Portfolio – Winery Co-op

Patrick Stumbaugh
ONID: stumbaup

URL: https://winery-coop.wn.r.appspot.com
Login URL: https://winery-coop.wn.r.appspot.com/login

Data Model

The app stores entities in Datastore: Wineries and Wines.

Wineries

Property	Data Type	Notes
id	Integer	The id of the winery. Datastore will automatically generate this.
name	String	Name of the winery.
grapeType	String	Type of grapes the winery has. If the winery has multiple types, add it to the same string.
region	String	The region the winery is located in.
numVarieties	Integer	The number of varieties the winery can make. This number does not correlate to the number of wines that the winery produces.
owner	String	The owner of the winery. API will automatically generate this.
Wines	Object	These are the wines that have been associated with this winery. The API will generate this field if wines are added to the winery automatically.

Wines

Property	Data Type	Notes
id	Integer	The id of the wine. Datastore will automatically generate this.
Name	String	Name of the wine.
Type	String	Type of grapes the winery has. If the winery has multiple types, add it to the same string.
Year	String	The region the winery is located in.
price	Integer	The number of varieties the winery can make. This number does not correlate to the number of wines that the winery produces.
Owner	String	The owner of the winery. API will automatically generate this.
Producer	Object	This is the producer of the wine if a producer has been associated with it. The API will generate this field if a producer is added to the wine automatically.

The API is modeled in this way:

- A main user
- Wineries
    - Every call to this winery must be made by the user that owns the winery (the user that created it).
    - If a wine (or wines) is associated with a specific winery, a “wines” section will show which wines belong to this winery.
- Wines
    - Every call to a wine must be made by the user that owns the wine (the user that created it).
    - Wines may be added or deleted from a winery. When a wine is part of a winery and a GET request is made to that wine, the API will return a “producer” section showing which winery owns the wine.
- Each call to an entity that requires an owner authentication will have to be supplied a Bearer token with the owner’s id_token (Postman collection handles this in test suite).
- This API does not have endpoints with user id’s.
    - Only JWT’s (which are saved from id_token to postman environment variables for testing) are needed.
- Every non-user entity endpoint is protected (wines and wineries).

API Notes

- All responses from the API will come in JSON format.
- All requests that include body must be made with JSON format.
- Any request to an endpoint that is part of the API, but not allowed (ie requesting a delete to an endpoint that doesn't support this) will return an error message saying so.
- Both databases require a user to be logged in. The API will reject any requests to either wineries or wines if the user is not the owner of that winery or wine.
- ID tokens generated from the website and postman collection are to be only valid for 24 hours.
- While using the website, if you receive an error saying a new user login is forbidden, you must return to the home page and hit the logout button. Then you may try again. The website will assume you area logged in already if you click on the new user registration button.
- It is safe to assume any internal server error will result in status code 500.

For this project grading:

- There are 2 utility functions in the API. These are for development testing (aka – for postman tests for this project). These include an endpoint to delete all wineries and delete all wines. This was included so that one may adjust the postman collection if needed and be able to clear out the database. Many of the postman tests are reliant upon previous requests/tests succeeding, and if not starting with a clean database, they may yield failed tests later (i.e.: if a winery is added, and two wines are added to the winery, a following GET request will check for 2 wines in that winery. If the postman collection is altered, and the database not cleared, and the user checks that GET request again, there may be more wines in that winery (copies) and will yield failed test results (will see 4 wines instead of the 2 it is testing for), even, though the API is working correctly).
- A management token is included so that the grader may access all users info at the endpoint: /users. This management token will expire approximately 2 weeks after the class ends.
- Logging in to get an id_token requires username/password. Postman will handle two events for this.
- The Postman collection will login two users using username/password (Baz and Pat). Postman will save their respective id_tokens to jwt1 and jwt2 environment variables.
    - You are welcome to change the login information for these two requests. The two requests are near the top of the collection.

- If you are unable to login or have troubles logging in, you may use the following two users/passwords to login to get the id_token. These are also saved in the postman collection
    - username": baz@bop.com
    - password: Abc123456

    - username: pat@foo.com
    - password: Pat10001
        - Logging in and creating a new user via localhost works 100% of the time. For some reason, when using GCP, new user creation doesn't always work

Get all Wineries

List all the wineries for this user.

