Final Project: Data Model and API Spec

CS 493: Cloud Application Development

Fall 2023

Oregon State University

Last Update: Dec 11. 12:30 pm Pacific

Project URL: https://roulearo-final.wl.r.appspot.com/

Change log

Version	Change	Date
1.0	Initial version.	Dec 11, 2023

Data Model

This app stores three kinds of entities in Datastore - Users, Playlists, and Songs.

There is a one-to-many relationship between Users and Playlists. Playlists have a many-to-many relationship with songs.

When a new User is created, the value of "sub" in the JWT is stored as the unique identifier for that user in the "owner" property.

When a request is made to a protected endpoint, the Authorization header must be set to Bearer Token. The token should be a valid JWT generated by logging in or creating a new account at https://roulearo-final.wl.r.appspot.com. The value of "sub" in this JWT is used to authorize the request.

User

Users have a one-to-many relationship with Playlists, but the playlists are not stored in the User entity.

Property	**Data Type**	**Notes**
id	Integer	The id of the user that is automatically generated by Datastore.
owner	String	Unique ID associated with the owner of this User. The value of "owner" is the value of "sub" in the JWT when the User is created.

Playlist

Playlists have a many-to-one relationship with a User. Playlists also have a many-to-many relationship with Songs.

Property	**Data Type**	**Notes**
id	Integer	The id of the playlist that is automatically generated by Datastore. Don't add it yourself as a property of the entity.
name	String	Name of the playlist. Must be unique across all playlists that belong to the owner. Limited to 100 characters.
description	String	A brief description of the playlist. Limited to 300 characters.
owner	String	Unique ID associated with the owner of this Playlist. The value of "owner" is the value of "sub" in the JWT when the Playlist is created.
song_count	Integer	The total number of songs contained in this playlist. This is set automatically when a song is added or removed from a playlist.
songs	JSON	JSON containing integer "id", string "name", and string "self" fields representing the song ID, name, and URL to directly access that song.

Song

Songs have a many-to-many relationship with Playlists.

Property	**Data Type**	**Notes**

id	Integer	The id of the song that is automatically generated by Datastore.
title	String	The title of the song. Limited to 50 upper-case or lower-case alphabetic characters.
artist	String	The artist credited for this song. Limited to 50 characters.
album	String	The album this song is from. This can be an empty string if the song is not part of an album.

Account Creation and Login

This app integrates with Auth0 for user account creation and logging in. To create a new account or log in to an existing one, visit https://roulearo-final.wl.r.appspot.com and register using a valid email address and password. An encoded JWT will be supplied that can be used in requests to protected endpoints.

View all Users

Unprotected Endpoint

Allows you to view all existing usernames and public user information.

GET /users

Request

Headers

Accept - application/json

Path Parameters

None

Request Body

None

Response

Response Body Format

JSON

Response Statuses

Outcome	Status Code	Notes
Success	200 OK	
Failure	404 Not Found	No users exist.

Response Examples

Success

```
Status: 200 OK

[
 {
  "id": 123,
  “owner”: “123abc”
 },
 {
  "id": 456,
  “owner”: “456def”
 },
]
```

Failure

```
Status: 404 Not Found

{
"Error":  "No users exist"
}
```

Create a Playlist

Protected Endpoint

Allows you to create a new playlist for a user. The value of name must be unique across all existing playlists that belong to this user. Extraneous attributes are ignored. Race conditions are not handled, such as when two or more requests simultaneously try to create and/or update a playlist.

```
POST /playlists
```

Request

Headers

Content-type - application/json

Accept - application/json

Path Parameters

None

Request Body

Required

Request Body Format
JSON

Request JSON Attributes

Name	Description	Required?
name	Name of the playlist. Must be unique across all playlists that belong to the owner. Limited to 100 characters.	Yes
description	A brief description of the playlist. Limited to 300 characters.	No

Request Body Example

```
{
  "name": "Playlist1"
}
```

Response

Response Body Format
JSON

Response Statuses

Outcome	Status Code	Notes
Success	201 Created	
Failure	400 Bad Request	If the request is missing the required attribute or the value is invalid, the playlist is not created, and 400 status code is returned.
Failure	401 Unauthorized	Invalid JWT in the request.
Failure	403 Forbidden	If the name in the request is not unique across all playlists that belong to this user, the playlist is not created, and 403 status code is returned.

