Photo Tag API

Table of Contents

Data Model Section

My project models a “instagram” type of application, where users can add “photos” (uploaded elsewhere) then add tags to those photos. Tags do not need authorization, and can be added by anyone. The photos are associated with and owned by the user (in other words, they are a protected entity) and the user must provide authorization to when doing an operation involving them. When adding or deleted a tag from a photo, the user need only have access to the photo.

1. Entities:
    a. Tags
        i. name (string, required), valid values:

length	min: 2 character max: 24 characters
first character	The first character must be a hashtag character “#”
uniqueness	Only one tag with a given name may exist in the system at once. Name is case insensitive.
allowed characters	Only ASCII printable characters (except for DEL) are allowed. For a complete list, see Appendix A

        ii. description (string, required), valid values:

length	min: 1 character max: 256 characters
allowed characters	Only ASCII printable characters (except for DEL) are allowed. For a complete list, see Appendix A

        iii. type (string, required)

allowed	Only the following values will be accepted: “company”, “hashtag”, “location”. Input is case sensitive.

    b. Photos
        i. url (string, required), valid values:

length	min: 4 character max: 256 characters
uniqueness	Only one photo with a given url may exist for a user in the system at once. url is case insensitive.
allowed characters	Only the following are allowed (conforms to W3 Uniform Resource Identifier specification): 1. Alphanumeric: (a-z), (A-Z), (0-9) 2. Unreserved Characters: - _ . ~ 3. Reserved Characters: ! * ' ( ) ; : @ & = + $, / ? % # [ ]

ii. description (string, required)

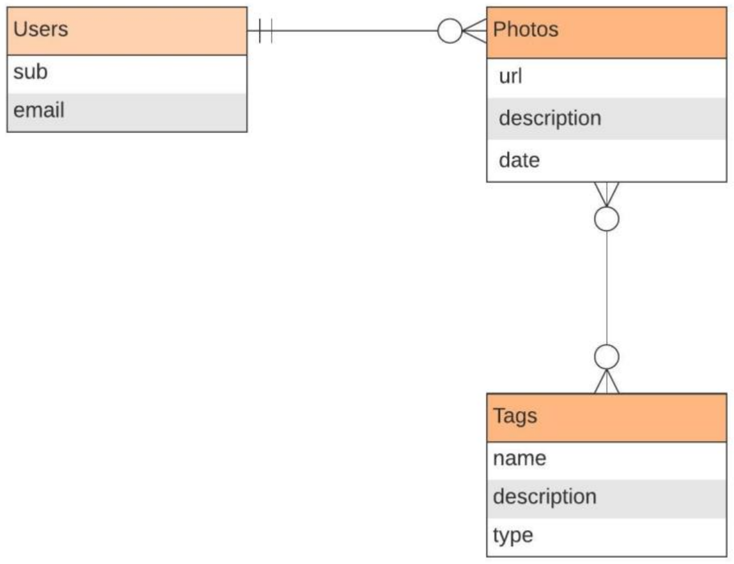
length	min: 1 character max: 256 characters
allowed characters	Only ASCII printable characters (except for DEL) are allowed. For a complete list, see Appendix A

iii. date (string, required)

range	Date must be a valid date of format YYYY-MM-DD (E.g. 2000-15-15 would be invalid)

c. Users
   i. sub (string, required) – autogenerated by Google
   ii. email (string, required) – auto received with Google OAuth login

2. Relationships
    a. ERD



    b. Users
        i. Users can have 0 or more photos
    c. Photos:
        i. Photos must have 1 user
        ii. Photos can have 0 or more tags
    d. Tags:
        i. Tags can belong to 0 or more photos
3. User Entity:
    a. Unique Identifier: “sub” is received in the JWT
    b. Protected Requests: The user’s credentials (JWT) must be supplied in the Authorization header, and are type “Bearer.” In other words, the Header should include {Authorization: Bearer [jwt]}
    c. Relationship: Users may add photos, and all photos must belong to one user. Thus, Users has a 1:M relationship with photos, and is implemented with photos having a owner property store in the database. This property is the user’s “sub”, received from the provided JWT.

Create a Tag

Allows you to create a new tag.

```
POST /tags
```

Request

Request Header Parameters

Name	Valid Options	Description	Required?
Content-Type	application/json	Content type of the request body	No
Accept	application/json	Content type of the created returned tag	Yes

Request Body

Required

Request Body Format

application/json

Request JSON Attributes

Name	Type	Description	Required?
name	String	The tag string. (e.g. `#HelloWorld`)	Yes
description	String	Short description of tag.	Yes
type	String	The type of the tag (hashtag, company, location)	Yes

Constraints for Request Attributes

Attribute	Type	Description
name	length	min: 2 character max: 24 characters
name	first character	The first character must be a hashtag character "#"
name	uniqueness	Only one tag with a given name may exist in the system at once. Name is case insensitive.
name	allowed characters	Only ASCII printable characters (except for DEL) are allowed. For a complete list, see Appendix A
description	length	min: 1 character max: 256 characters
description	allowed characters	Only ASCII printable characters (except for DEL) are allowed. For a complete list, see Appendix A
type	allowed	Only the following values will be accepted: "company", "hashtag", "location".

		Input is case sensitive.

Request Body Example

```
Headers:
    Content-Type: application/json
    Accept: application/json

Request Body:
{
"name": "#Chipotle",
"description": "American chain of fast casual restaurants specializing in
                tacos and Mission-style burritos",
"type": "company"
}
```

Response

Response Body Format

application/json

Response Statuses

Outcome	Status Code	Notes
Success	201 Created	
Failure	415 Unsupported Media Type	If the request content-type header is provided and is not of type 'application/json', the tag will not be created and a 415 status code will be returned
Failure	406 Not Acceptable	If the request accept header has specified a data type other than application/json, the tag must not be created and 406 statue code must be returned.
Failure	400 Bad Request	If the request is missing any of the required attributes, or if there is a problem with one of the tag attributes (e.g. the name is not a string), or if the request has extra attributes sent, the tag must not be created, and 400 status code must be returned.
Failure	409 Conflict	If the application has a tag with that name already, the tag will not be created and a 409 status code will be returned.

*Note: If a failure occurs, the first failure to occur in the list above will be the failure message received. E.g. If the request is missing a required attribute and the accept header has requested an unsupported MIME type, a 406 will be returned.