GET /wineries

Request

Path Parameters

None

Request Body

None

Response

Response Body Format

JSON

Response Statuses

Outcome	Status Code	Notes
Success	200 OK	
Failure	401 Unauthorized	Missing or invalid user id token
Failure	404 Not Found	Error retrieving list of wineries
Failure	406 Not Acceptable	The requested Accept type is not supported by this API

Response Examples

Success


```
Status: 200 OK
{
  "Total_Number_of_Wineries_in_Collection": 2,
  "wineries": [
    {
      "numVarieties": 1,
      "region": "Napa Valley",
      "grapeType": "White",
      "name": "Winery Z",
      "owner": "auth0|61a41df1af07b300718bdd12",
      "id": "6714551131176960",
      "self": "https://winery-coop.wn.r.appspot.com/wineries/6714551131176960"
    },
    {
      "name": "Winery F",
      "owner": "auth0|61a41df1af07b300718bdd12",
      "region": "Willamette Valley",
      "numVarieties": 3,
      "wines": [
        {
          "id": "4520678346719232",
          "name": "Wine A",
```

```
            "self": "https://winery-coop.wn.r.appspot.com/wines/4520678346719232"
          },
          {
            "name": "Wine I",
            "id": "4568304433758208",
            "self": "https://winery-coop.wn.r.appspot.com/wines/4568304433758208"
          }
        ],
        "grapeType": "Red",
        "id": "6746089881337856",
        "self": "https://winery-coop.wn.r.appspot.com/wineries/6746089881337856"
    }
  ]
}
```


Failure

Status: 401 Unauthorized


```
{
  "unauthorized": "Invalid or missing token trying to get all wineries from this owner."
}
```


Status: 404 Not Found


```
{
  "Error": " Could not find wineries from this owner."
}
```


Status: 406 Not Acceptable
Requested (Accept) content type not available


```
{
  "Error": " Requested (Accept) content type not available”
}
```

Get a Winery

Allows you to get an existing winery via the winery id

GET /wineries/:winery_id

Request

Path Parameters

Name	Description
winery_id	ID of the winery

Request Body

None

Response

Response Body Format

JSON

Response Statuses

Outcome	Status Code	Notes
Success	200 OK	
Failure	401 Unauthorized	Missing or invalid user id token
Failure	403 Forbidden	Invalid owner for this winery
Failure	404 Not Found	No winery with this winery_id found
Failure	406 Not Acceptable	The requested Accept type is not supported by this API

Response Examples

Success

```
Status: 200 OK
{
  “numVarieties”: 3,
  “owner”: “auth0|61a41df1af07b300718bdd12”,
  “wines”: [
    {
      “id”: “4802153323429888”,
      “name”: “Wine A”,
      “self”: “https://winery-coop.wn.r.appspot.com/wines/4802153323429888”
    },
    {
      “id”: “5685933206667264”,
      “name”: “Wine I”,
      “self”: “https://winery-coop.wn.r.appspot.com/wines/5685933206667264”
    }
  ],
  “region”: “Willamette Valley”,
  “grapeType”: “Red”,
  “name”: “Winery A”,
  “id”: “5064609648082944”,
```

```
    "self": "https://winery-coop.wn.r.appspot.com/wineries/5064609648082944"
}
```

Failure

Status: 401 Unauthorized

```
{
    "unauthorized": "Invalid or missing token trying to get all wineries from this owner."
}
```

Status: 403 Forbidden
Invalid winery owner. Only the winery's owner can get this winery

```
{
    "Error": "Invalid or missing token trying to get all wineries from this owner."
}
```

Status: 404 Not Found

```
{
    "Error": "Cannot find winery with this winery_id"
}
```

Status: 406 Not Acceptable
Requested (Accept) content type not available

```
{
    "Error": " Requested (Accept) content type not available"
}
```

Create a Winery

Allows you to create a new winery.

POST /wineries

Request

Path Parameters

None

Request Body

Required

Request must be made in JSON format

JSON

Request JSON Attributes

Name	Description	Required?
Name	The name of the winery.	Yes
grapeType	The type of the grapes this winery makes (if more than one, include them all in a single string. Format of that string up to the user).	Yes
region	The region this winery is in	Yes
numVarieties	The possible number of varieties the owner of the winery believes this winery can make.	Yes

***Validation NOTE**

- Name, grapeType and region must be less than 35 characters and sent as a string.
- numVarieties must be less than 10 numbers and sent as a positive number or 0

Request Body Example

```
{
   “name”: “Winery B”,
   “grapeType”: “Red”,
   “numVarieties”: 3,
   “region”: “Willamette Valley”
}
```

***NOTE**

- If extra attributes are added to the JSON request, they will not be added to the newly created winery. Only the required attributes above will be added. The winery will be created successfully, but extra attributes will be discarded without warning.

