Portfolio: API Spec

CS 493: Cloud Application Development

Spring 2022

Oregon State University

Megan Steele (ONID marshmeg)

Last Update: June 3, 2022

URL

https://cs493-portfolio-marshmeg.uc.r.appspot.com

URL for account creation/login

https://cs493-portfolio-marshmeg.uc.r.appspot.com

Data Model

The app stores three kinds of entities in Datastore: Users, Playlists, and Songs.

Users

Property	**Data Type**	**Notes**
id	Integer	Unique ID of the user; derived from the sub field of the JWT. No changes are made to the decrypted sub field, since Google does not prepend any additional information.

		Used to match up users after login to existing user entities
firstname	String	First name of the user
lastname	String	Last name of the user
email	String	Email address associated with the user
self	String	URL pointing to the canonical representation of this user

Playlists

Property	**Data Type**	**Notes**
id	Integer	The id of the playlist. Datastore automatically generates it.
owner	Integer	The id of the user who owns this playlist.
shuffle	String	Allowed values are “true” and “false”. (Primarily because Postman did not do well with

		submitting boolean values.) If true, the playlist will shuffle by default. If false, the playlist will play sequentially by default.
name	String	Name of this playlist
description	String	Description of this playlist
songs	List	IDs of the songs that are part of this playlist
self	String	URL pointing to the canonical representation of this playlist

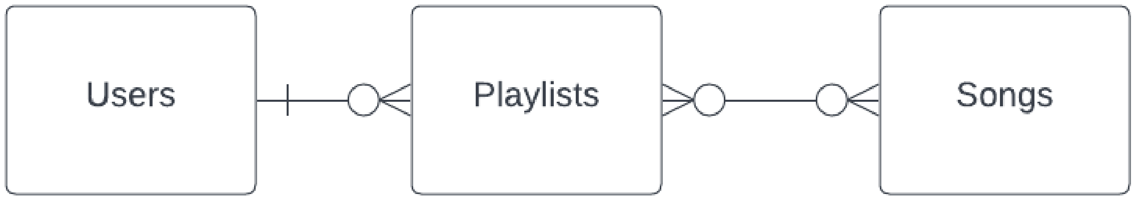
Songs

Property	**Data Type**	**Notes**
id	Integer	The id of the song. Datastore automatically generates it.
name	String	Name of the song
length	Integer	Length of the song in seconds. Must be between 1 and 2^20.

artist	String	Name of the artist who performs this song
album	String	Name of the album where this song appears
playlists	List	The ids of the playlists that contain this song. Required since relationships between entities must be displayed as part of both entities' properties.
self	String	URL pointing to the canonical representation of this song

A user may own a playlist. This relationship is one to many with required participation; a user may own no playlists, or many playlists. A playlist must be owned by exactly one user. A playlist may not have no owner; if a user is deleted, their playlists will also be deleted. (TODO : may not support user deletion)

The relationship between playlists and songs is many to many with optional participation. A playlist may contain many songs, or no songs. A song may be part of many playlists, or no playlists. The relationship is tracked as part of the playlist and song entities.



Info Modeling Note

A user’s unique id is derived from the JWT; it is validated/decrypted, and the resulting “sub” property is used directly as the unique identifier “id” in Datastore. When a user makes a request, the only information they must provide is the JWT.

Google does not prepend anything to sub (i.e. the value I receive is “1234”, not “auth0|1234”). Therefore no changes need to be made. For example, if querying a user’s playlists, the endpoint would be of the form “GET /playlists”, and require a JWT to be submitted as a bearer token. The JWT would be decrypted, and the sub field (i.e. “1234”) would be used to query Datastore, filtering for playlists owned by the user with an id of 1234.

When a request is submitted to a protected endpoint, the request must contain the JWT as a bearer token in the Authentication header. No other user identification is required.

