description |
|---|---|---|---|---|
| | | **Errors**<br>404 if an event with that eventID does not exist.<br>500 if any internal server issue | | |
| /api/v0/events/{eventID}/guests/checkedin | POST | **Input**<br>{“nric”: “1234F”}<br><br>**Output (Successful)**<br>Status 200<br>“BOB MCBOB”<br><br>**Errors**<br>400 if body could not be read as JSON<br>404 if an event with that eventID does not exist or if guest with that NRIC does not exist (and thus cannot be checked in).<br>500 if any internal server issue | None | Checks in a guest with the provided NRIC into the given event, given that the guest is in the guest list to start with (if not an error is returned). Returns a string with their name. Event with that ID must exist. If a guest is already checked in, no error is returned and the name is returned as usual, as if the guest was not already checked in. |
| /api/v0/events/{eventID}/guests/checkedin | DELETE | **Input**<br>{“nric”: “1234F”}<br><br>**Output (Successful)**<br>Status 200<br><br>**Errors**<br>400 if body could not be read as JSON<br>404 if an event with that eventID does not exist or if guest with that NRIC does not exist (and thus cannot be checked in).<br>500 if any internal server issue | | Checks out a guest with the provided NRIC/marks them as absent for the given event, gien that the guest is in the guest list to start with (if not an error is returned). Event with that ID must exist. If the guest is already not checked in/marked as absent, no error is returned, as if the guest |

| URL Endpoint | Method | Input/Output (Request Body/Response Body) | Permissions | Description |
|---|---|---|---|---|
| | | | | was initially checked in. |
| /api/v0/events/{eventID}/guests/notcheckedin<br><br>**DEPRECATED, use GET api/v0/events/{eventID}/guests?checkedin=false** | GET | **Input**<br>None<br><br>**Output (Successful)**<br>Status 200<br>["PERSON MCPERSON","CHINESE ENGLISHNAME",…]<br><br>**Errors**<br>404 if an event with that eventID does not exist.<br>500 if any internal server issue | Valid access token<br>One of the following:<br>User account which is a host of that event<br>Administrator account | Gets the list, as a JSON arrays, of names of guests who have yet to/did not check in to (i.e. attend) a given event. Event with that ID must exist.<br>**Marked for deletion.** |
| /api/v0/events/{eventID}/guests/stats<br><br>**DEPRECATED, use GET api/v0/events/{eventID}/guests with checkedin=true, checkedin=false and no arguments and calculate stats yourself** | GET | **Input**<br>None<br><br>**Output (Successful)**<br>Status 200<br>{"total":335,"checkedIn":1,"percentCheckedIn":0.0029850746268656717}<br><br>If total=0, checkedIn=0, percentCheckedIn will be 0.<br><br>**Errors**<br>404 if an event with that eventID does not exist.<br>500 if any internal server issue | Valid access token<br>One of the following:<br>User account which is a host of that event<br>Administrator account | Gets summary statistics about the attendance of a given event. The statistics are the number who have checked in, the total number of guests, and the percentage attendance. Event with that ID must exist. |
| /api/v0/events/{eventID}/guests/report | GET | **Input**<br>None<br><br>**Output (Successful)**<br>Returns a CSV file (with header Content-Type text/csv). The first row has two cells, “Name” and “Present” | Valid access token<br>One of the following:<br>User account which is a host of that event<br>Administrator account | Gets a CSV file that serves as an attendance report of the given event. The file is formatted with the names of guests followed by a “1” or “0”, |

| URL Endpoint | Method | Input/Output (Request Body/Response Body) | Permissions | Description |
|---|---|---|---|---|
| | | Each row has a guest name, and then 1 or 0 in the Present column. 1 means present.<br><br>**Errors**<br>404 if an event with that eventID does not exist.<br>500 if any internal server issue | | with 1 indicating that that guest attended, and 0 indicating that they did not. Event with that ID must exist. |

## Utility

| URL Endpoint | Method | Input/Output (Request Body/Response Body) | Permissions | Description |
|---|---|---|---|---|
| /api/v0/utility/qrcode | POST | **Input**<br>{“nric” : “1234F”}<br><br>**Output (Successful)**<br>A PNG image (with header Content-Type image/png) of a QR code that encodes the NRIC supplied.<br><br>**Errors**<br>400 if body could not be read as JSON<br>500 if any internal server issue | None | Generates and returns the QR code that encodes the provided NRIC in it. The QR code is in a PNG format. |
| /api/v1-3/utility/time | GET | **Input**<br>None<br><br>**Query String Arguments**<br>*loc* to specify the timezone the response time should be in<br><br>**Errors**<br>None (except the 400 for invalid timezone name, mentioned in the section Times and Timezones) | None | Returns the current server time, in UTC by default, unless a *loc* argument is supplied. |

## Websocket endpoints

Websocket endpoints should be called using wss://<host>.com/api/whatever. In Javascript, the new Websocket(addr) method should be used. Google about the Javascript websocket API.

## Guests

| URL Endpoint | Method | Input/Output (Request Body/Response Body) | Permissions | Description |
|---|---|---|---|---|
| /api/v1-2/events/{eventID}/guests/checkedin/listener/{nric} | | **Input**<br>None<br><br>**Output**<br>Status 101 Protocol Changed<br><br>**Websocket Output**<br>All output will be in the form<br>{“title”: “sometitle”, “content”: {}}<br><br>Where {} can be any object (or a string, number, array etc) – you should figure out exactly what by looking at the title.<br><br>Two possible results:<br><br>The guest was marked present (checked in):<br><br>{“title”: “checkedin/1”, “content”:{“name”: “Jim”, “nric”: ”1234A”}}<br><br>The guest was marked absent :<br><br>{“title”: “checkedin/0”,”content”:null} | None | Opens a websocket connection with the server, to “listen” on the checked in status of a guest with the given NRIC in the URL (using wss so the query string is not exposed, so using NRIC like this is secure). If that guest is checked in, or marked absent, a message will be sent over the websocket connection. |