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 4 required attributes, the winery will not be created, and 400 status code will be returned.
Failure	401 Unauthorized	Missing or invalid user id token
Failure	415 Unsupported Media Type	If the request is made with content that is not JSON.

Response Examples

- The self attribute will contain the live link to the REST resource corresponding to this winery.

Success

```
Status: 201 Created
{ “id”: 6108145671733248 }
```

Failure

```
Status: 400 Bad Reqeust
{
   “Error”: [
      “The name must be a string between 1 and 35 characters”,
      “The grapeType must be a string between 1 and 35 characters”,
      “The region must be a string between 1 and 35 characters”,
      “The numVarieties must be a number between 1 and 10 characters”
   ]
}
```

```
Status: 401 Unauthorized
{
   “unauthorized”: “Invalid or missing token while creating a new winery.”
}
```

```
Status: 415 Unsupported Media Type
{
   “Error”: “Server only accepts application/json data.”
}
```

Delete a Winery

Allows you to delete a winery.

DELETE /wineries/:winery_id

Request

Path Parameters

Name	Description
winery_id	ID of the winery

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	403 Forbidden	If the request is made by an invalid owner
Failure	404 Not Found	No winery with this winery_id found

**** NOTE***

- If deleting a winery that includes wines, the wines will have their "producer" tag removed

Response Examples

Success

Status: 204 No Content

Failure

Status: 403 Forbidden
{
"Error": " Invalid winery owner. Only the winery's owner can delete this winery"
}

Status: 404 Not Found
{
"Error": "JWT valid, but could not find winery with this winery id:"
}

Update a Winery (PUT)

Allows you to update a winery, all attributes

PUT /wineries/:winery_id

Request

Path Parameters

Name	Description
winery_id	ID of the winery

Request Body

Required

Request must be made in JSON format

JSON

Request JSON Attributes

Name	Description	Required?
Name	The name of the winery.	Yes
grapeType	The type of the grapes this winery makes (if more than one, include them all in a single string. Format of that string up to the user).	Yes
region	The region this winery is in	Yes
numVarieties	The possible number of varieties the owner of the winery believes this winery can make.	Yes

Request Body Example

```
{
   “name”: “Winery B”,
   “grapeType”: “Red”,
   “numVarieties”: 3,
   “region”: “Willamette Valley”
}
```

***NOTE**

- If extra attributes are added to the JSON request, they will not be added to the winery. Only the required attributes above will be added. The winery will be updated successfully, but extra attributes will be discarded without warning.

Response Statuses

Outcome	Status Code	Notes
Success	204 No Content	
Failure	400 Bad Request	If the request is missing any of the 4 required attributes, the winery will not be created, and 400 status code will be returned.
Failure	401 Unauthorized	Missing or invalid user id token
Failure	403 Forbidden	If the request is made by an invalid owner
Failure	404 Not Found	No winery with this winery_id found

Failure	415 Unsupported Media Type	If the request is made with content that is not JSON.

Response Examples

Success

Status: 204 No Content

Failure

Status: 400 Bad Request
{
“Error”: “ Missing one or more attributes. For a PUT request, all attributes must be given.”
}

Status: 400 Bad Request (if bad attribute)
{
“Error”: “The numVarieties must be a number between 1 and 10 characters”
}

Status: 401 Unauthorized
{
“unauthorized”: “Invalid or missing token while creating a new winery.”
}

Status: 403 Forbidden
{
“Error”: “ Invalid winery owner. Only the winery’s owner can edit this winery”
}

Status: 404 Not Found
{
“Error”: “JWT valid, but could not find winery with this winery id:”
}

Status: 415 Unsupported Media Type
{
“Error”: “Server only accepts application/json data.”
}

Update a Winery (PATCH)

Allows you to update only certain attributes of a winery

PUT /wineries/:winery_id

Request

Path Parameters

Name	Description
winery_id	ID of the winery

Request Body

Required

Request must be made in JSON format

JSON

Request JSON Attributes

Name	Description	Required?
Name	The name of the winery.	No
grapeType	The type of the grapes this winery makes (if more than one, include them all in a single string. Format of that string up to the user).	No
region	The region this winery is in	No
numVarieties	The possible number of varieties the owner of the winery believes this winery can make.	No

Request Body Example

```
{
   “region”: “Willamette Valley”
}
```

***NOTE**

- If extra attributes are added to the JSON request, they will not be added to the winery. Only the required attributes above will be added. The winery will be updated successfully, but extra attributes will be discarded without warning.