Create a User

Allows new users to create accounts via Google OAuth and JWTs. Users may not be modified, and are created using data received from the Google OAuth flow. The scope is set to include 'https://www.googleapis.com/auth/userinfo.email' and 'https://www.googleapis.com/auth/userinfo.profile', which provide access to user names and email addresses (non-sensitive profile information).

```
Handled via root URL in-browser
```

Request

Path Parameters

None

Request Body

Handled via Google OAuth flow

Google OAuth/API Properties used

Name	**Description**	**Required?**
sub	The sub attribute of the decrypted JWT is used as a unique identifier. This is provided to Datastore as “id”.	Yes
given_name	The given_name attribute of the Google OAuth response is used to identify the user’s first name. This is provided to Datastore as “firstname”.	Yes

last_name	The family_name attribute of the Google OAuth response is used to identify the user’s first name. This is provided to Datastore as “lastname”.	Yes
email	The family_name attribute of the Google OAuth response is used to identify the user’s first name. This is provided to Datastore as “email”.	Yes

List all Users

Allows you to get a list of all users registered in the app. The list will be paginate; give users per page; pagination implemented via offsets and limits. Depending on the number of existing users, a Next attribute may be added as appropriate. See below for an example.

Note: If invalid methods (such as DELETE) are used against this endpoint, error code 405 will be returned.

This endpoint is not protected.

```
GET /users
```

Request

Path Parameters

None

Request Headers

Accept header must include application/json

Request Body

None

Response

Response Body Format

JSON

Response Statuses

Outcome	Status Code	Notes

Success	200 OK	Paginated via offsets and limits.
Failure	406 Not Acceptable	If the request’s Accept header does not contain application/json, 406 status code is returned

Response Examples

Success

```
Status: 200 OK
{
    "songs": [
        {    "id”: 123,
             “name": "I’m mn Gonna Be (500 Miles)",
             "artist": “The Proclaimers”,
             “album”: “Sunshine on Leith”,
             "length": 219
             "playlists": [],
             "self": “http://baseUrl/songs/123”
        },
        {    “id”: 456,
             "name": "Don’t Get Me Wrong",
             "artist": “The Pretenders”,
             “album”: “ “,
             "length": 239
             "self": “http://baseUrl/songs/456”
        }
    ]
}
```

```
If more than 5 users are present, an additional attribute will
be included in the response:
“next”: “http://baseUrl/users?limit=5&offset=5”
```

Failure

```
Status: 406 Not Acceptable
{
     “Error”: “No match was found between the request's Accept
header and the available options for this endpoint”
}
```

Create a Song

Allows new songs to be created. Note that songs do not require unique properties, since songs may be covered by multiple artists; an artist might perform different versions of the same song on different albums. If additional song properties are provided in the request, they are ignored.

Since songs are not associated with users, this endpoint is unprotected and does not require a JWT. If one is provided, it will be ignored.

```
POST /songs
```

Request

Accepts MIME Type

application/json

Path Parameters

None

Request Headers

Accept header must include application/json

Request Body

Required

Request Body Format

JSON

Request JSON Attributes

Name	**Description**	**Required?**

name	The name of the song	Yes
artist	The artist who performs this song	Yes
album	The album where this song appears	No
length	Length of the song in seconds. Must be an integer value between 1 and 1,048,576.	Yes

Request Body Example

```
{
    "name": "I’m Gonna Be (500 Miles)",
    "artist": “The Proclaimers”,
    “album”: “Sunshine on Leith”,
    "length": 219
}
```

```
{
    "name": "Don’t Get Me Wrong",
    "artist": “The Pretenders”,
    "length": 239
}
```

Response

Response Body Format

JSON

Response Statuses

Outcome	**Status Code**	**Notes**
Success	201 Created	

Failure	400 Bad Request	If the request is missing any of the 3 required attributes, or has an invalid specified length, the song must not be created, and 400 status code must be returned.
Failure	406 Not Acceptable	If the request’s Accept header does not contain application/json, the song is not be created, and 406 status code is returned
Failure	415 Unsupported Media Type	If the request is not JSON, the song is not created and 415 status code is returned.

Response Examples

- Songs are initialized with an empty playlists list.

