MESGANTRY BACKEND API V1.1 DOCUMENTATION

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 JavaScript 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

Authentication

URL Endpoint	Method	Input/Output (Request Body/Response Body)	Permissions	Description
/api/v0/auth/admins/login	POST	**Input** {“username”: “abc”, “password”: “abc”} **Output (Successful)** Status 200 {“accessToken”: “eyzbdshdbsyd256”} **Errors** Error 400 if JSON body not parseable	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

		Error 401 if login details incorrect (username or password wrong) Error 500 if any internal server issues		value **Bearer {{accessToken}}** Note the space between
/api/v0/auth/users/login	POST	**Input** {“username”: “abc”, “password”: “abc”} **Output (Successful)** Status 200 {“accessToken”: “eyzbdshdbsyd256”} **Errors** 401 if login details incorrect (username or password wrong) 400 if JSON body not parseable 500 if any internal server issues	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 **Authorization** with value **Bearer {{accessToken}}** Note the space between

Users

URL Endpoint	Method	Input/Output (Request Body/Response Body)	Permissions	Description
/api/v0/users	GET	**Input** None **Output (Successful)** Status 200 [{“username”: “abc”, “name”: “abc”, “createdAt”: “2019-01-31T14:07:30.796471Z", “updatedAt”: “2019-01-31T14:07:30.796471Z”, “lastLoggedIn”:null}, ...] All fields must be non-null except lastLoggedIn **Errors** 500 if any internal server issues	Valid access token Administrator account	Returns a JSON array of all users

URL Endpoint	Method	Input/Output (Request Body/Response Body)	Permissions	Description
/api/v0/users	POST	**Input** {“name”: “Ashwin”, “username”: “Hackerboy”, “password”: “unsecured”} The other user fields will be ignored if supplied **Output (Successful)** Status 201 {"message":"Registration successful"} **Errors** 400 if JSON body not parseable 409 if user with that username already exists 500 if any internal server issue	Valid access token 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** None **Output (Successful)** Status 200 {“username”: “abc”, “name”: “abc”, “createdAt”: “2019-01-31T14:07:30.796471Z", “updatedAt”: “2019-01-31T14:07:30.796471Z”, “lastLoggedIn”:null} **Errors** 404 if the user doesn’t exist 500 if any internal server issues	Valid access token One of the following: Administrator account 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. The user with the {username} needs to exist
/api/v0/users/{username}	PATCH	**Input** Supply **only** the fields of the user you wish to update, with the new values e.g. {"username":"LCPSANIL", "name":"Sanil","password":"AbetterPassword"}	Valid access token One of the following: Administrator account User with the given username (you can	Update a user’s details. You only need to give the fields that need changing. The user given by {username} needs to exist.

URL Endpoint	Method	Input/Output (Request Body/Response Body)	Permissions	Description
		Note that you can update the username. Above are the only valid update fields (creation/update/last log in time are not valid). **Output (Successful)** Status 200 {"message":"User updated"} **Errors** 400 if JSON body not parseable 400 if invalid field is specified 404 if the user does not exist 409 if newly specified username is already in username 500 if any internal server issues	update your own details)	
/api/v0/users/{username}	DELETE	**Input** None **Output (Successful)** Status 200 {"message":"Successfully deleted user"} **Errors** 404 if the user does not exist 500 if any internal server issues	Valid access token One of the following: Administrator account User with the given username (you can delete yourself)	Deletes the user from the database.

Events

URL Endpoint	Method	Input/Output (Request Body/Response Body)	Permissions	Description
/api/v0/events	GET	**Input** None	Valid access token	Returns all the events hosted by the **user**

URL Endpoint	Method	Input/Output (Request Body/Response Body)	Permissions	Description
		Output (Successful) [{"eventId":"1004ad4d-685c-40a0-b6d1-be1b28cb9a4b","name":"New Event","startDateTime":null,"endDateTime":null,"lat":null,"long":null,"radius":null,"url":"newevent2","updatedAt":"2019-03-07T09:25:43.240095Z","createdAt":"2019-03-07T09:25:43.240095Z"}, …] The nullable fields above are listed as null. Radius is in KM. **Errors** 500 if any internal server issue		**making the request** (the user is identified using the web token), as a JSON array.
/api/v0/events	POST	**Input** {“name”:”New Event”, “url”: “blah”, “startDateTime”: “2019-01-08T14:00:00Z”} The name and url cannot be empty string. All other fields are optional. See GET /api/v0/events for the full list of possible fields which can be supplied in creation. Note that updatedAt/createdAt are not supposed to be supplied. If startDateTime/endDateTime is supplied, it MUST be in the format above for startDateTime (cannot drop out the seconds). Radius is in KM. **Output (Successful)** Status 201 {"message":"Event created"}	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
		Errors 400 if invalid field values supplied, or if badly formatted time string, or request body could not be read as JSON. 409 if event with the given URL already exists 500 if any internal server issues **As of v1.1, admin account trying to create an event will not be blocked by permissions, but will produce an error 500**.		
/api/v0/events/exists/{eventURL}	GET	**Input** None **Output (Successful)** Status 200 {“available”: “true”} or {“available”: “false”} **Errors** 500 if any internal server issues.	Valid access token	Checks if an event already exists with that given **URL**, **(not event ID!)**
/api/v0/events/{eventID}	GET	**Input** None **Output (Successful)** Status 200 {"eventId":"1004ad4d-685c-40a0-b6d1-be1b28cb9a4b","name":"New Event","startDateTime":null,"endDateTime":null,"lat":null,"long":null,"radius":null,"url":"newevent2","updatedAt":"2019-03-07T09:25:43.240095Z","createdAt":"2019-03-07T09:25:43.240095Z"} The null fields above are the only nullable ones. URL/Name cannot be empty strings. Event ID is unique.	Valid access token One of the following: User account which is a host of that event Administrator account	Returns the details of a specific event given its event ID.