Response Statuses

Outcome	Status Code	Notes
Success	204 No Content	
Failure	400 Bad Request	Something in the request isn’t correct
Failure	401 Unauthorized	Missing or invalid user id token
Failure	403 Forbidden	If the request is made by an invalid owner
Failure	404 Not Found	No winery with this winery_id found
Failure	415 Unsupported Media Type	If the request is made with content that is not JSON.

Response Examples

Success

Status: 204 No Content

Failure

Status: 400 Bad Request (if bad attribute)
{
"Error": "The numVarieties must be a number between 1 and 10 characters"
}

Status: 401 Unauthorized
{
"unauthorized": "Invalid or missing token while creating a new winery."
}

Status: 403 Forbidden
{
"Error": " Invalid winery owner. Only the winery's owner can edit this winery"
}

Status: 404 Not Found
{
"Error": "JWT valid, but could not find winery with this winery id:"
}

Status: 415 Unsupported Media Type
{
"Error": "Server only accepts application/json data."
}

Add a wine to a Winery (PUT)

Allows you to add a wine to a winery

PUT /wineries/:winery_id/wines/:wine_id

Request

Path Parameters

Name	Description
winery_id	ID of the winery
wine_id	ID of the wine

Request Body

None

Response Statuses

Outcome	Status Code	Notes
Success	204 No Content	
Failure	401 Unauthorized	Missing or invalid user id token
Failure	403 Forbidden	If the request is made by an invalid owner
Failure	404 Not Found	No winery with this winery_id found

Response Examples

Success

Status: 204 No Content

Failure

Status: 401 Unauthorized
{
“unauthorized”: “Invalid or missing token while creating a new winery.”
}

Status: 403 Forbidden
{
“Error”: “This wine is already assigned to a winery”
}

Status: 403 Forbidden (if different owner of winery than wine)
{
“Error”: “Invalid winery owner. Only the winery’s owner can add wines to this winery”
}

Status: 404 Not Found
{
“Error”: “The specified winery and/or wine does not exist”
}

Delete a wine from a Winery (DELETE)

Allows you to delete a wine from a winery

DELETE /wineries/:winery_id/wines/:wine_id

Request

Path Parameters

Name	Description
winery_id	ID of the winery
wine_id	ID of the wine

Request Body

None

Response Statuses

Outcome	Status Code	Notes
Success	204 No Content	
Failure	401 Unauthorized	Missing or invalid user id token
Failure	403 Forbidden	Invalid owner trying to delete wine
Failure	404 Not Found	No winery with this winery_id found

Response Examples

Success

Status: 204 No Content

Failure

Status: 401 Unauthorized
{
“unauthorized”: “Invalid or missing token while creating a new winery.”
}

Status: 403 Forbidden
{
“unauthorized”: “Invalid winery owner. Only the winery’s owner can delete wines from this winery”
}

Status: 404 Not Found (this wine is not associated with a winery)
{
“Error”: “ This wine_id doesn’t exist or is does not have a winery associated”
}

Status: 404 Not Found
{
“Error”: “The specified winery and/or wine does not exist”
}

Get all Wines

List all the wines for this user.

GET /wines

Request

Path Parameters

None

Request Body

None

Response

Response Body Format

JSON

Response Statuses

Outcome	Status Code	Notes
Success	200 OK	
Failure	401 Unauthorized	Missing or invalid user id token
Failure	404 Not Found	Error retrieving list of wines
Failure	406 Not Acceptable	The requested Accept type is not supported by this API