Success

```
Status: 201 Created
{
    "id”: 123,
    "name": "I’m Gonna Be (500 Miles)",
    "artist": “The Proclaimers”,
    “album”: “Sunshine on Leith”,
    "length": 219
    "playlists": [],
    "self": “http://baseUrl/songs/123”
}
```

Failure

```
Status: 400 Bad Request
{
     “Error”: “The request object is missing at least one of
the required attributes”
}
```

```
Status: 400 Bad Request
{
     “Error”: “Valid song lengths are between 1 and 1048576”
}
```

```
Status: 406 Not Acceptable
{
     “Error”: “No match was found between the request's Accept
header and the available options for this endpoint”
}
```

```
Status: 415 Unsupported Media Type
{
     “Error”: “Only application/json MIMEtype is accepted”
}
```

Update Partial Song Attributes

Allows you to update an existing song. The request must include values for at least one of the following: song name, artist, album, and length. Note that playlist membership cannot be modified from this endpoint; this is controlled via the /playlists/:playlist_id/songs/:song_id endpoint.

If additional attributes are provided, they are ignored by the application. This includes ID; the song ID is determined by Datastore and cannot be affected or modified by requests.

NOTE: I chose to **allow PATCH requests with all four attributes** to be updated. Decision made based on https://www.rfc-editor.org/rfc/rfc5789, which states that a PATCH request may be made in such a way as to be idempotent.

I opted for status code 200 as my success code based on https://edstem.org/us/courses/21379/discussion/1448753.

Since songs are not associated with users, this endpoint is unprotected and does not require a JWT. If one is provided, it will be ignored.

```
PATCH /songs/:song_id
```

Request

Path Parameters

Name	**Description**
song_id	The id of the song.

Accepts MIME Type

application/json

Path Parameters

None

Request Headers

Accept header must include application/json

Request Body

Required

Request Body Format

JSON

Request JSON Attributes

Name	Description	Required?
name	The name of the song	At least one of these values must be present.
artist	The artist who performs this song	
album	The album where this song appears	
length	Length of the song in seconds. Must be an integer value between 1 and 1,048,576.	

Request Body Example

```
{
    "name": "I’m Gonna Be (500 Miles)",
    "artist": “The Proclaimers”,
    “album”: “Sunshine on Leith”,
    "length": 219
}
```

```
{
    "name": "Don’t Get Me Wrong",
    "artist": “The Pretenders”,
    "length": 239
}
```

Response

Response Body Format

JSON

Response Statuses

Outcome	Status Code	Notes
Success	200 OK	Updated song information will be in the response body.
Failure	400 Bad Request	If the request is missing all of the required attributes, or has an invalid specified length, the song must not be created, and 400 status code must be returned.
Failure	404 Not Found	No song with this song_id exists
Failure	406 Not Acceptable	If the request's Accept header does not contain application/json, the song is not be created, and 406 status code is returned
Failure	415 Unsupported Media Type	If the request is not JSON, the song is not created and 415 status code is returned.

Response Examples

- Songs are initialized with an empty playlists list.

Success

```
Status: 201 Created
{
    "id”: 123,
    "name": "I’m Gonna Be (500 Miles)",
    "artist": “The Proclaimers”,
    “album”: “Sunshine on Leith”,
    "length": 219
    "playlists": [],
    "self": “http://baseUrl/songs/123”
}
```

Failure

```
Status: 400 Bad Request
{
     “Error”: “The request object is missing at least one of
the required attributes”
}
```

```
Status: 400 Bad Request
{
     “Error”: “Valid song lengths are between 1 and 1048576”
}
```

```
Status: 404 Not Found
{
     “Error”: “No song with this song_id exists”
}
```

```
Status: 406 Not Acceptable
{
     “Error”: “No match was found between the request's Accept
header and the available options for this endpoint”
}
```

```
Status: 415 Unsupported Media Type
{
     “Error”: “Only application/json MIMEtype is accepted”
}
```

Update Full Song Attributes

Allows you to update an existing song. The request must include values for all required fields: song name, artist, and length. It may optionally include a value for album. Note that playlist membership cannot be modified from this endpoint; this is controlled via the /playlists/:playlist_id/songs/:song_id endpoint.

If additional attributes are provided, they are ignored by the application. This includes ID; the song ID is determined by Datastore and cannot be affected or modified by requests.

This endpoint is not protected.

```
PUT /songs/:song_id
```

Request

Path Parameters

Name	Description
song_id	The id of the song.