Response Examples

· Datastore will automatically generate an ID and store it with the entity being created. The server will also generate a URL that links directly to the resource. The app sends back these values in the response body as shown in the example.

Success

```
Status: 201 Created

{
  "id": 123,
  "name": "Playlist1"
}
```

Failure

```
Status: 400 Bad Request

{
"Error":  "The request object is missing the required attribute, or the value of
the attribute is invalid"
}
```

```
Status: 401 Unauthorized

{
"Error":  "Invalid JWT in the request"
}
```

```
Status: 403 Forbidden

{
"Error":  "The attribute name in the request object is not unique to this user"
}
```

View all Playlists

Protected Endpoint

Allows you to get all existing playlists belonging to a user.

```
GET /playlist
```

Request

Headers

Accept - application/json

Path Parameters

None

Request Body

None

Response

Response Body Format

JSON or HTML depending on the value of the accept header.

Response Statuses

Outcome	Status Code	Notes
Success	200 OK	
Failure	401 Unauthorized	Invalid JWT in the request.
Failure	404 Not Found	No playlists exist for this user.

Response Examples

Success

```
Status: 200 OK
```

```
[
 “count”: “6”,
 “next”: “http://localhost:8080/playlists?cursor=...”,
 “playlists”: [
  {
   "id": 123,
   "name": "Playlist1",
    "count": 0,
    “owner”: “auth0|523452352”,
    "self": “http://localhost:8080/playlist/123”
  },
  {
   "id": 134,
   "name": "Playlist2",
    "count": 0,
    “owner”: “auth0|523452352”,
    "self": “http://localhost:8080/playlist/134”
  },
  {
   "id": 145,
   "name": "Playlist3",
    "count": 0,
    “owner”: “auth0|523452352”,
    "self": “http://localhost:8080/playlist/145”
  },
  {
   "id": 156,
   "name": "Playlist4",
    "count": 0,
    “owner”: “auth0|523452352”,
    "self": “http://localhost:8080/playlist/156”
  },
  {
   "id": 167,
   "name": "Playlist5",
    "count": 0,
    “owner”: “auth0|523452352”,
    "self": “http://localhost:8080/playlist/167”
  },
]
```

```
Status: 401 Unauthorized

{
"Error":  "Invalid JWT in the request"
}
```

Failure

```
Status: 404 Not Found

{
"Error":  "No playlists belonging to this user exist."
}
```

View a Playlist

Protected Endpoint

Allows you to get an existing playlist belonging to a user.

```
GET /playlist/:playlist_id
```

Request

Headers

Accept - application/json

Path Parameters

Name	**Description**
playlist_id	ID of the playlist

Request Body

None

Response

Response Body Format

JSON or HTML depending on the value of the accept header.

Response Statuses

Outcome	**Status Code**	**Notes**
Success	200 OK	
Failure	404 Not Found	No user with this playlist_id exists

Response Examples

Success

```
Status: 200 OK

{
  "id": 123,
  "name": "Playlist1",
  "count": 0,
  “owner”: “auth0|523452352”,
  "self": “http://localhost:8080/playlist/123”
}
```

```
Status: 401 Unauthorized
```

```
{
"Error":  "Invalid JWT in the request"
}
```

Failure

```
Status: 404 Not Found

{
"Error":  "No playlist with this playlist_id belonging to this user"
}
```

Partially Edit a Playlist

Protected Endpoint

Allows you to edit a subset of the features for a specific playlist.

```
PATCH /playlist/:playlist_id
```

Request

Headers

Content-type - application/json

Accept - application/json

Path Parameters

Name	Description
playlist_id	ID of the playlist.