Response Examples

Success


```
Status: 200 OK
{
  “Total_Number_of_Wines_in_Collection”: 2,
  “wines”: [
    {
      “name”: “Wine I”,
      “type”: “Red”,
      “producer”: [
        {
          “name”: “Winery A”,
          “id”: “6397891782377472”,
          “self”: “https://winery-coop.wn.r.appspot.com/wineries/6397891782377472”
        }
      ],
      “owner”: “auth0|61a41df1af07b300718bdd12”,
      “year”: 2000,
      “price”: 12,
      “id”: “5057574760087552”,
      “self”: “https://winery-coop.wn.r.appspot.com/wines/5057574760087552”
    },
    {
      “owner”: “auth0|61a41df1af07b300718bdd12”,
      “price”: 50,
      “name”: “Wine A”,
```

```
        "type": "Merlot",
        "year": 2015,
        "id": "6464949643640832",
        "self": "https://winery-coop.wn.r.appspot.com/wines/6464949643640832"
      }
  ]
}
```


Failure

Status: 401 Unauthorized


```
{
  "unauthorized": "Invalid or missing token trying to get all wines from this owner."
}
```


Status: 404 Not Found


```
{
  "Error": "Could not find wines."
}
```


Status: 406 Not Acceptable
Requested (Accept) content type not available


```
{
  "Error": "Requested (Accept) content type not available"
}
```

Get a Wine

Allows you to get an existing wine via the wine id

GET /wines/:wine_id

Request

Path Parameters

Name	Description
wine_id	ID of the wine

Request Body

None

Response

Response Body Format

JSON

Response Statuses

Outcome	Status Code	Notes
Success	200 OK	
Failure	401 Unauthorized	Missing or invalid user id token
Failure	403 Forbidden	Unauthorized owner trying to access wine
Failure	404 Not Found	No wine with this wine_id found
Failure	406 Not Acceptable	The requested Accept type is not supported by this API

Response Examples

Success

```
Status: 200 OK
{
  "name": "Wine A",
  "price": 50,
  "owner": "auth0|61a41df1af07b300718bdd12",
  "type": "Merlot",
  "producer": [
    {
      "name": "Winery F",
      "id": "4990516898824192",
      "self": "https://winery-coop.wn.r.appspot.com/wineries/4990516898824192"
    }
  ],
  "year": 2015,
  "id": "5756301950844928",
  "self": "https://winery-coop.wn.r.appspot.com/wines/5756301950844928"
}
```

Failure

```
Status: 401 Unauthorized
{
```

"unauthorized": "Invalid or missing token trying to get all wines from this owner." }

Status: 403 Forbidden
{
"Error": "Invalid wine owner. Only the wine's owner can get this wine"
}

Status: 404 Not Found
{
"Error": " Could not find wine from this owner."
}

Status: 406 Not Acceptable
Requested (Accept) content type not available
{
"Error": " Requested (Accept) content type not available”
}

Create a Wine

Allows you to create a new wine.

POST /wines

Request

Path Parameters

None

Request Body

Required

Request must be made in JSON format

JSON

Request JSON Attributes

Name	Description	Required?
name	The name of the wine.	Yes
type	The type of wine.	Yes
year	The year the wine was bottled.	Yes
price	The price of the wine.	Yes

***Validation NOTE**

- name and type must be less than 35 characters and sent as a string.
- price must be less than 10 numbers and sent as a positive number or 0.
- year must be a 4 digit number less than 3000 (this future number assumes an owner may be planning a wine in future years).

Request Body Example

```
{
   "name": "Wine I",
   "type": "Red",
   "year": 2000,
   "price": 12
}
```

***NOTE**

- If extra attributes are added to the JSON request, they will not be added to the newly created wine. Only the required attributes above will be added. The wine will be created successfully, but extra attributes will be discarded without warning.

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 4 required attributes, the wine will not be created, and 400 status code will be returned.
Failure	401 Unauthorized	Missing or invalid user id token
Failure	415 Unsupported Media Type	If the request is made with content that is not JSON.

Response Examples

- The self attribute will contain the live link to the REST resource corresponding to this wine.

Success

```
Status: 201 Created
{ "id": 6108145671733248 }
```

Failure

```
Status: 400 Bad Reqeust
{
   "Error": [
      "The year must be a 4 digit number",
      "The price must be a number between 1 and 10 characters"
   ]
}
```

```
Status: 401 Unauthorized
{
   "unauthorized": "Invalid or missing token while creating a new wine."
}
```

```
Status: 415 Unsupported Media Type
{
   "Error": "Server only accepts application/json data."
}
```

Delete a Wine

Allows you to delete a wine.