Accepts MIME Type

application/json

Path Parameters

None

Request Headers

Accept header must include application/json

Request Body

Required

Request Body Format

JSON

Request JSON Attributes

Name	Description	Required?
name	The name of the song	Yes
artist	The artist who performs this song	Yes
album	The album where this song appears	No
length	Length of the song in seconds. Must be an integer value between 1 and 1,048,576.	Yes

Request Body Example

```
{
    "name": "I’m Gonna Be (500 Miles)",
    "artist": “The Proclaimers”,
    “album”: “Sunshine on Leith”,
    "length": 219
}
```

```
{
    "name": "Don’t Get Me Wrong",
    "artist": “The Pretenders”,
    "length": 239
}
```

Response

Response Body Format

JSON

Response Statuses

Outcome	**Status Code**	**Notes**
Success	303 See Other	Successful POST responses will have a link to the updated song in the header field Location.
Failure	400 Bad Request	If the request is missing all of the required attributes, or has an invalid specified length, the song must not be created, and 400 status code must be returned.
Failure	404 Not Found	No song with this song_id exists
Failure	406 Not Acceptable	If the request's Accept header does not contain application/json, the song is not be created, and 406 status code is returned
Failure	415 Unsupported Media Type	If the request is not JSON, the song is not created and 415 status code is returned.

Response Examples

- Songs are initialized with an empty playlists list.

Success

```
Status: 303 See Other
```

Failure

```
Status: 400 Bad Request
{
     “Error”: “The request object is missing at least one of
the required attributes”
}
```

```
Status: 400 Bad Request
{
     “Error”: “Valid song lengths are between 1 and 1048576”
}
```

```
Status: 404 Not Found
{
     “Error”: “No song with this song_id exists”
}
```

```
Status: 406 Not Acceptable
{
     “Error”: “No match was found between the request's Accept
header and the available options for this endpoint”
}
```

```
Status: 415 Unsupported Media Type
{
     “Error”: “Only application/json MIMEtype is accepted”
}
```

Get a Song

Allows you to get an existing song.

This endpoint is not protected.

```
GET /songs/:song_id
```

Request

Path Parameters

Name	Description
song_id	The id of the song.

Request Headers

Accept header must include application/json

Request Body

None

Response

Response Body Format

JSON

Response Statuses

Outcome	Status Code	Notes
Success	200 OK	
Failure	404 Not Found	No song with this song_id exists

Failure	406 Not Acceptable	If the request’s Accept header does not contain application/json, 406 status code is returned

Response Examples

Success

```
Status: 200 OK
{
    "id”: 123,
    “name": "I’m Gonna Be (500 Miles)",
    "artist": “The Proclaimers”,
    “album”: “Sunshine on Leith”,
    "length": 219
    "playlists": [],
    "self": “http://baseUrl/songs/123”
}
```

Failure

```
Status: 404 Not Found
{
     “Error”: “No song with this song_id exists”
}
```

```
Status: 406 Not Acceptable
{
     “Error”: “No match was found between the request's Accept
header and the available options for this endpoint”
}
```

Delete a Song

Allows you to delete an existing song.

Playlists that contain this song are updated; the song ID is removed from each playlist's songs list.

This endpoint is not protected.

```
DELETE /songs/:song_id
```

Request

Path Parameters

Name	Description
song_id	The id of the song.

Request Body

None

Response

Response Body Format

JSON

Response Statuses

Outcome	Status Code	Notes
Success	204 No Content	Removes this song from any playlists where it was present
Failure	404 Not Found	No song with this song_id exists

Response Examples

Success

```
Status: 204 No Content
```

Failure

```
Status: 404 Not Found
{
     “Error”: “No song with this song_id exists”
}
```

List all Songs

Allows you to get a paginated list of all songs. Five songs per page; pagination implemented via offsets and limits. Depending on the number of existing songs, a Next attribute may be added as appropriate. See below for an example.

Note: If invalid methods (such as DELETE) are used against this endpoint, error code 405 will be returned.

