

Documentation

API Documentation

Introduction

The RESTful API is hosted using Firebase Functions as a “serverless” solution. Currently the root endpoint where HTTP requests are sent is <https://us-central1-ecsas-4e69c.cloudfunctions.net/api>.

All the code for the ECSas is in one centralized repository <https://github.com/blakefuller/ECSas>, and code for the API is in the “functions” folder of the ECSas project.

Deploying Code Changes

To deploy changes made to the API, the FIrebase CLI command firebase deploy inside the functions folder (for information on how to install and use the Firebase CLI, see <https://firebase.google.com/docs/functions/get-started>).

Actions

Inside “functions”, index.js is the first file executed when the API is called. Actions here are separated into Announcement Routes, Archive Routes, and Admin Routes. Listed below are the HTTP verbs used with each route, the url path for the action, body for the request, and the response format.

Note that any protected route with “Authentication token” listed in the headers will require an authentication token in the header of the request, with the following key-value pair:

{ Authentication: Bearer `*token`* }

The /admin-login route is used to obtain a login token (see below).

Announcement Routes

**Get all announcements**

* Method: **GET**
* Route: /anncs
* Headers: Authentication token
* Body: None
* Response:

[

{

"id": "string",

"audience": "string",

"evnt\_title": "string",

"evnt\_date": "string",

"sub\_name": "string",

"sub\_email": "string",

"cont\_email": "string",

"description": "string",

"evnt\_loc": "string",

"url": "string",

"num\_weeks": "string",

"cont\_name": "string",

"category": "string",

"timestamp": "string (ISO format)"

},

{

"id": "string",

...

}

]

**Get announcement by id**

* Method: **GET**
* Route: /anncs/{id}
* Headers: Authentication token
* Body: None
* Response:

Success

{

"id": "string",

"url": "string",

"num\_weeks": "string",

"cont\_name": "string",

"category": "string",

"timestamp": "string",

"audience": "string",

"evnt\_title": "string",

"evnt\_date": "string",

"sub\_name": "string",

"sub\_email": "string",

"cont\_email": "string",

"description": "string",

"evnt\_loc": "string"

}

Failure

{ "error": “Document not found” }

**Submit announcement**

* Method: **POST**
* Route: /anncs
* Headers: None
* Body:

{

"url": "string",

"num\_weeks": "string",

"cont\_name": "string",

"category": "string",

"audience": "string",

"evnt\_title": "string",

"evnt\_date": "string",

"sub\_name": "string",

"sub\_email": "string",

"cont\_email": "string",

"description": "string",

"evnt\_loc": "string"

}

* Response:

Success

{ "message": "Document `id` created successfully" }

Failure

{ "message": `property` is not allowed to be empty" }

**Update announcement**

* Method: **PUT**
* Route: /anncs/{id}
* Headers: Authentication token
* Body:

{

"url": "string",

"num\_weeks": "string",

"cont\_name": "string",

"category": "string",

"audience": "string",

"evnt\_title": "string",

"evnt\_date": "string",

"sub\_name": "string",

"sub\_email": "string",

"cont\_email": "string",

"description": "string",

"evnt\_loc": "string"

}

* Response:

Success

{}

Failure

{ "error": “Document not found” }

Archive Routes

**Get all archived announcements**

* Method: **GET**
* Route: /archive
* Headers: Authentication token
* Body: None
* Response:

[

{

"id": "string",

"audience": "string",

"evnt\_title": "string",

"evnt\_date": "string",

"sub\_name": "string",

"sub\_email": "string",

"cont\_email": "string",

"description": "string",

"evnt\_loc": "string",

"url": "string",

"num\_weeks": "string",

"cont\_name": "string",

"category": "string",

"timestamp": "string (ISO format)"

},

{

"id": "string",

...

}

]

**Get archived announcement by id**

* Method: **GET**
* Route: /archive/{id}
* Headers: Authentication token
* Body: None
* Response:

Success

{

"id": "string",

"url": "string",

"num\_weeks": "string",

"cont\_name": "string",

"category": "string",

"timestamp": "string",

"audience": "string",

"evnt\_title": "string",

"evnt\_date": "string",

"sub\_name": "string",

"sub\_email": "string",

"cont\_email": "string",

"description": "string",

"evnt\_loc": "string"

}

Failure

{ "error": “Document not found” }

**Archive Announcement** - moves an announcement from the regular “announcement” collection to the “archive” collection

* Method: **PUT**
* Route: /archive/{id}
* Headers: Authentication token
* Body: None
* Response:

Success

{ "message": “Document `id` archived successfully” }

Failure

{ "message": “Document `id` not found” }

**Delete Archived Announcement**

* Method: **DELETE**
* Route: /archive/{id}
* Headers: Authentication token
* Body: None
* Response:

Success

{}

Failure

{ "message": “Document not found” }

Archive Routes

**Admin Login**

* Method: **POST**
* Route: /admin-login
* Headers: Authentication token
* Body:

{

“email”: “string”,

“password”: “string”

}

* Response:  
  Success

{ “token”: `token` }

Failure

{ "general": "Incorrect password" }

<or>

{ "error": "auth/user-not-found" }

Administrative Actions

Admin Sign Up / Change Email or Password

Currently, there is no signup functionality in the client app for ECSas; if a new user is to be created, the account needs to be created in the Authentication section of the ECSas Firebase console.

To access this, open <https://console.firebase.google.com/> with a Google account that has access to the ECSas Firebase project. Open the ECSas project, navigate to “Authentication” for the project, and perform the necessary actions.

**Important:** When a new admin account is created in the Authentication section of the Firebase project, a field also needs to be added to the database for that user in order for the app authentication to work. To do this, copy the “User UID” in the Authentication section, then navigate to the “Database” section of the Firebase project. In the collection admins, add a document with “Document ID” of the new user’s email, “name” field with the user’s full name as the value, and “userId” field with the copied “User UID” from the Authentication page.