Success

```
Status: 201 Created

Headers:
   Content-Type: application/json

Response Body:
{
"id": "abc123",
"name": "#Chipotle",
"description": "American chain of fast casual restaurants specializing in
                 tacos and Mission-style burritos",
“photos”:[],
"type": "company",
"self": "[url]/tags/abc123"
}
```

Failure

```
Status: 400 Bad Request

Headers:
   Content-Type: application/json

Response Body:
{
"Error": "The request object is missing at least one of the required
          attributes, there is an issue with one of the required attributes,
          or additional attributes sent."
}
```

```
Status: 406 Not acceptable

Headers:
   Content-Type: application/json

Response Body:
{
"Error":  "Unsupported MIME type sent in request Accept Header or no Accept
Header sent."
}
```

```
Status: 409 Conflict

Headers:
   Content-Type: application/json
```

```
Response Body:
{
"Error":  "The name specified in the request object is already taken."
}
```

```
Status: 415 Unsupported Media Type

Headers:
   Content-Type: application/json

Response Body:
{
"Error":  "Only application/json acceptable."
}
```

List all Tags

Allows you to get all existing tag

```
GET /tags
```

Request

Request Parameters

Request Header Parameters

Name	Valid Options	Description	Required?
Accept	application/json	Content type of the returned tags	Yes

Request Body

None

*Note: If contents sent in the request body, or content-type header included, contents will be ignored.

Response

Response Body Format

application/json

Response Statuses

Outcome	Status Code	Notes
Success	200 OK	
Failure	406 Not Acceptable	If the request accept header has specified a data type other than application/json, the tag must not be created and 406 statue code must be returned.

Response Examples

Success

```
Status: 200 OK

Headers:
   Content-Type: application/json

Response Body:
[
“tags”:
[
      {
      "id": "abc123",
      "name": "#Chipotle",
      "description": "American chain of fast casual restaurants specializing
      in tacos and Mission-style burritos",
      "type": "company",
      “photos”:[],
      "self": "[url]/tags/abc123"
      },
      {
      "id": "def123",
      "name": "#McDonald’s",
      "description": "McDonald's Corporation is an American fast food
      company, founded in 1940 as a restaurant operated by Richard and
      Maurice McDonald, in San Bernardino, California, United States",
      "type": "company",
      “photos”:[
          {"id": "mno123",
           "self": "https://project-laprestm-
                    cs493.uc.r.appspot.com/photos/def123"
          }
      ],
      "self": "[url]/tags/def123"
      },
      {
      "id": "ghi123",
      "name": "#GoodTimes",
      "description": "A period of prosperity or happiness",
      "type": "hashtag",
      “photos”:[],
      "self": "[url]/tags/ghi123"
      },
      {
      "id": "jkl123",
      "name": "#chilling",
      "description": "a cool way of telling someone you sat around doing
      nothing",
      "type": "hashtag",
      “photos”:[],
      "self": "[url]/tags/jkl123"
      },
      {
      "id": "mno123",
      "name": "#Chicago",
      "description": "Chicago, officially the City of Chicago, is the most
      populous city in the U.S. state of Illinois, and the third-most-
      populous city in the United States. ",
      "type": "location",
      “photos”:[],
      "self": "[url]/tags/mno123"
      }
],
“next”: “[url]/tags?limit=5&offset=5”,
“count”: 7

]
```

Failure

```
Status: 406 Not acceptable

Headers:
   Content-Type: application/json
```

```
Response Body:
{
"Error":  "Unsupported MIME type sent in request Accept Header or no Accept
           Header sent."
}
```

OTHER /tags

```
[PUT/PATCH/DELETE] /tags
```

Response

Response Body Format

application/json

Response Status

Outcome	Status Code	Notes
Failure	405 Method Not Allowed	If a PUT, PATCH, or DELETE request is sent to /tags, request fails, and 405 status code is returned.

Response Examples

Failure

```
Status: 405 Method Not Allowed

Headers:
   Content-Type: application/json

Response Body:
{
"Error":  "Unsupported method at this URL.”
}
```

View a Tag

Allows you to get an existing tag

```
GET /tags/:tag_id
```

Request

Request Parameters

Name	Type	Description	Required?
tag_id	String	ID of the tag	Yes

Request Header Parameters

Name	Valid Options	Description	Required?
Accept	application/json	Content type of the returned tag	Yes

Request Body

None

*Note: If contents sent in the request body, or content-type header included, contents will be ignored.

Response

Response Body Format

application/json

Response Statuses

Outcome	Status Code	Notes
Success	200 OK	
Failure	406 Not Acceptable	If the request accept header has specified a data type other than application/json, request fails and a 406 status code must be returned.
Failure	404 Not Found	No tag with this tag_id exists

*Note: If a failure occurs, the first failure to occur in the list above will be the failure message received. E.g. If the tag_id does not exist and the request header is for an unsupported MIME type (such as application/pdf), a 406 not acceptable error will be returned.

Response Examples

Success

```
Status: 200 OK

Headers:
   Content-Type: application/json

Response Body:
{
```

```
"id": "abc123",
"name": "#Chipotle",
"description": "American chain of fast casual restaurants specializing in
                tacos and Mission-style burritos",
“photos”:[],
"type": "company",
"self": "[url]/tags/abc123"
}
```

Failure

```
Status: 404 Not Found

Headers:
   Content-Type: application/json

Response Body:
{
"Error":  "No tag with this tag_id exists"
}
```

```
Status: 406 Not Acceptable

Headers:
   Content-Type: application/json

Response Body:
{
"Error":  "Unsupported MIME type sent in request Accept Header or no Accept
Header sent."
}
```

Edit Partial Tag

Allows you to edit a tag without sending all editable parameters.

```
PATCH /tags/:tag_id
```

Request

Request Parameters

Name	Type	Description	Required?
tag_id	String	ID of the tag	Yes

Request Header Parameters

Name	Valid Options	Description	Required?
Content-Type	application/json	Content type of the request body	No
Accept	application/json	Content type of the created returned tag	Yes

Request Body

Required

Request Body Format

application/json

Request JSON Attributes

Name	Type	Description	Required?
name	String	The tag string. (e.g. `#HelloWorld`)	No
description	String	Short description of tag.	No
type	String	The type of the tag (hashtag, company, location)	No