This endpoint is not protected.

```
GET /songs
```

Request

Path Parameters

None

Request Headers

Accept header must include application/json

Request Body

None

Response

Response Body Format

JSON

Response Statuses

Outcome	**Status Code**	**Notes**
Success	200 OK	Paginated via offsets and limits.
Failure	406 Not Acceptable	If the request’s Accept header does not contain application/json, 406 status code is returned

Response Examples

Success

```
Status: 200 OK
{
    "songs": [
        {    "id”: 123,
             “name": "I’m mn Gonna Be (500 Miles)",
             "artist": “The Proclaimers”,
             “album”: “Sunshine on Leith”,
             "length": 219
             "playlists": [],
             "self": “http://baseUrl/songs/123”
        },
        {    “id”: 456,
             "name": "Don’t Get Me Wrong",
             "artist": “The Pretenders”,
             “album”: “ “,
             "length": 239
             "self": “http://baseUrl/songs/456”
        }
    ]
}
```

```
If more than 5 songs are present, an additional attribute will
be included in the response:
“next”: “http://baseUrl/songs?limit=5&offset=5”
```

Failure

```
Status: 406 Not Acceptable
{
     “Error”: “No match was found between the request's Accept
header and the available options for this endpoint”
}
```

Create a Playlist

Allows new playlist to be created. Note that playlists do not require unique properties.

Since playlists are associated with users, this endpoint is protected and requires a JWT. The corresponding user will be the playlist’s owner.

Note that playlists are initialized with an empty songs list.

```
POST /songs
```

Request

Authorization

A valid JWT must be provided as a bearer token

Accepts MIME Type

application/json

Path Parameters

None

Request Headers

Accept header must include application/json

Request Body

Required

Request Body Format

JSON

Request JSON Attributes

Name	**Description**	**Required?**
name	The name of the playlist	Yes
shuffle	If true, the playlist will shuffle by default. If false, the playlist will play sequentially by default.	Yes
description	Description of this playlist	Yes

Request Body Example

```
{
    "name": "Study Playlist",
    "shuffle": “true”,
    “description”: “chill songs for focusing”
}
```

Response

Response Body Format

JSON

Response Statuses

Outcome	**Status Code**	**Notes**

Success	201 Created	
Failure	400 Bad Request	If the request is missing any of the 3 required attributes, or has an invalid shuffle value, the playlist must not be created, and 400 status code must be returned.
Failure	406 Not Acceptable	If the request's Accept header does not contain application/json, the playlist is not be created, and 406 status code is returned
Failure	415 Unsupported Media Type	If the request is not JSON, the playlist is not created and 415 status code is returned.

Response Examples

- Playlists are initialized with an empty songs list.

Success

```
Status: 201 Created
{
    "id”: 123,
    "name": "Study Playlist",
    "shuffle”: “true”,
    “description”: “chill songs for focusing”,
    "songs": [],
    "self": “http://baseUrl/playlists/123”
}
```

Failure

```
Status: 400 Bad Request
{
     “Error”: “The request object is missing at least one of
the required attributes”
}
```

```
Status: 400 Bad Request
{
     “Error”: “Valid shuffle values are true and false”
}
```

```
Status: 406 Not Acceptable
{
     “Error”: “No match was found between the request's Accept
header and the available options for this endpoint”
}
```

```
Status: 415 Unsupported Media Type
{
     “Error”: “Only application/json MIMEtype is accepted”
}
```

List all Playlists

Allows you to get a paginated list of all playlists owned by a specific user. Five playlists per page; pagination implemented via offsets and limits. Depending on the number of existing playlists, a Next attribute may be added as appropriate. See below for an example.

Note: If invalid methods (such as DELETE) are used against this endpoint, error code 405 will be returned.

This endpoint requires a JWT as it is protected.

```
GET /playlists
```

Request

Authorization

A valid JWT must be provided as a bearer token

Path Parameters

None

Request Headers

Accept header must include application/json

Request Body

None

Response

Response Body Format

JSON

Response Statuses

Outcome	Status Code	Notes
Success	200 OK	Paginated via offsets and limits. If the user exists but owns no playlists, the results will be empty.
Failure	401 Unauthorized	If the request does not have a valid JWT, 401 status code is returned
Failure	406 Not Acceptable	If the request's Accept header does not contain application/json, 406 status code is returned

Response Examples

Success

```
Status: 200 OK
{
    “playlists": [
        {
            "id”: 123,
             "name": "Study Playlist",
             "shuffle”: “true”,
             “description”: “chill songs for focusing”,
             "songs": [],
             "self": “http://baseUrl/playlists/123”
        }
        {    “id”: 456,
             "name": "Party Playlist",
             "shuffle”: “false”,
             “description”: “songs for dancing”,
             "songs": [],
             "self": “http://baseUrl/playlists/456”
        }
    ]
```

