Amy Salley

Cloud Application Development

29 May 2022

Oregon State University

Deployed using Google Cloud App Engine

Data Model

The app stores three kinds of entities in Datastore: Users, Boats, and Loads.

Users

Property	Data Type	Notes	Required
name	String	Email address the user uses to log in.	yes
unique_id	String	User “sub” value from the JWT. This is the unique identifier for a user.	yes

A user can be the “owner” (“owner” is an attribute of a boat entity) of zero to many boats.

Boats

Property	Data Type	Notes	Required
id	Integer	The id of the boat. Datastore automatically generates it.	yes
name	String	Name of the boat.	yes
type	String	Type of the boat. E.g., Sailboat, Catamaran, etc.	yes
length	Integer	The length of the boat in feet.	yes
loads	List of integers	A list of the ids of the loads the boat is carrying. Empty list if the boat is not carrying any loads.	yes, but can be an empty list
owner	String	The unique_id of the user who is the owner of the boat.	yes

A boat is owned by exactly one owner/user, indicated in the “owner” attribute.

A boat may carry zero to many loads, indicated in the “loads” attribute.

Loads

Property	Data Type	Notes	Required
id	Integer	The id of the load. Datastore automatically generates it.	yes
volume	Integer	The volume of the load.	yes
carrier	Integer	If the load is on a boat, contains the id for that boat. Otherwise null.	no
item	String	Description of the load.	yes
creation_date	String	Date the load was created.	yes

A load may be loaded on zero or one boat, indicated in the “carrier” attribute.

View all Users

List all the users.

Unprotected endpoint.

GET /users

Request

Path Parameters

None

Request Body

None

Response

Response Body Format

JSON

Response Statuses

Outcome	Status Code	Notes
Success	200 OK	Returns all users

Response Examples

Success

Status: 200 OK

```
[
{
  "name": “bob@email.com”,
  “unique_id": "auth0|628bc76958307f00681d7a49"
},
{
  "name": “kim@email.com”,
  “unique_id": "auth0|627d9b129d042c006938cd09"
}
]
```

Create a Boat

Allows you to create a new boat. The boat will be owned by the user who is authenticated by the JWT supplied in the request. Requires a valid JWT corresponding to the user.

Protected endpoint.

POST /boats

Request

Path Parameters

None

Request Headers

Content-type: application/json

Accept: application/json

Authorization: Bearer Token

Request Body

Required

Request Body Format

JSON

Request JSON Attributes

Name	Description	Required?
name	The name of the boat.	Yes
type	The type of the boat. E.g., Sailboat, Catamaran, etc.	Yes
length	Length of the boat in feet.	Yes

Request Body Example

```
{
  "name": "Sea Witch",
  "type": "Catamaran",
  "length": 28
}
```

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 boat is not created, and 400 status code is returned.
Failure	401Unauthorized	Authorization header is missing, Invalid header, Token is expired, Incorrect claims, Unable to parse authentication token, or No RSA key in JWKs
Failure	406 Not Acceptable	The Accept header in the request must be application/json.

Response Examples

- Datastore will automatically generate an ID and store it with the entity being created. This value is sent back in the response body as shown in the example.
- The owner attribute is the unique_id of the user who owns the boat. This value is obtained from the "sub" value from the JWT of the current user and is stored in Datastore.
- The loads attribute is a list of ids of loads the boat is carrying. This list will be empty when the boat is first created. The loads attribute is stored in Datastore.
- The self attribute is a link to the boat resource, and is generated dynamically from information from the request.

Success

Status: 201 Created

```
{
  "id": 123,
  "name": "Sea Witch",
  "type": "Catamaran",
  "length": 28,
  "owner": "auth0|628bc76958307f00681d7a49",
  "loads": [],
  "self": "http://127.0.0.1:8080/boats/123"

}
```

Failure

Status: 400 Bad Request

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

Status: 401 Unauthorized

```
{
“code”: “no auth header”,
“description”: “Authorization header is missing”
}
```

Status: 406 Not Acceptable

```
{
"Error": “Client must accept JSON response”
}
```

View all Boats

List all the boats owned by the user who is authenticated by the JWT supplied in the request. Requires a valid JWT corresponding to the user.