Request Body

Required

Request Body Format

JSON

Request JSON Attributes

Name	Description	Required?
name	Name of the playlist. Must be unique across all playlists that belong to the owner. Limited to 100 characters.	No
description	A brief description of the playlist. Limited to 300 characters.	No

Request Body Example

```
{
  "description": "A description of the songs in this playlist."
}
```

Response

Response Body Format

JSON

Response Statuses

Outcome	Status Code	Notes
Success	200 OK	
Failure	400 Bad Request	If the request does not contain any valid attributes, the playlist is not updated, and 400 status code is returned.
Failure	403 Forbidden	If the name in the request is not unique across all playlists, the playlist is not updated, and 403 status code is returned.
Failure	404 Not Found	No playlist with this playlist_id exists

Response Examples

Success

```
Status: 200 OK

{
  "id": 123,
  "name": "Playlist1",
  “description”: “A description of the songs in this playlist.”,
  "count": 0,
  “owner”: “auth0|523452352”,
   "self": “http://localhost:8080/playlists/123”
}
```

Failure

```
Status: 400 Bad Request

{
"Error":  "The request object is missing a required attribute, or the value of an
attribute is invalid"
}
```

```
Status: 401 Unauthorized

{
"Error":  "Invalid JWT in the request"
}
```

```
Status: 403 Forbidden

{
"Error":  "The attribute name in the request object is not unique to this user"
}
```

```
Status: 404 Not Found

{
"Error":  "No playlist with this playlist_id exists"
}
```

Edit a Playlist

Protected Endpoint

Allows you to edit the features for a specific playlist.

```
PUT /playlist/:playlist_id
```

Request

Headers

Content-type - application/json

Accept - application/json

Path Parameters

Name	Description
playlist_id	ID of the playlist

Request Body

Required

Request Body Format

JSON

Request JSON Attributes

Name	Description	Required?
name	Name of the playlist. Must be unique across all playlists that belong to the owner. Limited to 100 characters.	No
description	A brief description of the playlist. Limited to 300 characters.	No

Request Body Example

```
{
  "name": "Playlist1-2",
  "description": “A description of the songs in this playlist.”
}
```

Response

Headers

Location - The URL of the updated playlist.

Response Body Format

JSON

Response Statuses

Outcome	Status Code	Notes

Success	303 See Other	The URL of the updated playlist is returned in the Location header, but also in the response body.
Failure	400 Bad Request	If the request does not contain any valid attributes, the playlist is not updated, and 400 status code is returned.
Failure	403 Forbidden	If the name in the request is not unique across all playlists belonging to this user, the playlist is not updated, and 403 status code is returned.
Failure	404 Not Found	No playlist with this playlist_id exists

Response Examples

Success

```
Status: 200 OK

{
  "id": 123,
  "name": "Playlist1-2",
  “description”: “A description of the songs in this playlist.”,
  "count": 0,
  “owner”: “auth0|523452352”,
  "self": “http://localhost:8080/playlists/123”
}
```

Failure

```
Status: 400 Bad Request

{
"Error":  "The request object is missing a required attribute, or the value of an attribute is invalid"
}
```

```
Status: 401 Unauthorized

{
"Error":  "Invalid JWT in the request"
}
```

```
Status: 403 Forbidden

{
"Error":  "The attribute name in the request object is not unique"
}
```

```
Status: 404 Not Found

{
"Error":  "No user with this playlist_id exists"
}
```

Delete a Playlist

Protected Endpoint

Allows you to delete a playlist.

```
DELETE /playlists/:playlist_id
```

Request

Headers

Accept - application/json

Path Parameters

Name	Description
playlist_id	ID of the playlist

Request Body

None

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 playlist with this playlist_id exists

Response Examples

Success

```
Status: 204 No Content
```

```
Status: 401 Unauthorized

{
"Error":  "Invalid JWT in the request"
}
```

Failure

```
Status: 404 Not Found

{
"Error":  "No playlist with this playlist_id exists"
}
```

Add Song to a Playlist

Protected Endpoint

Allows you to add a song to a playlist.

```
POST /:playlist_id/songs/:song_id
```