}
If more than 5 playlists are present, an additional attribute will be included in the response: “next”: “http://baseUrl/playlists?limit=5&offset=5”

Failure

```
Status: 401 Not Authorized
{
     “Error”: “Invalid token”
}
```

```
Status: 406 Not Acceptable
{
     “Error”: “No match was found between the request's Accept
header and the available options for this endpoint”
}
```

Update Partial Playlist Attributes

Allows you to update an existing playlist. The request must include values for at least one of the following: playlist name, description, or shuffle. Note that playlist membership cannot be modified from this endpoint; this is controlled via the /playlists/:playlist_id/songs/:song_id endpoint.

If additional attributes are provided, they are ignored by the application. This includes ID; the playlist ID is determined by Datastore and cannot be affected or modified by requests.

NOTE: I chose to **allow PATCH requests with all three attributes** to be updated. Decision made based on https://www.rfc-editor.org/rfc/rfc5789, which states that a PATCH request may be made in such a way as to be idempotent.

I opted for status code 200 as my success code based on https://edstem.org/us/courses/21379/discussion/1448753.

Since playlists are associated with users, all requests to this endpoint require a valid JWT that corresponds to the user who owns the playlist.

```
PATCH /playlists/:playlist_id
```

Request

Path Parameters

Name	Description
playlist_id	The id of the playlist.

Authorization

A valid JWT must be provided as a bearer token

Accepts MIME Type

application/json

Path Parameters

None

Request Headers

Accept header must include application/json

Request Body

Required

Request Body Format

JSON

Request JSON Attributes

Name	**Description**	**Required?**
name	The name of the playlist	At least one of these values must be present.
description	Description of the playlist	
shuffle	If true, the playlist will shuffle by default. If false, the playlist will play sequentially by default.	

Request Body Example

```
{
    "name": "Study Playlist",
    "shuffle": "false",
    "description": "songs for focusing and coding"
}
```

Response

Response Body Format

JSON

Response Statuses

Outcome	**Status Code**	**Notes**
Success	200 OK	Updated playlist information will be in the response body.
Failure	400 Bad Request	If the request is missing all of the required attributes, or has an invalid shuffle value, the playlist must not be changed, and 400 status code must be returned.
Failure	403 Forbidden	If the provided JWT does not correspond to the playlist owner, no changes are made and 403 status code is returned.
Failure	404 Not Found	No playlist with this playlist_id exists
Failure	406 Not Acceptable	If the request's Accept header does not contain application/json, the song is not be created, and 406 status code is returned

Failure	415 Unsupported Media Type	If the request is not JSON, the song is not created and 415 status code is returned.

Response Examples

Success

```
Status: 200 OK
{
    "id”: 123,
     "name": "Road Trip Playlist",
     "description”: “songs for driving”,
     “shuffle”: true,
     "songs": [],
     "self": “http://baseUrl/playlists/123”
}
```

Failure

```
Status: 400 Bad Request
{
     “Error”: “The request object is missing any recognized
attributes”
}
```

```
Status: 400 Bad Request
{
     “Error”: “Valid shuffle values are true and false”
}
```

```
Status: 403 Forbidden
```

```
Status: 404 Not Found
{
     “Error”: “No playlist with this playlist_id exists”
}
```

```
Status: 406 Not Acceptable
{
     “Error”: “No match was found between the request's Accept
header and the available options for this endpoint”
}
```

```
Status: 415 Unsupported Media Type
{
     “Error”: “Only application/json MIMEtype is accepted”
}
```

Update Full Song Attributes

Allows you to update an existing playlist. The request must include values for playlist name, description, and shuffle. Note that playlist membership cannot be modified from this endpoint; this is controlled via the /playlists/:playlist_id/songs/:song_id endpoint.

If additional attributes are provided, they are ignored by the application. This includes ID; the playlist ID is determined by Datastore and cannot be affected or modified by requests.

Since playlists are associated with users, all requests to this endpoint require a valid JWT that corresponds to the user who owns the playlist.

```
PUT /playlists/:playlist_id
```

Request

Path Parameters

Name	**Description**
playlist_id	The id of the playlist.