Protected endpoint.

GET /boats

Request

Path Parameters

None

Request Headers

Accept: application/json

Authorization: Bearer Token

Request Body

None

Response

Response Body Format

JSON

Response Statuses

Outcome	Status Code	Notes
Success	200 OK	Returns 5 boats per page and a link to the next 5 boats. Results implement pagination using offsets and limits.
Failure	401 Unauthorized	Invalid or missing JWT, Authorization header is missing, Invalid header, Token is expired, Incorrect claims, Unable to parse authentication token, or No RSA key in JWKs
Failure	406 Not Acceptable	The Accept header in the request must be application/json.

Response Examples

- Boat entity is related to a user. Shows only those boats in the collection which are related to the user corresponding to the valid JWT provided in the request.

Success

Status: 200 OK


```
{ “boats”:[
{
  "id": 123,
  "name": "Sea Witch",
  "type": "Catamaran",
  "length": 28,
  “owner”: “auth0|628bc76958307f00681d7a49”,
  “loads”: [],
  “self”: “http://127.0.0.1:8080/boats/123”
},
{
  "id": 456,
  "name": "Adventure",
  "type": "Sailboat",
  "length": 50,
  “owner”: “auth0|628bc76958307f00681d7a49”,
  “loads”: [321],
  “self”: “http://127.0.0.1:8080/boats/456”
},
{
  "id": 789,
  "name": "Hocus Pocus",
  "type": "Sailboat",
  "length": 100,
  “owner”: “auth0|628bc76958307f00681d7a49”,
  “loads”: [],
  “self”: “http://127.0.0.1:8080/boats/789”
},
{
  "id": 234,
  "name": “Adventure 2",
  "type": "Sailboat",
  "length": 50,
  “owner”: “auth0|628bc76958307f00681d7a49”,
  “loads”: [654, 987],
  “self”: “http://127.0.0.1:8080/boats/234”
},
{
  "id": 345,
  "name": "Toot",
  "type": “Tug boat",
  "length": 34,
  “owner”: “auth0|628bc76958307f00681d7a49”,
  “loads”: [],
  “self”: “http://127.0.0.1:8080/boats/345”
} ],
  ”total_items”: 7,
  ”next”: http://127.0.0.1:8080/boats?limit=5&offset=5”
}
```

Failure

Status: 401 Unauthorized

```
{
“code”: “no auth header”,
“description”: “Authorization header is missing”
}
```

Status: 406 Not Acceptable

```
{
"Error": “Client must accept JSON response”
}
```

View a Boat

Allows you to view an existing boat. Requires a valid JWT corresponding to the current user.

Protected endpoint.

GET /boats/:boat_id

Request

- Requires a valid JWT corresponding to the current user/boat owner to view the boat.

Path Parameters

Name	Description
boat_id	ID of the boat

Request Headers

Accept: application/json

Authorization: Bearer Token

Request Body

None

Response

Response Body Format

JSON

Response Statuses

Outcome	Status Code	Notes
Success	200 OK	
Failure	401 Unauthorized	Authorization header is missing, Invalid header, Token is expired, Incorrect claims, Unable to parse authentication token, or No RSA key in JWKs
Failure	403 Forbidden	Only the boat owner may view this boat.
Failure	404 Not Found	No boat with this boat_id exists.
Failure	406 Not Acceptable	The Accept header in the request must be application/json.

Response Examples

Success

Status: 200 OK

```
{
  "id": 123,
  "name": "Sea Witch",
  "type": "Catamaran",
  "length": 28,
  “owner”: “auth0|628bc76958307f00681d7a49”,
  “loads”: [],
  “self”: “http://127.0.0.1:8080/boats/123”
}
```

Failure

Status: 401 Unauthorized

```
{
“code”: “no auth header”,
“description”: “Authorization header is missing”
}
```

Status: 403 Forbidden

```
{
"Error": “Only the boat owner may view this boat”
}
```

Status: 404 Not Found

```
{
"Error": "No boat with this boat_id exists"
}
```

Status: 406 Not Acceptable

```
{
"Error": “Client must accept JSON response”
}
```

Edit a Boat

Allows you to replace all attributes of a boat. Requires a valid JWT corresponding to the current user/owner of the boat.