* Note: At least one attribute (name, description, type) must be sent

Constraints for Request Attributes

Attribute	Type	Description
name	length	min: 2 character max: 24 characters
name	first character	The first character must be a hashtag character “#”
name	uniqueness	Only one tag with a given name may exist in the system at once. Name is case insensitive.
name	allowed characters	Only ASCII printable characters (except for DEL) are allowed. For a complete list, see Appendix A
description	length	min: 1 character

		max: 256 characters
description	allowed characters	Only ASCII printable characters (except for DEL) are allowed. For a complete list, see Appendix A
type	allowed	Only the following values will be accepted: “company”, “hashtag”, “location”

Request Example

```
Headers:
    Content-Type: application/json
    Accept: application/json

Request Body:
{
"description": "American chain of fast casual restaurants specializing in
                tacos and Mission-style burritos, founded by Steve Ells"
}
```

Response

Response Body Format

application/json

Response Statuses

Outcome	Status Code	Notes
Success	200 OK	
Failure	415 Unsupported Media Type	If the request content-type header is provided and is not of type ‘application/json’, the tag will not be updated and a 415 status code will be returned
Failure	406 Not Acceptable	If the request accept header has specified a data type other than application/json, the tag must not be updated and 406 statue code must be returned.
Failure	400 Bad Request	If the request has an extra attribute or no attributes, or one of the attributes has failed a constraint, the tag must not be updated, and 400 status code must be returned.

Failure	409 Conflict	If the application has a **different** tag with that name already, the tag will not be updated and a 409 status code will be returned. (Note: if tag_id abc123 is being updated, and the same tag name is sent, the tag will be successfully updated).
Failure	404 Not Found	No tag with this tag_id exists

*Note: If a failure occurs, the first failure to occur in the list above will be the failure message received. E.g. If the request has an extra invalid attribute and the accept header has requested an unsupported MIME type, a 406 will be returned.

Response Examples

Success

```
Status: 200 OK

Headers:
   Content-Type: application/json

Response Body:
{
"id": "abc123",
"name": "#Chipotle",
"description": "American chain of fast casual restaurants specializing in
                tacos and Mission-style burritos, founded by Steve Ells",
“photos”:[],
"type": "company",
"self": "[url]/tags/abc123"
}
```

Failure

```
Status: 400 Bad Request

Headers:
   Content-Type: application/json

Response Body:
{
"Error": "There is an issue with one of the attributes, no attributes sent,
or additional attributes sent."
}
```

```
Status: 404 Not Found

Headers:
   Content-Type: application/json

Response Body:
{
"Error":  "No tag with this tag_id exists"
}
```

```
Status: 406 Not acceptable

Headers:
   Content-Type: application/json

Response Body:
{
"Error":  "Unsupported MIME type sent in request Accept Header or no Accept
Header sent."
}
```

```
 Status: 409 Conflict

Headers:
   Content-Type: application/json

Response Body:
{
"Error":  "The name specified in the request object is already taken."
}
```

```
Status: 415 Unsupported Media Type

Headers:
   Content-Type: application/json

Response Body:
{
“Error”:  “Only application/json acceptable.”
}
```

Edit entire Tag

Allows you to edit a tag.

```
PUT /tags/:tag_id
```

Request

Request Parameters

Name	Type	Description	Required?
tag_id	String	ID of the tag	Yes

Request Header Parameters

Name	Valid Options	Description	Required?
Content-Type	application/json	Content type of the request body	No
Accept	application/json	Content type of the created returned tag	Yes

Request Body

Required

Request Body Format

application/json

Request JSON Attributes

Name	Type	Description	Required?
name	String	The tag string. (e.g. `#HelloWorld`)	Yes
description	String	Short description of tag.	Yes
type	String	The type of the tag (hashtag, company, location)	Yes

Constraints for Request Attributes

Attribute	Type	Description
name	length	min: 2 character max: 24 characters
name	first character	The first character must be a hashtag character "#"
name	uniqueness	Only one tag with a given name may exist in the system at once. Name is case insensitive.
name	allowed characters	Only ASCII printable characters (except for DEL) are allowed. For a complete list, see Appendix A
description	length	min: 1 character max: 256 characters

description	allowed characters	Only ASCII printable characters (except for DEL) are allowed. For a complete list, see Appendix A
type	allowed	Only the following values will be accepted: “company”, “hashtag”, “location”

Request Example

```
Headers:
    Content-Type: application/json
    Accept: application/json

Request Body:
{
"name": "#Chipotle",
"description": "American chain of fast casual restaurants specializing in
                tacos and Mission-style burritos, founded by Steve Ells",
"type": "company"
}
```

Response

Response Body Format

application/json

Response Statuses

Outcome	**Status Code**	**Notes**
Success	200 OK	
Failure	415 Unsupported Media Type	If the request content-type header is provided and is not of type ‘application/json’, the tag will not be updated and a 415 status code will be returned
Failure	406 Not Acceptable	If the request accept header has specified a data type other than application/json, the tag must not be updated and 406 statue code must be returned.
Failure	400 Bad Request	If the request has an extra attribute or is missing a required attribute, or one of the attributes has failed a constraint, the tag must not be updated, and 400 status code must be returned.
Failure	409 Conflict	If the application has a **different** tag with that name already, the boat will not be updated and a 409 status code will be returned. (Note: if tag_id abc123 is being updated, and the same tag name is sent, the tag will be successfully updated).

Failure	404 Not Found	No tag with this tag_id exists

*Note: If a failure occurs, the first failure to occur in the list above will be the failure message received. E.g. If the request has an extra invalid attribute and the accept header has requested an unsupported MIME type, a 406 will be returned.