Authorization

A valid JWT must be provided as a bearer token

Accepts MIME Type

application/json

Path Parameters

None

Request Headers

Accept header must include application/json

Request Body

Required

Request Body Format

JSON

Request JSON Attributes

Name	**Description**	**Required?**
name	The name of the song	Yes
artist	The artist who performs this song	Yes
album	The album where this song appears	No
length	Length of the song in seconds. Must be an integer value between 1 and 1,048,576.	Yes

Request Body Example

```
{
    "name": "I’m Gonna Be (500 Miles)",
    "artist": “The Proclaimers”,
    “album”: “Sunshine on Leith”,
    "length": 219
}
```

```
{
    "name": "Don’t Get Me Wrong",
    "artist": “The Pretenders”,
    "length": 239
}
```

Response

Response Body Format

JSON

Response Statuses

Outcome	Status Code	Notes
Success	303 See Other	Successful POST responses will have a link to the updated playlist in the header field Location.
Failure	400 Bad Request	If the request is missing any of the required attributes, or has an invalid shuffle value, the playlist is not changed, and 400 status code must be returned.
Failure	403 Forbidden	If the provided JWT does not correspond to the playlist owner, no changes are made and 403 status code is returned.
Failure	404 Not Found	No playlist with this playlist_id exists
Failure	406 Not Acceptable	If the request's Accept header does not contain application/json, the playlist is not changed, and 406 status code is returned
Failure	415 Unsupported Media Type	If the request is not JSON, the playlist is not changed and 415 status code is returned.

Success

```
Status: 303 See Other
```

Failure

```
Status: 400 Bad Request
{
     "Error": "The request object is missing at least one of the required attributes"
}
```

```
Status: 400 Bad Request
{
     "Error": "Valid shuffle values are true and false"
}
```

```
Status: 403 Forbidden
```

```
Status: 404 Not Found
{
     "Error": "No playlist with this playlist_id exists"
}
```

```
Status: 406 Not Acceptable
{
     "Error": "No match was found between the request's Accept header and the available options for this endpoint"
}
```

```
Status: 415 Unsupported Media Type
{
     "Error": "Only application/json MIMEtype is accepted"
}
```

Delete a Playlist

Allows you to delete an existing playlist.

Since playlists are associated with users, this endpoint is protected; all requests to this endpoint require a valid JWT that corresponds to the user who owns the playlist.

Songs that are part of this playlist are updated; the playlist ID is removed from each song’s playlists list.

```
DELETE /playlist/:playlist_id
```

Request

Path Parameters

Name	Description
playlist_id	The id of the playlist.

Authorization

A valid JWT must be provided as a bearer token

Request Body

None

Response

Response Body Format

JSON

Response Statuses

Outcome	Status Code	Notes

Success	204 No Content	Removes relationships between this playlist and its songs
Failure	403 Forbidden	If the provided JWT does not correspond to the playlist owner, no changes are made and 403 status code is returned.
Failure	404 Not Found	No song with this song_id exists

Response Examples

Success

```
Status: 204 No Content
```

Failure

```
Status: 403 Forbidden
```

```
Status: 404 Not Found
{
    “Error”: “No song with this song_id exists”
}
```

Create a Load

Allows you to create a new load.

```
POST /loads
```

Request

Path Parameters

None

Request Body

Required

Request Body Format

JSON

Request JSON Attributes

Name	**Description**	**Required?**
volume	The space the load occupies.	Yes
item	What is in the load.	Yes
creation_date	When the load was created.	Yes

Request Body Example

```
{
    "volume": 5,
    "item": "Cat toys",
    "creation_date": “04/19/2022”
}
```

Response

Response Body Format

JSON

Response Statuses

Outcome	Status Code	Notes
Success	201 Created	
Failure	400 Bad Request	If the request is missing any of the 3 required attributes, the load must not be created, and 400 status code must be returned.

Response Examples

- Loads are initialized with no carrier (i.e. they are not initially on a boat).

Success

```
Status: 201 Created
{
    “id”: 123,
    "volume": 5,
    "item": "Cat toys",
    "creation_date": “04/19/2022
    “carrier”: null,
    "self": “http://baseUrl/boats/123”
}
```

Failure

```
Status: 400 Bad Request
{
     “Error”: “The request object is missing at least one of
the required attributes”
}
```

Get a Load

Allows you to get an existing load.

```
GET /loads/:load_id
```

Request

Path Parameters

Name	Description
load_id	The id of the load.