Protected endpoint.

```
PUT /boats/:boat_id
```

Request

- Requires a valid JWT corresponding to the current user/boat owner to edit the boat.

Path Parameters

Name	Description
boat_id	ID of the boat

Request Headers

Content-type: application/json

Accept: application/json

Authorization: Bearer Token

Request Body

Required

Request Body Format

JSON

Request JSON Attributes

Name	Description	Required?
name	The name of the boat.	Yes
type	The type of the boat. E.g., Sailboat, Catamaran, etc.	Yes
length	Length of the boat in feet.	Yes

Request Body Example

```
{
  “name”: “Troll”
  "type": “Tiny Yacht”,
  "length": 1
}
```

Response

Response Headers

Content-Type: application/json

Response Body Format

JSON

Response Statuses

Outcome	Status Code	Notes
Success	200 OK	The boat was edited successfully.
Failure	400 Bad Request	If the request is missing one of the 3 required attributes, the boat is not edited, and 400 status code is returned.
Failure	401 Unauthorized	Authorization header is missing, Invalid header, Token is expired, Incorrect claims, Unable to parse authentication token, or No RSA key in JWKs
Failure	403 Forbidden	Only the boat owner may edit this boat.
Failure	404 Not Found	No boat with this boat_id exists.
Failure	406 Not Acceptable	The Accept header in the request must be application/json.

Response Examples

- Requires a valid JWT corresponding to the current user/boat owner to edit the boat.
- The self attribute is a link to the boat resource, and is generated dynamically from information in the request. The self attribute is not stored in Datastore.
- Id and owner attributes may not be modified.
- Sets loads attribute to empty list.
- Any extraneous attributes included by the user will be ignored.

Success

Status: 200 OK


```
{
  "id": 123,
  "name": “Troll”,
  "type": “Tiny Yacht”,
  "length": 1,
  “loads”: [],
  “owner”: “auth0|628bc76958307f00681d7a49”,
  “self”: “http://127.0.0.1:8080/123”
}
```


Failure

Status: 400 Bad Request


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


Status: 401 Unauthorized


```
{
“code”: “no auth header”,
“description”: “Authorization header is missing”
}
```


Status: 403 Forbidden


```
{
"Error": “Only the boat owner may edit this boat”
}
```


Status: 404 Not Found


```
{
"Error": "No boat with this boat_id exists”
}
```


Status: 406 Not Acceptable


```
{
"Error": “Client must accept JSON response”
}
```

Edit a Boat

Allows you to edit individual attributes of a boat. Requires a valid JWT corresponding to the current user/owner of the boat.

Protected endpoint.

```
PATCH /boats/:boat_id
```

Request

- Requires a valid JWT corresponding to the current user/boat owner to edit the boat.

Path Parameters

Name	Description
boat_id	ID of the boat

Request Headers

Content-type: application/json

Accept: application/json

Authorization: Bearer Token

Request Body

Required

Request Body Format

JSON

Request JSON Attributes

Name	Description	Required?
name	The name of the boat.	No
type	The type of the boat. E.g., Sailboat, Catamaran, etc.	No
length	Length of the boat in feet.	No

Request Body Example

```
{
  "type": “Mega Yacht”,
  "length": 13
}
```

Response

Response Headers

Content-Type: application/json

Response Body Format

JSON

Response Statuses

Outcome	Status Code	Notes
Success	200 OK	
Failure	401 Unauthorized	Authorization header is missing, Invalid header, Token is expired, Incorrect claims, Unable to parse authentication token, or No RSA key in JWKs
Failure	404 Not Found	No boat with this boat_id exists.
Failure	403 Forbidden	Only the boat owner may edit this boat.
Failure	406 Not Acceptable	The Accept header in the request must be application/json.

Response Examples

- Requires a valid JWT corresponding to the current user/boat owner to edit the boat.