Response Examples

Success

```
Status: 200 OK

Headers:
   Content-Type: application/json
   location : https://hw5-laprestm-cs493.uc.r.appspot.com/boats/abc123

Response Body:
{
"id": "abc123",
"name": "#Chipotle",
"description": "American chain of fast casual restaurants specializing in
                tacos and Mission-style burritos, founded by Steve Ells",
“photos”:[],
"type": "company",
"self": [url]/tags/abc123"
}
```

Failure

```
Status: 400 Bad Request

Headers:
   Content-Type: application/json

Response Body:
{
"Error": "The request object is missing at least one of the required
          attributes, there is an issue with one of the required attributes,
          or additional attributes sent."
}
```

```
Status: 404 Not Found

Headers:
   Content-Type: application/json

Response Body:
{
"Error": "No tag with this tag_id exists"
}
```

```
Status: 406 Not acceptable
```

```
Headers:
   Content-Type: application/json

Response Body:
{
"Error": "Unsupported MIME type sent in request Accept Header or no Accept
Header sent."
}
```

```
Status: 409 Conflict

Headers:
   Content-Type: application/json

Response Body:
{
"Error": "The name specified in the request object is already taken."
}
```

```
Status: 415 Unsupported Media Type

Headers:
   Content-Type: application/json

Response Body:
{
“Error”: “Only application/json acceptable.”
}
```

Delete a Tag

Allows you to delete a tag. Note that deleting a tag will remove it from any photos it was associated.

```
DELETE /tags/:tag_id
```

Request

Request Parameters

Name	Type	Description	Required?
tag_id	String	ID of the tag	Yes

Request Body

None

*Note: If contents sent in the request body, or content-type header included, contents will be ignored.

Response

No body

Response Body Format

Success: No body

Failure: JSON

Response Statuses

Outcome	Status Code	Notes
Success	204 No Content	
Failure	404 Not Found	No tag with this tag_id exists

Response Examples

Success

```
Status: 204 No Content
```

Failure

```
Status: 404 Not Found

Headers:
   Content-Type: application/json

Response Body:
{
"Error":  "No tag with this tag_id exists"
}
```

Create a Photo [Protected]

Allows user to create a new photo.

```
POST /photos
```

Request

Request Parameters

Request Header Parameters

Name	Valid Options	Description	Required?
Content-Type	application/json	Content type of the request body	No
Accept	application/json	Content type of the returned photo	Yes
Authorization	Bearer [jwt]	Valid JSON Authorization Web Token	Yes

Request Body

Required

Request Body Format

application/json

Request JSON Attributes

Name	Type	Description	Required?
url	String	URL of the photo	Yes
description	String	Short description of photo	Yes
date	String	Date photo was taken (YYYY-MM-DD)	Yes

Constraints for Request Attributes

Attribute	Type	Description
url	length	min: 4 character max: 256 characters
url	uniqueness	Only one photo with a given url may exist for a user in the system at once. url is case insensitive.
url	allowed characters	Only the following are allowed (conforms to W3 Uniform Resource Identifier specification): 1. Alphanumeric: (a-z), (A-Z), (0-9) 2. Unreserved Characters: - _ . ~ 3. Reserved Characters: ! * ' ( ) ; : @ & = + $, / ? % # [ ]
description	length	min: 1 character max: 256 characters
description	allowed	Only ASCII printable characters (except for DEL) are

	characters	allowed. For a complete list, see Appendix A
date	range	Date must be a valid date, (E.g. 2000-15-15 would be invalid)

Request Body Example

```
Headers:
    Content-Type: application/json
    Accept: application/json
    Authorization: Bearer sfo1o3ew08u123ekjqweacjcsj

Request Body:
{
"url": " https://i.redd.it/45cbn2dm6kj41.jpg",
"description": "Bulldog in bathtub",
"date": "2020-03-04"
}
```

Response

Response Body Format

application/json

Response Statuses

Outcome	Status Code	Notes
Success	201 Created	
Failure	401 Unauthorized	If the client has failed to provid credentials for authorization (through the authorization header) or has provided bad credentials, photo will not be created and 401 will be returned.
Failure	415 Unsupported Media Type	If the request content-type header is provided and is not of type 'application/json', the photo will not be created and a 415 status code will be returned
Failure	406 Not Acceptable	If the request accept header has specified a data type other than application/json, the photo must not be created and 406 statue code must be returned.
Failure	400 Bad Request	If the request is missing any of the required attributes, or if there is a problem with one of the photo attributes (e.g. the name is not a string), or if the request has extra attributes sent, the photo must not be created, and 400 status code must be returned.

Failure	409 Conflict	If the user has a photo with that url already, the photo will not be created and a 409 status code will be returned.

*Note: If a failure occurs, the first failure to occur in the list above will be the failure message received. E.g. If the request is missing a required attribute and the accept header has requested an unsupported MIME type, a 406 will be returned.

Success

```
Status: 201 Created

Headers:
   Content-Type: application/json

Response Body:
{
"id": "123abc",
"url": " https://i.redd.it/45cbn2dm6kj41.jpg",
"description": "Bulldog in bathtub",
"date": "2020-03-04",
"tags":[],
"owner": "sub12380234081",
"self": "[url]/photos/123abc"
}
```

Failure

```
Status: 400 Bad Request

Headers:
   Content-Type: application/json

Response Body:
{
"Error": "The request object is missing at least one of the required
          attributes, there is an issue with one of the required attributes,
          or additional attributes sent."
}
```

```
Status: 401 Unauthorized

Headers:
   Content-Type: application/json

Response Body:
{
"Error":  "No credentials provided or provided credentials invalid."
}
```

```
Status: 406 Not acceptable
```

```
Headers:
   Content-Type: application/json

Response Body:
{
"Error":  "Unsupported MIME type sent in request Accept Header or no Accept
Header sent."
}
```

```
Status: 409 Conflict

Headers:
   Content-Type: application/json

Response Body:
{
"Error":  "The url specified in the request object is already taken."
}
```

```
Status: 415 Unsupported Media Type

Headers:
   Content-Type: application/json

Response Body:
{
"Error":  "Only application/json acceptable."
}
```

List all Photos [Protected]

Allows user to get all existing photos associated with the authorized user

```
GET /photos
```

Request

Request Header Parameters

Name	Valid Options	Description	Required?
Accept	application/json, text/html	Content type of the returned tags	Yes
Authorization	Bearer [jwt]	Valid JSON Authorization Web Token	Yes

Request Parameters

None

Request Body

None

*Note: If contents sent in the request body, or content-type header included, contents will be ignored.

Response

Response Body Format

application/json

Response Statuses

Outcome	Status Code	Notes
Success	200 OK	
Failure	401 Unauthorized	If the client has failed to provide credentials for authorization (through the authorization header) or has provided bad credentials, request fails and 401 will be returned.
Failure	406 Not Acceptable	If the request accept header has specified a data type other than application/json, the photo must not be created and 406 statue code must be returned.