DELETE /wines/:wine_id

Request

Path Parameters

Name	Description
wine_id	ID of the wine

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	403 Forbidden	If the request is made by an invalid owner
Failure	404 Not Found	No wine with this wine_id found

**** NOTE***

- If deleting a wine that includes a producer, the producer will have their “wines” tag removed for this wine.

Response Examples

Success

Status: 204 No Content

Failure

Status: 403 Forbidden
{
"Error": " Invalid wine owner. Only the wine’s owner can delete this wine”
}

Status: 404 Not Found
{
"Error": "JWT valid, but could not find wine with this wine id:”
}

Update a Wine (PUT)

Allows you to update a wine, all attributes

PUT /wines/:wine_id

Request

Path Parameters

Name	Description
wine_id	ID of the wine

Request Body

Required

Request must be made in JSON format

JSON

Request JSON Attributes

Name	Description	Required?
name	The name of the wine.	Yes
type	The type of wine.	Yes
year	The year the wine was bottled.	Yes
price	The price of the wine.	Yes

Request Body Example

```
{
   "name": "Wine I",
   "type": "Red",
   "year": 2000,
   "price": 12
}
```

***NOTE**

- If extra attributes are added to the JSON request, they will not be added to the wine. Only the required attributes above will be added. The wine will be updated successfully, but extra attributes will be discarded without warning.

Response Statuses

Outcome	Status Code	Notes
Success	204 No Content	
Failure	400 Bad Request	If the request is missing any of the 4 required attributes, the wine will not be created, and 400 status code will be returned.
Failure	401 Unauthorized	Missing or invalid user id token
Failure	403 Forbidden	If the request is made by an invalid owner
Failure	404 Not Found	No wine with this wine_id found
Failure	415 Unsupported Media Type	If the request is made with content that is not JSON.

Response Examples

Success

Status: 204 No Content

Failure

Status: 400 Bad Request
{
"Error": " Missing one or more attributes. For a PUT request, all attributes must be given.”
}

Status: 400 Bad Request (if bad attribute)
{
"Error": "The price must be a number between 1 and 10 characters"
}

Status: 401 Unauthorized
{
"unauthorized": "Invalid or missing token while creating a new wine."
}

Status: 403 Forbidden
{
"Error": "Invalid wine owner. Only the wine’s owner can delete this wine”
}

Status: 404 Not Found
{
"Error": "No wine with this wine_id found”
}

Status: 415 Unsupported Media Type
{
"Error": "Server only accepts application/json data."
}

Update a Wine (PATCH)

Allows you to update only certain attributes of a wine

PUT /wines/:wine_id

Request

Path Parameters

Name	Description
wine_id	ID of the wine

Request Body

Required

Request must be made in JSON format

JSON

Request JSON Attributes

Name	Description	Required?
name	The name of the wine.	No
type	The type of wine.	No
year	The year the wine was bottled.	No
price	The price of the wine.	No

Request Body Example

```
{
   "price": 49
}
```

***NOTE**

- If extra attributes are added to the JSON request, they will not be added to the wine. Only the required attributes above will be added. The wine will be updated successfully, but extra attributes will be discarded without warning.

Response Statuses

Outcome	Status Code	Notes
Success	204 No Content	
Failure	400 Bad Request	Something in the request doesn't work
Failure	401 Unauthorized	Missing or invalid user id token
Failure	403 Forbidden	If the request is made by an invalid owner
Failure	404 Not Found	No wine with this wine_id found
Failure	415 Unsupported Media Type	If the request is made with content that is not JSON.

Response Examples

Success

Status: 204 No Content

Failure

Status: 400 Bad Request (if bad attribute)

{
"Error": "The price must be a number between 1 and 10 characters"
}

Status: 401 Unauthorized
{
"unauthorized": "Invalid or missing token while creating a new wine."
}

Status: 403 Forbidden
{
"Error": " Invalid wine owner. Only the wine’s owner can delete this wine”
}

Status: 404 Not Found
{
"Error": “No wine with this wine_id found”
}

Status: 415 Unsupported Media Type
{
"Error": "Server only accepts application/json data."
}

Invalid URL requests

If a request is made to an invalid URL (ie: DELETE to wineries/), then an appropriate response will be returned.