- Id and owner attributes may not be modified.

- The self attribute is a link to the boat resource, and is generated dynamically from information in the request. The self attribute is not stored in Datastore.

- Any attributes not modified in the request body will remain the same.

- Any extraneous attributes included by the user will be ignored.

Success

Status: 200 OK


```
{
  "id": 123,
  "name": "Sea Witch",
  "type": “Mega Yacht”,
  "length": 13,
  “loads”: [],
  “owner”: “auth0|628bc76958307f00681d7a49”,
  “self”: “http://127.0.0.1:8080/123”
}
```

Failure

Status: 401 Unauthorized

```
{
“code”: “no auth header”,
“description”: “Authorization header is missing”
}
```

Status: 403 Forbidden

```
{
"Error": “Only the boat owner may edit this boat”
}
```

Status: 404 Not Found

```
{
"Error": "No boat with this boat_id exists”
}
```

Status: 406 Not Acceptable

```
{
"Error": “Client must accept JSON response”
}
```

Delete a Boat

Allows you to delete a boat. Note that if the boat currently has a load, deleting the boat will unload any loads on the boat. Requires a valid JWT corresponding to the current user/owner of the boat.

Protected endpoint.

```
DELETE /boats/:boat_id
```

Request

- Requires a valid JWT corresponding to the current user/boat owner to delete the boat.

Path Parameters

Name	Description
boat_id	ID of the boat

Request Headers

Authorization: Bearer Token

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	401 Unauthorized	Authorization header is missing, Invalid header, Token is expired, Incorrect claims, Unable to parse authentication token, or No RSA key in JWKs
Failure	403 Forbidden	Only the boat owner may delete this boat
Failure	404 Not Found	No boat with this boat_id exists

Response Examples

- Requires a valid JWT corresponding to the current user/boat owner to delete the boat.

Success

Status: 204 No Content

Failure

Status: 401 Unauthorized

```
{
“code”: “no auth header”,
“description”: “Authorization header is missing”
}
```

Status: 403 Forbidden

```
{
"Error": “Only the boat owner may delete this boat”
}
```

Status: 404 Not Found

```
{
"Error": "No boat with this boat_id exists"
}
```

Create a Load

Allows you to create a new load.

Unprotected endpoint.

POST /loads

Request

Path Parameters

None

Request Headers

Content-type: application/json

Accept: application/json

Request Body

Required

Request Body Format

JSON

Request JSON Attributes

Name	Description	Required?
volume	The volume of the load	Yes
item	Description of the load	Yes
creation_date	Date the load was created	Yes

Request Body Example

```
{
  “volume”: 5,
  “item”: “LEGO Blocks”,
  “creation_date”: “10/18/2021”
}
```

Response

Response Body Format

JSON

Response Statuses

Outcome	Status Code	Notes
Success	201 Created	
Failure	400 Bad Request	If the request is missing an attribute, the load is not created, and 400 status code is returned.
Failure	406 Not Acceptable	The Accept header in the request must be application/json.

Response Examples

- Datastore will automatically generate an ID and store it with the entity being created. This value is sent in the response body as shown in the example.
- The value of the carrier attribute is the boat the load is assigned to. All newly created loads are not assigned to a boat, so the carrier value will be null.
- The self attribute is a link to the load resource, and is generated dynamically from information form the request.

Success

Status: 201 Created

```
{
  "id": 123,
  “volume”: 5,
  "carrier”: null,
  “item”: “LEGO Blocks”,
  “creation_date”: “10/18/2021”
  “self”: “http://127.0.0.1:8080/loads/123”
}
```

Failure

Status: 400 Bad Request

```
{
"Error": "The request object is missing the required number"
}
```

Status: 406 Not Acceptable

```
{
"Error": “Client must accept JSON response”
}
```

View all Loads

List all the loads.

Unprotected endpoint.

GET /loads

Request

Path Parameters

None

Request Headers

Accept: application/json

Request Body

None

Response

Response Body Format

JSON

Response Statuses

Outcome	Status Code	Notes
Success	200 OK	Returns 5 loads per page and a link to the next 5 loads. Results implement pagination using offsets and limits.
Failure	406 Not Acceptable	The Accept header in the request must be application/json.