*Note: If a failure occurs, the first failure to occur in the list above will be the failure message received. E.g. If the request is missing authorization, and the accept header has requested an unsupported MIME type, a 401 will be returned.

Response Examples

Success

```
Status: 200 OK

Headers:
   Content-Type: application/json

Response Body:
[
“photos”:
[
      {
      "id": "123abc",
      "url": “https://i.redd.it/45cbn2dm6kj41.jpg",
      "description": "Bulldog in bathtub ",
      "date": "2020-03-04",
      "tags":[],
      "owner": "sub12380234081",
      "self": "[url]/photos/123abc"
      },
      {
      "id": "def123",
      "url": “https://i.redd.it/45cbn2dmsdfj41.jpg",
      "description": "My first time at Chipotle",
      "date": "2010-03-04",
      "tags":
      [
            {
            "id": "ghi123",
            "name": "#GoodTimes",
            "self": "https://project-laprestm-
                     cs493.uc.r.appspot.com/tags/ghi123"
            },{
            "id": "abc123",
            "name": "#Chipotle",
            "self": "https://project-laprestm-
                     cs493.uc.r.appspot.com/tags/abc123"
            }
      ],
      "owner": "sub12380234081",
      "self": "[url]/photos/def123"
      },
      {
      "id": "ghi123",
      "url": “https://i.redd.it/45cbn2dm6kj42.jpg",
      "description": "My trip to Paris",
      "date": "2014-12-04",
      "tags":[],
      "owner": "sub12380234081",
      "self": "[url]/photos/ghi123"
      },
      {
      "id": "jkl123",
      "url": “https://i.redd.it/45cb12fasdfj41.jpg",
      "description": "Skiing",
      "date": "2009-02-04",
      "tags":[],
      "owner": "sub12380234081",
      "self": "[url]/photos/def123"
      },
      {
      "id": "mno123",
      "url": “https://i.redd.it/45cbn2123dmsdfj41.jpg",
      "description": "Cheeseburger",
      "date": "2020-05-04",
      "tags":
```

```
      [
            {
            "id": "def123",
            "name": "#McDonald’s",
            "self": "https://project-laprestm-
                     cs493.uc.r.appspot.com/tags/def123"
            }
      ],
      "owner": "sub12380234081",
      "self": "[url]/photos/def123"
      }
],
“next”: “[url]/photos?limit=5&offset=5”,
“count”: 11
]
```

Failure

```
Status: 401 Unauthorized

Headers:
   Content-Type: application/json

Response Body:
{
"Error":  "No credentials provided or provided credentials invalid."
}
```

```
Status: 406 Not acceptable

Headers:
   Content-Type: application/json

Response Body:
{
"Error":  "Unsupported MIME type sent in request Accept Header or no Accept
Header sent."
}
```

OTHER /photos

```
[PUT/PATCH/DELETE] /photos
```

Response

Response Body Format

application/json

Response Status

Outcome	Status Code	Notes
Failure	405 Method Not Allowed	If a PUT, PATCH, or DELETE request is sent to /photos, request fails, and 405 status code is returned.

Response Examples

Failure

```
Status: 405 Method Not Allowed

Headers:
   Content-Type: application/json

Response Body:
{
"Error":  "Method Not Allowed”
}
```

View a Photo [Protected]

Allows user to get a user's existing photo

```
GET /photos/:photo_id
```

Request

Request Header Parameters

Name	Valid Options	Description	Required?
Accept	application/json	Content type of the returned photo	Yes
Authorization	Bearer [jwt]	Valid JSON Authorization Web Token	Yes

Request Parameters

Name	Type	Description	Required?
photo_id	String	ID of the photo	Yes

Request Body

None

*Note: If contents sent in the request body, or content-type header included, contents will be ignored.

Response

Response Body Format

application/json

Response Statuses

Outcome	Status Code	Notes
Success	200 OK	
Failure	401 Unauthorized	If the client has failed to provide credentials for authorization (through the authorization header) or has provided bad credentials, request fails and 401 will be returned.
Failure	406 Not Acceptable	If the request accept header has specified a data type other than application/json, the photo must not be created and 406 statue code must be returned.
Failure	404 Not Found	No photo with this photo_id exists, requests fails and 404 is returned
Failure	403 Forbidden	The provided credentials are valid, but the client does not have access to this photo. Request fails, and 403 is returned

*Note: If a failure occurs, the first failure to occur in the list above will be the failure message received. E.g. If the photo_id does not exist and the request header is for an unsupported MIME type (such as application/pdf), a 406 not acceptable error will be returned.

Response Examples

Success

```
Status: 200 OK

Headers:
   Content-Type: application/json

Response Body:
{
"id": "def123",
"url": “https://i.redd.it/45cbn2dmsdfj41.jpg",
"description": "My first time at Chipotle",
"date": "2010-03-04",
"tags":
 [
     {
      "id": "ghi123",
      "name": "#GoodTimes",
      "self": "[url]/tags/ghi123"
     },
     {
      "id": "abc123",
      "name": "#Chipotle",
      "self": "[url]/tags/abc123"
     }
],
"owner": "sub12380234081",
"self": "[url]/photos/def123"
}
```

Failure

```
Status: 401 Unauthorized

Headers:
   Content-Type: application/json

Response Body:
{
"Error":  "No credentials provided or provided credentials invalid."
}
```

```
Status: 403 Forbidden

Headers:
   Content-Type: application/json

Response Body:
{
"Error":  "Credentials provided do not have access to this photo”
}
```

```
Status: 404 Not Found
```

```
Headers:
   Content-Type: application/json

Response Body:
{
"Error":  "No photo with this photo_id exists"
}
```

```
Status: 406 Not Acceptable

Headers:
   Content-Type: application/json

Response Body:
{
"Error":  "Unsupported MIME type sent in request Accept Header or no Accept
Header sent."
}
```

Edit Partial Photo [Protected]

Allows a user to edit their photo without sending all editable parameters.

```
PATCH /photos/:photo_id
```

Request

Request Parameters

Name	Type	Description	Required?
photo_id	String	ID of the photo	Yes

Request Header Parameters

Name	Valid Options	Description	Required?
Content-Type	application/json	Content type of the request body	No
Accept	application/json	Content type of the created returned tag	Yes
Authorization	Bearer [jwt]	Valid JSON Authorization Web Token	Yes

Request Body

Required

Request Body Format

application/json

Request JSON Attributes

Request JSON Attributes

Name	Type	Description	Required?
url	String	URL of the photo	No
description	String	Short description of photo	No
date	String	Date photo was taken (YYYY-MM-DD)	No

*Note: At least one attribute (url, description, date) must be sent.