Request

Headers

Accept - application/json

Path Parameters

Name	Description
playlist_id	ID of the playlist

Request Body

Request Body Example

```
{
  "song_id": “123”
}
```

Response

No body

Response Body Format

Success: No body

Failure: JSON

Response Statuses

Outcome	Status Code	Notes
Success	204 No Content	
Failure	403 Forbidden	No song with this song_id exists
Failure	404 Not Found	No playlist with this playlist_id exists

Response Examples

Success

```
Status: 204 No Content
```

```
Status: 401 Unauthorized

{
"Error":  "Invalid JWT in the request"
}
```

Failure

```
Status: 403 Forbidden

{
```

```
"Error":  "No song with this song_id exists"
}
```

Failure

```
Status: 404 Not Found

{
"Error":  "No playlist with this playlist_id exists"
}
```

Remove a Song from a Playlist

Protected Endpoint

Allows you to remove a song from a playlist.

```
DELETE /playlists/:playlist_id/:song_id
```

Request

Headers

Accept - application/json

Path Parameters

Name	**Description**
playlist_id	ID of the playlist
song_id	ID of the song

Request Body

None

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 playlist with this playlist_id exists or it does not contain this song.

Response Examples

Success

```
Status: 204 No Content
```

```
Status: 401 Unauthorized

{
"Error":  "Invalid JWT in the request"
}
```

Failure

```
Status: 404 Not Found

{
"Error":  "No playlist with this playlist_id exists or it does not contain this
song"
}
```

Create a Song

Unprotected Endpoint

Allows you to create a new song. Extraneous attributes are ignored. Race conditions are not handled, such as when two or more requests simultaneously try to create and/or update a song.

```
POST /songs
```

Request

Headers

Content-type - application/json

Accept - application/json

Path Parameters

None

Request Body

Required

Request Body Format

JSON

Request JSON Attributes

Name	**Description**	**Required?**
title	The title of the song. Limited to 50 upper-case or lower-case alphabetic characters.	Yes
artist	The artist credited for this song. Limited to 50 characters.	Yes
album	The album this song is from. This can be an empty string if the song is not part of an album.	No

Request Body Example

```
{
  "title": "Song1",
  "artist": “Artist1”,
  “album”: “Album1”
}
```

Response

Response Body Format

JSON

Response Statuses

Outcome	Status Code	Notes
Success	201 Created	
Failure	400 Bad Request	If the request is missing a required attribute or any value is invalid, the song is not created, and 400 status code is returned.

Response Examples

- Datastore will automatically generate an ID and store it with the entity being created. The server will also generate a URL that links directly to the resource. The app sends back these values in the response body as shown in the example.

Success

```
Status: 201 Created

{
  "id": 123,
  "title": "Song1",
  "artist": "Artist1",
  "album": “Album1”,
  "self": “http://localhost:8080/songs/123”
}
```

Failure

```
Status: 400 Bad Request

{
"Error":  "The request object is missing a required attribute, or the value of an
attribute is invalid"
}
```

View all Songs

Unprotected Endpoint

Allows you to get all existing songs.

```
GET /songs
```

Request

Headers

Accept - application/json

Path Parameters

None

Request Body

None

Response

Response Body Format

JSON or HTML depending on the value of the accept header.

Response Statuses

Outcome	**Status Code**	**Notes**
Success	200 OK	
Failure	404 Not Found	No songs exist.

Response Examples

Success

```
Status: 200 OK

{
 “count”: “6”,
 “next”: "http://localhost:8080/songs?cursor=…",
 “songs”: [
  {
   "id": 123,
   "title": "Song1",
    "artist": "Artist1",
   "album": “Album1”,
    "self": “http://localhost:8080/songs/123”
  },
 {
   "id": 456,
   "title": "Song2",
    "artist": "Artist2",
   "album": “Album2”,
    "self": “http://localhost:8080/songs/456”
  },
 {
   "id": 567,
   "title": "Song3",
    "artist": "Artist3",
   "album": “Album3”,
    "self": “http://localhost:8080/songs/567”
 }
 {
   "id": 890,
   "title": "Song4",
    "artist": "Artist4",
   "album": “Album4”,
    "self": “http://localhost:8080/songs/890”
 },
 {
   "id": 223,
   "title": "Song5",
    "artist": "Artist5",
   "album": “Album5”,
    "self": “http://localhost:8080/songs/223”
 }
 ]
}
```

Failure

Status: 404 Not Found

```
{
"Error":  "No songs exist"
}
```

View a Song

Unprotected Endpoint

Allows you to get an existing song.