Request Body

None

Response

Response Body Format

JSON

Response Statuses

Outcome	Status Code	Notes
Success	200 OK	
Failure	404 Not Found	No load with this load_id exists

Response Examples

Success

```
Status: 200 OK
{
    “id”: 123,
    "volume": 5,
    "item": "Cat toys",
    "creation_date": “04/19/2022
    “carrier”: null,
    "self": “http://baseUrl/loads/123”
}
```

Failure

```
Status: 404 Not Found
{
     “Error”: “No load with this load_id exists”
}
```

Delete a Load

Allows you to delete an existing load. Removes the load from any carrie if applicable.

```
DELETE /loads/:load_id
```

Request

Path Parameters

Name	**Description**
load_id	The id of the load.

Request Body

None

Response

Response Body Format

JSON

Response Statuses

Outcome	**Status Code**	**Notes**
Success	204 No Content	Removes this load from its carrier (if a carrier existed)
Failure	404 Not Found	No load with this load_id exists

Response Examples

Success

```
Status: 204 No Content
```

Failure

```
Status: 404 Not Found
{
     “Error”: “No load with this load_id exists”
}
```

List all Loads

Allows you to get a paginated list of all loads. Three loads per page; pagination implemented via offsets and limits. Depending on the number of existing boats, a Next attribute may be added as appropriate. See below for an example.

```
GET /loads
```

Request

Path Parameters

None

Request Body

None

Response

Response Body Format

JSON

Response Statuses

Outcome	**Status Code**	**Notes**
Success	200 OK	Paginated via offsets and limits.

Response Examples

Success

```
Status: 200 OK
{
    "loads": [
        {“id”: 123,
        "volume": 5,
        "item": "Cat toys",
        "creation_date": “04/19/2022
        “carrier”: null,
        "self": “http://baseUrl/boats/123”},
        {“id”: 456,
        "volume": 6,
        "item": "Snacks",
        "creation_date": “04/19/2022
        “carrier”: null,
        "self": “http://baseUrl/boats/456”}
    ]
}
```

If more than 3 loads are present, an additional attribute will be included in the response:
“next”: “http://baseUrl/loads?limit=3&offset=3”

Song is Added to a Playlist

Allows you to add a song to a playlist. The playlist now contains the song id in its songs list, and the song now contains the playlist id in its playlist list.

Since playlists are associated with users, this endpoint is protected; all requests to this endpoint require a valid JWT that corresponds to the user who owns the playlist.

```
PUT /playlists/:playlist_id/songs/:song_id
```

Request

Path Parameters

Name	Description
playlist_id	The id of the playlist.
song_id	The id of the song.

Authorization

A valid JWT must be provided as a bearer token

Request Body

None

Response

Response Body Format

JSON

Response Statuses

Outcome	Status Code	Notes
Success	204 No Content	

Failure	403 Forbidden	The playlist does not belong the user associated with the provided JWT.
Failure	404 Not Found	The specified playlist and/or song does not exist.

Response Examples

Success

```
Status: 204 No Content
```

Failure

```
Status: 403 Forbidden
```

```
Status: 404 Not Found
{
     “Error”: “The specified playlist and/or song does not
exist”
}
```

Song is Removed from a Playlist

Allows you to remove a song from a playlist.

Since playlists are associated with users, this endpoint is protected; all requests to this endpoint require a valid JWT that corresponds to the user who owns the playlist.

```
DELETE /playlists/:playlist_id/songs/:songs_id
```

Request

Path Parameters

Name	Description
playlist_id	The id of the playlist.
song_id	The id of the song

Authorization

A valid JWT must be provided as a bearer token

Request Body

None

Response

Response Body Format

JSON

Response Statuses

Outcome	Status Code	Notes
Success	204 No Content	

Failure	403 Forbidden	The playlist does not belong the user associated with the provided JWT.
Failure	404 Not Found	The specified playlist and/or song does not exist, , or both exist but the song is not currently in this playlist.

Response Examples

Success

```
Status: 204 No Content
```

Failure

```
Status: 403 Forbidden
```

```
Status: 404 Not Found
{
     “Error”: “The specified playlist does not contain the
specified song”
}
```