Constraints for Request Attributes

Attribute	Type	Description
url	length	min: 4 character max: 256 characters
url	uniqueness	Only one photo with a given url may exist for a user in the system at once. url is case insensitive.
url	allowed characters	Only the following are allowed (conforms to W3 Uniform Resource Identifier specification): 1. Alphanumeric: (a-z), (A-Z), (0-9) 2. Unreserved Characters: - _ . ~ 3. Reserved Characters: ! * ' ( ) ; : @ & = + $, / ? % # [ ]

description	length	min: 1 character max: 256 characters
description	allowed characters	Only ASCII printable characters (except for DEL) are allowed. For a complete list, see Appendix A
date	range	Date must be a valid date, (E.g. 2000-15-15 would be invalid)

Request Example

```
Headers:
    Content-Type: application/json
    Accept: application/json
    Authorization: Bearer sfo1o3ew08u123ekjqweacjcsj

Request Body:
{
"description": "Bulldog (Bonnie) in bathtub"
}
```

Response

Response Body Format

application/json

Response Statuses

Outcome	**Status Code**	**Notes**
Success	200 OK	
Failure	401 Unauthorized	If the client has failed to provide credentials for authorization (through the authorization header) or has provided bad credentials, request fails and 401 will be returned.
Failure	415 Unsupported Media Type	If the request content-type header is provided and is not of type 'application/json', the photo will not be updated and a 415 status code will be returned

Failure	406 Not Acceptable	If the request accept header has specified a data type other than application/json, the photo must not be created and 406 statue code must be returned.
Failure	400 Bad Request	If the request has an extra invalid attribute or no attributes, or one of the attributes has failed a constraint, the photo must not be updated, and 400 status code must be returned.
Failure	409 Conflict	If the user has a photo with that url already, the photo will not be created and a 409 status code will be returned. (Note: if photo_id 123abc is being updated, and the same url is sent, the photo will be successfully updated).
Failure	404 Not Found	No photo with this photo_id exists
Failure	403 Forbidden	The provided credentials are valid, but the client does not have access to this photo. Request fails, and 403 is returned

*Note: If a failure occurs, the first failure to occur in the list above will be the failure message received. E.g. If the request has an invalid attribute and the accept header has requested an unsupported MIME type, a 406 will be returned.

Response Examples

Success

```
Status: 200 OK

Headers:
   Content-Type: application/json

Response Body:
{
"id": "123abc",
"url": ‶https://i.redd.it/45cbn2dm6kj41.jpg",
"description": "Bulldog (Bonnie) in bathtub ",
"date": "2020-03-04",
"tags":[],
"owner": "sub12380234081",
"self": "[url]/photos/123abc"
}
```

Failure

```
Status: 400 Bad Request

Headers:
   Content-Type: application/json

Response Body:
{
"Error": "There is an issue with one of the attributes, no attributes sent, or additional attributes sent."
}
```

```
Status: 401 Unauthorized

Headers:
   Content-Type: application/json

Response Body:
{
"Error":  "No credentials provided or provided credentials invalid."
}
```

```
Status: 403 Forbidden

Headers:
   Content-Type: application/json

Response Body:
{
"Error":  "Credentials provided do not have access to this photo”
}
```

```
Status: 404 Not Found

Headers:
   Content-Type: application/json

Response Body:
{
"Error":  "No photo with this photo_id exists"
}
```

```
Status: 406 Not acceptable

Headers:
   Content-Type: application/json

Response Body:
{
"Error":  "Unsupported MIME type sent in request Accept Header or no Accept Header sent."
}
```

```
Status: 409 Conflict

Headers:
   Content-Type: application/json

Response Body:
{
"Error":  "The url specified in the request object is already taken."
}
```

```
Status: 415 Unsupported Media Type

Headers:
   Content-Type: application/json

Response Body:
{
“Error”: “Only application/json acceptable.”
}
```

Edit entire Photo [Protected]

Allows a user to edit their photo.

```
PUT /photos/:photo_id
```

Request

Request Parameters

Name	Type	Description	Required?
photo_id	String	ID of the photo	Yes

Request Header Parameters

Name	Valid Options	Description	Required?
Content-Type	application/json	Content type of the request body	No
Accept	application/json	Content type of the created returned tag	Yes
Authorization	Bearer [jwt]	Valid JSON Authorization Web Token	Yes

Request Body

Required

Request Body Format

application/json

Request JSON Attributes

Request JSON Attributes

Name	Type	Description	Required?
url	String	URL of the photo	Yes
description	String	Short description of photo	Yes
date	String	Date photo was taken (YYYY-MM-DD)	Yes

Constraints for Request Attributes

Attribute	Type	Description
url	length	min: 4 character max: 256 characters
url	uniqueness	Only one photo with a given url may exist for a user in the system at once. url is case insensitive.
url	allowed characters	Only the following are allowed (conforms to W3 Uniform Resource Identifier specification): 1. Alphanumeric: (a-z), (A-Z), (0-9) 2. Unreserved Characters: - _ . ~ 3. Reserved Characters: ! * ' ( ) ; : @ & = + $, / ? % # [ ]
description	length	min: 1 character

		max: 256 characters
description	allowed characters	Only ASCII printable characters (except for DEL) are allowed. For a complete list, see Appendix A
date	range	Date must be a valid date, (E.g. 2000-15-15 would be invalid)

Request Example

```
Headers:
    Content-Type: application/json
    Accept: application/json
    Authorization: Bearer sfo1o3ew08u123ekjqweacjcsj

Request Body:
{
"url": " https://i.redd.it/45cbn2dm6kj41.jpg",
"description": "Bulldog (Bonnie) in bathtub",
"date": "2020-03-04"
}
```