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

Request

Headers

Accept - application/json

Path Parameters

Name	Description
song_id	ID of the song

Request Body

None

Response

Response Body Format

JSON or HTML depending on the value of the accept header.

Response Statuses

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

Response Examples

Success

```
Status: 200 OK

{
  "id": 123,
  "title": "Song1",
  "artist": "Artist1",
  "album": “Album1”,
  "self": “http://localhost:8080/songs/123”
}
```

Failure

```
Status: 404 Not Found

{
"Error":  "No song with this song_id exists"
}
```

Partially Edit a Song

Unprotected Endpoint

Allows you to edit a subset of the features for a specific song.

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

Request

Headers

Content-type - application/json

Accept - application/json

Path Parameters

Name	Description
song_id	ID of the song.

Request Body

Required

Request Body Format

JSON

Request JSON Attributes

Name	Description	Required?
title	The title of the song. Limited to 50 upper-case or lower-case alphabetic characters.	Yes
artist	The artist credited for this song. Limited to 50 characters.	Yes
album	The album this song is from. This can be an empty string if the song is not part of an album.	No

Request Body Example

```
{
  "album": “Album99”
}
```

Response

Response Body Format

JSON

Response Statuses

Outcome	Status Code	Notes
Success	200 OK	
Failure	400 Bad Request	If the request is missing the required attributes or does not contain any valid attributes, the song is not updated, and 400 status code is returned.
Failure	404 Not Found	No song with this song_id exists.

Response Examples

Success

```
Status: 200 OK

{
  "id": 123,
  "title": "Song1",
  "artist": "Artist1",
  "album": “Album99”,
  "self": “http://localhost:8080/songs/123”
}
```

Failure

```
Status: 400 Bad Request

{
"Error":  "The request object is missing a required attribute, or the value of an
attribute is invalid"
}
```

```
Status: 404 Not Found

{
"Error":  "No song with this song_id exists"
}
```

Edit a Song

Unprotected Endpoint

Allows you to edit the features for a specific song.

```
PUT /song/:song_id
```

Request

Headers

Content-type - application/json

Accept - application/json

Path Parameters

Name	Description
song_id	ID of the song.

Request Body

Required

Request Body Format

JSON

Request JSON Attributes

Name	Description	Required?
title	The title of the song. Limited to 50 upper-case or lower-case alphabetic characters.	Yes
artist	The artist credited for this song. Limited to 50 characters.	Yes
album	The album this song is from. This can be an empty string if the song is not part of an album.	No

Request Body Example

```
{
  "name": "Song1-2",
  "artist": “Artist1”,
  “album”: “Album99”,
}
```

Response

Headers

Location - The URL of the updated song.

Response Body Format

JSON

Response Statuses

Outcome	Status Code	Notes
Success	303 See Other	The URL of the updated song is returned in the Location header, but also in the response body.

Failure	400 Bad Request	If the request is missing the required attributes or does not contain any valid attributes, the song is not updated, and 400 status code is returned.
Failure	404 Not Found	No song with this song_id exists.

Response Examples

Success

```
Status: 200 OK

{
  "id": 123,
  "title": "Song1-2",
  "artist": "Artist1",
  "album": “Album99”,
  "self": “http://localhost:8080/songs/123”
}
```

Failure

```
Status: 400 Bad Request

{
"Error":  "The request object is missing a required attribute, or the value of an
attribute is invalid"
}
```

```
Status: 404 Not Found

{
"Error":  "No song with this song_id exists"
}
```

Delete a Song

Unprotected Endpoint

Allows you to delete a song.

```
DELETE /song/:song_id
```

Request

Headers

Accept - application/json

Path Parameters

Name	Description
song_id	ID of the song.

Request Body

None

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 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"
}
```