Response Statuses

Outcome	Status Code	Notes
Failure	405 Method Not Allowed	If a request for a method is not allowed on a specific URL, return error code

Response Examples

Failure (this list is not exhaustive)

```
Status: 405 Method Not Allowed
{
   “Error”: “ Invalid or missing token while trying to delete a winery.”
}
```

See all users

Allows the user to see all users. Primarily for use in development

Response Statuses

Outcome	Status Code	Notes
Success	200 OK	
Failure	400 Bad Request	

Response Examples

Success


```
Status: 200 OK
[
  {
    "created_at": "2021-11-29T00:25:53.196Z",
    "email": "pat@foo.com",
    "email_verified": false,
    "identities": [
      {
        "user_id": "61a41e1193d5400069c83f46",
        "provider": "auth0",
        "connection": "Username-Password-Authentication",
        "isSocial": false
      }
    ],
    "name": "pat@foo.com",
    "nickname": "pat",
    "picture":
"https://s.gravatar.com/avatar/4b3a4d35cf04bdaa6f808b04a716a87c?s=480&r=pg&d=https
%3A%2F%2Fcdn.auth0.com%2Favatars%2Fpa.png",
    "updated_at": "2021-12-06T05:22:31.663Z",
    "user_id": "auth0|61a41e1193d5400069c83f46",
    "last_login": "2021-12-06T05:22:31.663Z",
    "last_ip": "50.39.224.193",
    "logins_count": 41
  },
  {
    "created_at": "2021-11-29T00:25:21.207Z",
    "email": "baz@bop.com",
    "email_verified": false,
    "identities": [
      {
        "user_id": "61a41df1af07b300718bdd12",
```

```
            "provider": "auth0",
            "connection": "Username-Password-Authentication",
            "isSocial": false
          }
        ],
        "name": "baz@bop.com",
        "nickname": "baz",
        "picture":
"https://s.gravatar.com/avatar/2effce092193fae7ef8cd32cd1519b12?s=480&r=pg&d=https%
3A%2F%2Fcdn.auth0.com%2Favatars%2Fba.png",
        "updated_at": "2021-12-06T05:22:31.247Z",
        "user_id": "auth0|61a41df1af07b300718bdd12",
        "last_login": "2021-12-06T05:22:31.246Z",
        "last_ip": "50.39.224.193",
        "logins_count": 64
    },
    {
        "created_at": "2021-11-29T00:17:10.293Z",
        "email": "stumbaup@oregonstate.edu",
        "email_verified": true,
        "family_name": "Stumbaugh",
        "given_name": "Patrick",
        "identities": [
          {
            "provider": "google-oauth2",
            "access_token": "ya29.a0ARrdaM-
43P0KVbKhOPdhS8aRkOoETAWlUTBDU4IQlYxNPAwlvWsuvFj8ykvwc6d-Oxidc-
KYbAhPGCqkGf0rVpRLzFhgcu2iIwTqBeN2IwWWn7wresEXX5KxcQR5GWpOvLbfiHcFYskFVIXjf
WCZQAJgPdn9kd4",
            "expires_in": 3599,
            "user_id": "102203828062514312991",
            "connection": "google-oauth2",
            "isSocial": true
          }
        ],
        "locale": "en",
        "name": "Patrick Stumbaugh",
        "nickname": "stumbaup",
        "picture": "https://lh3.googleusercontent.com/a-
/AOh14GiVeKh072txaKIRyhh4ZUArFLaip2fyvr8cuecX=s96-c",
        "updated_at": "2021-12-05T21:44:49.818Z",
        "user_id": "google-oauth2|102203828062514312991",
        "last_login": "2021-12-05T21:44:49.818Z",
        "last_ip": "50.39.224.193",
```

```
      "logins_count": 4
   }
]
```

Utility function -> Delete all Wines: For development and testing (grading) purposes only

Allows you to delete all wines.

DELETE /wines/deleteAll

Request

Path Parameters

No parameters

Request Body

None

Response

No body

Response Body Format

Success: No body
Failure: JSON

Response Statuses

Outcome	**Status Code**	**Notes**
Success	200 Success	

Response Examples

Success

Status: 200 Success “all wines deleted”

Utility function -> Delete all Wineries: For development and testing (grading) purposes only

Allows you to delete all wineries.

DELETE /wineries/deleteAll

Request

Path Parameters

No parameters

Request Body

None

Response

No body

Response Body Format

Success: No body
Failure: JSON

Response Statuses

Outcome	**Status Code**	**Notes**
Success	200 Success	

Response Examples

Success

Status: 200 Success “all wineries deleted”