Response

Response Body Format

application/json

Response Statuses

Outcome	**Status Code**	**Notes**
Success	200 OK	
Failure	401 Unauthorized	If the client has failed to provide credentials for authorization (through the authorization header) or has provided bad credentials, request fails and 401 will be returned.
Failure	415 Unsupported Media Type	If the request content-type header is provided and is not of type ‘application/json’, the photo will not be updated and a 415 status code will be returned

Failure	406 Not Acceptable	If the request accept header has specified a data type other than application/json, the photo must not be created and 406 statue code must be returned.
Failure	400 Bad Request	If the request has an extra invalid attribute or is missing attributes, or one of the attributes has failed a constraint, the photo must not be updated, and 400 status code must be returned.
Failure	409 Conflict	If the user has a photo with that url already, the photo will not be created and a 409 status code will be returned. (Note: if photo_id 123abc is being updated, and the same url is sent, the photo will be successfully updated).
Failure	404 Not Found	No photo with this photo_id exists
Failure	403 Forbidden	The provided credentials are valid, but the client does not have access to this photo. Request fails, and 403 is returned

*Note: If a failure occurs, the first failure to occur in the list above will be the failure message received. E.g. If the request has an invalid attribute and the accept header has requested an unsupported MIME type, a 406 will be returned.

Response Examples

Success

```
Status: 200 OK

Headers:
   Content-Type: application/json

Response Body:
{
"id": "123abc",
"url": “https://i.redd.it/45cbn2dm6kj41.jpg",
"description": "Bulldog (Bonnie) in bathtub ",
"date": "2020-03-04",
"tags":[],
"owner": "sub12380234081",
"self": "[url]/photos/123abc"
}
```

Failure

```
Status: 400 Bad Request

Headers:
   Content-Type: application/json

Response Body:
{
"Error": "The request object is missing at least one of the required
attributes, there is an issue with one of the required attributes, or
additional attributes sent."
}
```

```
Status: 401 Unauthorized

Headers:
   Content-Type: application/json

Response Body:
{
"Error":  "No credentials provided or provided credentials invalid."
}
```

```
Status: 403 Forbidden

Headers:
   Content-Type: application/json

Response Body:
{
"Error":  "Credentials provided do not have access to this photo”
}
```

```
Status: 404 Not Found

Headers:
   Content-Type: application/json

Response Body:
{
"Error":  "No photo with this photo_id exists"
}
```

```
Status: 406 Not acceptable

Headers:
   Content-Type: application/json

Response Body:
{
"Error":  "Unsupported MIME type sent in request Accept Header or no Accept
Header sent."
}
```

```
Status: 409 Conflict

Headers:
   Content-Type: application/json

Response Body:
{
"Error":  "The url specified in the request object is already taken."
}
```

```
Status: 415 Unsupported Media Type

Headers:
   Content-Type: application/json

Response Body:
{
“Error”: “Only application/json acceptable.”
}
```

Delete a Photo [Protected]

Allows a user to delete one of their photos. Note that deleting a photo will remove it from any tags it was associated with.

```
DELETE /photos/:photo_id
```

Request

Request Parameters

Name	Type	Description	Required?
photo_id	String	ID of the photo	Yes

Request Header Parameters

Name	Valid Options	Description	Required?
Authorization	Bearer [jwt]	Valid JSON Authorization Web Token	Yes

Request Body

None

*Note: If contents sent in the request body, or content-type header included, contents will be ignored.

Response

No body

Response Body Format

Success: No body

Failure: JSON

Response Statuses

Outcome	Status Code	Notes
Success	204 No Content	Success only if a photo exists with this photo_id, and user has provided valid authorization to access this photo.
Failure	401 Unauthorized	If the client has failed to provide credentials for authorization (through the authorization header) or has provided bad credentials, request fails and 401 will be returned.
Failure	404 Not Found	No photo with this photo_id exists
Failure	403 Forbidden	The provided credentials are valid, but the client does not have access to this photo. Request fails, and 403 is returned

Response Examples

Success

```
Status: 204 No Content
```

Failure

```
Status: 401 Unauthorized

Headers:
   Content-Type: application/json

Response Body:
{
"Error":  "No credentials provided or provided credentials invalid."
}
```

```
Status: 403 Forbidden

Headers:
   Content-Type: application/json

Response Body:
{
"Error":  "Credentials provided do not have access to this photo”
}
```

```
Status: 404 Not Found

Headers:
   Content-Type: application/json

Response Body:
{
"Error":  "No photo with this photo_id exists"
}
```

Add tag to a Photo [Protected]

Allows user to add a tag to one of their photos

```
PUT /photos/:photo_id/tags/:tag_id
```

Request

Request Parameters

Name	Type	Description	Required?
photo_id	String	ID of the photo	Yes
tag_id	String	ID of the tag	Yes

Request Header Parameters

Name	Valid Options	Description	Required?
Authorization	Bearer [jwt]	Valid JSON Authorization Web Token	Yes

Request Body

None

*Note: If contents sent in the request body, or content-type header included, contents will be ignored.

Response

No body

Response Body Format

Success: No body

Failure: JSON

Response Statuses

Outcome	Status Code	Notes
Success	204 No Content	Success only if a photo exists with this photo_id, a tag exists with this tag_id, and user has provided valid authorization to access this photo.
Failure	401 Unauthorized	If the client has failed to provide credentials for authorization (through the authorization header) or has provided bad credentials, request fails and 401 will be returned.
Failure	404 Not Found	No photo with this photo_id exists and/or no tag with this tag_id exists
Failure	403 Forbidden	The provided credentials are valid, but the client does not have access to this photo. Request fails, and 403 is returned
Failure	400 Bad Request	If the photo already has that tag associated with it, request fails (in other words, no duplicate tags) and 400 is returned

Response Examples

Success

```
Status: 204 No Content
```

Failure

```
 Status: 400 Bad Request

Headers:
   Content-Type: application/json

Response Body:
{
"Error": "The photo is already tagged with that tag_id."
}
```

```
Status: 401 Unauthorized

Headers:
   Content-Type: application/json

Response Body:
{
"Error":  "No credentials provided or provided credentials invalid."
}
```

```
Status: 403 Forbidden

Headers:
   Content-Type: application/json

Response Body:
{
"Error":  "Credentials provided do not have access to this photo"
}
```

```
Status: 404 Not Found

Headers:
   Content-Type: application/json

Response Body:
{
"Error": "No photo with this photo_id exists and/or no tag with this tag_id
          exists"
}
```

Delete tag from a Photo [Protected]

Allows user to remove a tag from one of their photos

```
DELETE /photos/:photo_id/tags/:tag_id
```

Request

Request Parameters

Name	Type	Description	Required?
photo_id	String	ID of the photo	Yes
tag_id	String	ID of the tag	Yes

Request Header Parameters

Name	Valid Options	Description	Required?
Authorization	Bearer [jwt]	Valid JSON Authorization Web Token	Yes

Request Body

None

*Note: If contents sent in the request body, or content-type header included, contents will be ignored.