URL Endpoint	Method	Input/Output (Request Body/Response Body)	Permissions	Description
		Radius is in KM. **Errors** 404 if an event with that eventID doesn’t exist 500 if any internal server issues		
/api/v0/events/{eventID}	PATCH	**Input** Supply **only** the fields of the user you wish to update, with the new values e.g. {“name”: “abc”, “startDateTime”: “2019-01-08T14:00:00Z”, “radius”: 5} See GET /api/v0/events/{eventID} for the list of fields. Note that all fields listed in GET /api/v0/events/{eventID} can be updated, EXCEPT eventId and the updatedAt/createdAt timings. If URL or name is updated, updated value cannot be blank. Note that time formatting should follow the format for startDateTime above for most predictable behavior, **but other formats will not be rejected**. Radius is in KM. **Output (Successful)** Status 200 {"message":"Event updated"} **Errors** 400 if body could not be read as JSON, or blank URL/name supplied, or invalid fields supplied. 404 if an event with that eventID does not exist. 409 if updated URL is already used by another event.	Valid access token One of the following: User account which is a host of that event Administrator account	Updates the details of the event. You only need to give the fields that need changing. The user given by {username} needs to exist.

URL Endpoint	Method	Input/Output (Request Body/Response Body)	Permissions	Description
		500 if any internal server issue		
/api/v0/events/{eventID}	DELETE	**Input** None **Output (Successful)** {"message":"Successfully deleted event"} **Errors** 404 if an event with that eventID does not exist. 500 if any internal server issue	Valid access token One of the following: User account which is a host of that event Administrator account	Deletes the event with the given eventID.

Guests

URL Endpoint	Method	Input/Output (Request Body/Response Body)	Permissions	Description
/api/v0/events/{eventID}/guests	GET	**Input** None **Output (Successful)** Status 200 ["PERSON MCPERSON","CHINESE ENGLISHNAME",…] **Errors** 404 if an event with that eventID does not exist. 500 if any internal server issue	Valid access token One of the following: User account which is a host of that event Administrator account	Gets the list, as a JSON array, of names of all guests attending that particular event. Event with that ID must exist.
/api/v0/events/{eventID}/guests	POST	**Input** {“nric”: “1234F”, “name”: “BOB MCBOB”} **Output (Successful)** Status 201 {"message":"Registration successful"}	Valid access token One of the following: User account which is a host of that event 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

URL Endpoint	Method	Input/Output (Request Body/Response Body)	Permissions	Description
		Errors 400 if body could not be read as JSON 404 if an event with that eventID does not exist. 500 if any internal server issue		NRIC and blank name is acceptable.
/api/v0/events/{eventID}/guests	DELETE	**Input** {“nric”: “1234F”} **Output (Successful)** Status 200 {"message":"Successfully deleted guest"} **Errors** 400 if body could not be read as JSON 404 if an event with that eventID does not exist. 500 if any internal server issue	Valid access token One of the following: User account which is a host of that event 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	GET	**Input** None **Output (Successful)** Status 200 ["PERSON MCPERSON","CHINESE ENGLISHNAME",…] **Errors** 404 if an event with that eventID does not exist. 500 if any internal server issue	Valid access token One of the following: User account which is a host of that event 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.
/api/v0/events/{eventID}/guests/checkedin	POST	**Input** {“nric”: “1234F”} **Output (Successful)**	None	Checks in a guest with the provided NRIC into the given event, given that the guest is in the

URL Endpoint	Method	Input/Output (Request Body/Response Body)	Permissions	Description
		Status 200 “BOB MCBOB” **Errors** 400 if body could not be read as JSON 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). 500 if any internal server issue		guest list to start with (if not an error is returned). Returns a string with their name. Event with that ID must exist.
/api/v0/events/{eventID}/guests/notcheckedin	GET	**Input** None **Output (Successful)** Status 200 ["PERSON MCPERSON","CHINESE ENGLISHNAME",…] **Errors** 404 if an event with that eventID does not exist. 500 if any internal server issue	Valid access token One of the following: User account which is a host of that event 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.
/api/v0/events/{eventID}/guests/stats	GET	**Input** None **Output (Successful)** Status 200 {"total":335,"checkedIn":1,"percentCheckedIn":0.0029850746268656717} If total=0, checkedIn=0, percentCheckedIn will be 0. **Errors** 404 if an event with that eventID does not exist.	Valid access token One of the following: User account which is a host of that event 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.

URL Endpoint	Method	Input/Output (Request Body/Response Body)	Permissions	Description
		500 if any internal server issue		
/api/v0/events/{eventID}/ guests/report	GET	**Input** None **Output (Successful)** Returns a CSV file (with header Content-Type text/csv). The first row has two cells, “Name” and “Present” Each row has a guest name, and then 1 or 0 in the Present column. 1 means present. **Errors** 404 if an event with that eventID does not exist. 500 if any internal server issue	Valid access token One of the following: User account which is a host of that event 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”, 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** {“nric” : “1234F”} **Output (Successful)** A PNG image (with header Content-Type image/png) of a QR code that encodes the NRIC supplied. **Errors** 400 if body could not be read as JSON 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.