Response

No body

Response Body Format

Success: No body

Failure: JSON

Response Statuses

Outcome	Status Code	Notes
Success	204 No Content	Success only if a photo exists with this photo_id, a tag exists with this tag_id, and user has provided valid authorization to access this photo.
Failure	401 Unauthorized	If the client has failed to provide credentials for authorization (through the authorization header) or has provided bad credentials, request fails and 401 will be returned.
Failure	404 Not Found	No photo with this photo_id exists and/or no tag with this tag_id exists, or this photo was not tagged with the given tag_id
Failure	403 Forbidden	The provided credentials are valid, but the client does not have access to this photo. Request fails, and 403 is returned

Response Examples

Success

```
Status: 204 No Content
```

Failure

```
Status: 401 Unauthorized

Headers:
   Content-Type: application/json

Response Body:
{
"Error":  "No credentials provided or provided credentials invalid."
}
```

```
Status: 403 Forbidden

Headers:
   Content-Type: application/json

Response Body:
{
"Error":  "Credentials provided do not have access to this photo”
}
```

```
Status: 404 Not Found

Headers:
   Content-Type: application/json

Response Body:
{
"Error": "No photo with this photo_id exists and/or no tag with this tag_id
          exists, or this photo was not tagged by this tag_id"
}
```

List all Users

Allows you to get all authenticated users

```
GET /users
```

Request

Request Parameters

Request Header Parameters

Name	Valid Options	Description	Required?
Accept	application/json	Content type of the returned users	Yes

Request Body

None

*Note: If contents sent in the request body, or content-type header included, contents will be ignored.

Response

Response Body Format

application/json

Response Statuses

Outcome	Status Code	Notes
Success	200 OK	
Failure	406 Not Acceptable	If the request accept header has specified a data type other than application/json, the tag must not be created and 406 statue code must be returned.

Response Examples

Success

```
Status: 200 OK

Headers:
   Content-Type: application/json

Response Body:
["users":[
{
"sub": "sub1283972374",
"email": "user1@gmail.com"
},
{
"sub": "sub12839734274",
"email": "user2@gmail.com"
},
{
"sub": "sub12839114274",
"email": "user3@gmail.com"
}],
"Count":3
]
```

Failure

```
Status: 406 Not acceptable

Headers:
   Content-Type: application/json

Response Body:
{
"Error":  "Unsupported MIME type sent in request Accept Header or no Accept
Header sent."
}
```

APPENDIX A - TABLE OF ASCII PRINTABLE CHARACTERS

DEC	HEX	Symbol	Description
32	20		Space
33	21	!	Exclamation mark
34	22	"	Double quotes (or speech marks)
35	23	#	Number
36	24	$	Dollar
37	25	%	Per cent sign
38	26	&	Ampersand
39	27	'	Single quote
40	28	(	Open parenthesis (or open bracket)
41	29	)	Close parenthesis (or close bracket)
42	2A	*	Asterisk
43	2B	+	Plus
44	2C	,	Comma
45	2D	-	Hyphen
46	2E	.	Period, dot or full stop

47	2F	/	Slash or divide
48	30	0	Zero
49	31	1	One
50	32	2	Two
51	33	3	Three
52	34	4	Four
53	35	5	Five
54	36	6	Six
55	37	7	Seven
56	38	8	Eight
57	39	9	Nine
58	3A	:	Colon
59	3B	;	Semicolon
60	3C	<	Less than (or open angled bracket)
61	3D	=	Equals
62	3E	>	Greater than (or close angled bracket)
63	3F	?	Question mark

64	40	@	At symbol
65	41	A	Uppercase A
66	42	B	Uppercase B
67	43	C	Uppercase C
68	44	D	Uppercase D
69	45	E	Uppercase E
70	46	F	Uppercase F
71	47	G	Uppercase G
72	48	H	Uppercase H
73	49	I	Uppercase I
74	4A	J	Uppercase J
75	4B	K	Uppercase K
76	4C	L	Uppercase L
77	4D	M	Uppercase M
78	4E	N	Uppercase N
79	4F	O	Uppercase O
80	50	P	Uppercase P

81	51	Q	Uppercase Q
82	52	R	Uppercase R
83	53	S	Uppercase S
84	54	T	Uppercase T
85	55	U	Uppercase U
86	56	V	Uppercase V
87	57	W	Uppercase W
88	58	X	Uppercase X
89	59	Y	Uppercase Y
90	5A	Z	Uppercase Z
91	5B	[	Opening bracket
92	5C	\	Backslash
93	5D	]	Closing bracket
94	5E	^	Caret - circumflex
95	5F	_	Underscore
96	60	`	Grave accent
97	61	a	Lowercase a

98	62	b	Lowercase b
99	63	c	Lowercase c
100	64	d	Lowercase d
101	65	e	Lowercase e
102	66	f	Lowercase f
103	67	g	Lowercase g
104	68	h	Lowercase h
105	69	i	Lowercase i
106	6A	j	Lowercase j
107	6B	k	Lowercase k
108	6C	l	Lowercase l
109	6D	m	Lowercase m
110	6E	n	Lowercase n
111	6F	o	Lowercase o
112	70	p	Lowercase p
113	71	q	Lowercase q
114	72	r	Lowercase r

115	73	s	Lowercase s
116	74	t	Lowercase t
117	75	u	Lowercase u
118	76	v	Lowercase v
119	77	w	Lowercase w
120	78	x	Lowercase x
121	79	y	Lowercase y
122	7A	z	Lowercase z
123	7B	{	Opening brace
124	7C	|	Vertical bar
125	7D	}	Closing brace
126	7E	~	Equivalency sign - tilde