Response Examples

Success

Status: 200 OK


```
{ “loads”:
[
{
  "id": 123,
  “volume”: 5,
  "carrier”: null,
  “item”: “LEGO Blocks”,
  “creation_date”: “10/18/2021”
  “self”: “http://127.0.0.1:8080/loads/123”
},
{
  "id": 789,
  “volume”: 10,
  "carrier”: 321,
  “item”: “Stuffed Animals”,
  “creation_date”: “10/19/2021”
  “self”: “http://127.0.0.1:8080/loads/789”
},
{
  "id": 456,
  “volume”: 3,
  "carrier”: null,
  “item”: “Food”,
  “creation_date”: “10/19/2021”
  “self”: “http://127.0.0.1:8080/loads/456”
},
{
  "id": 246,
  “volume”: 3,
  "carrier”: null,
  “item”: “Rats”,
  “creation_date”: “10/19/2021”
  “self”: “http://127.0.0.1:8080/loads/246”
},
{
  "id": 579,
  “volume”: 3,
  "carrier”: 876,
  “item”: “Furniture”,
  “creation_date”: “10/19/2021”
  “self”: “http://127.0.0.1:8080/loads/579”
}
],
 “total_items”: 7,
 “next”: “http://127.0.0.1:8080/loads?limit=3&offest=3”
}
```

Failure

Status: 406 Not Acceptable


```
{
"Error": “Client must accept JSON response”
}
```

View a Load

Allows you to get an existing load.

Unprotected endpoint.

GET /loads/:load_id

Request

Path Parameters

Name	Description
load_id	ID of the load

Request Headers

Accept: application/json

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
Failure	406 Not Acceptable	The Accept header in the request must be application/json.

Response Examples

Success

Status: 200 OK

```
{
  "id": 123,
  “volume”: 5,
  "carrier”: null,
  “item”: “LEGO Blocks”,
  “creation_date”: “10/18/2021”
  “self”: “http://127.0.0.1:8080/loads/123”
}
```

Failure

Status: 404 Not Found

```
{
"Error": "No load with this load_id exists"
}
```

Status: 406 Not Acceptable

```
{
"Error": “Client must accept JSON response”
}
```

Edit a Load

Allows you to edit all attributes of a load.

Unprotected endpoint.

PUT /loads/:load_id

Request

Path Parameters

Name	Description
load_id	ID of the load

Request Headers

Content-type: application/json

Accept: application/json

Request Body

Required

Request Body Format

JSON

Request JSON Attributes

Name	Description	Required?
item	Description of the load.	Yes
volume	Volume of the load.	Yes
creation_date	Date the load was created.	Yes

Request Body Example

```
{
  “volume”: 15,
  “item”: “LEGO Blocks”,
  “creation_date”: “10/18/2021”
}
```

Response

Response Headers

Content-Type: application/json

Response Body Format

JSON

Response Statuses

Outcome	Status Code	Notes
Success	200 OK	The load was edited successfully.
Failure	400 Bad Request	If the request is missing at least one of the 3 required attributes, the load is not edited, and 400 status code is returned.
Failure	404 Not Found	No load with this load_id exists.
Failure	406 Not Acceptable	The Accept header in the request must be application/json.

Response Examples

- The self attribute is a link to the load resource, and is generated dynamically from information in the request. The self attribute is not stored in Datastore.

- The carrier attribute will be reset to None.

- Any extraneous attributes included by the user will be ignored.

Success

Status: 200 OK

```
{
  "id": 123,
  "volume": 15,
  "carrier": null,
  "item": "LEGO Blocks",
  "creation_date": "10/18/2021"
  "self": "http://127.0.0.1:8080/loads/123"
}
```

Failure

Status: 400 Bad Request

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

Status: 404 Not Found

```
{
"Error": "No load with this load_id exists”
}
```

Status: 406 Not Acceptable

```
{
"Error": “Client must accept JSON response”
}
```

Edit a Load

Allows you to edit individual attributes of a load.

Unprotected endpoint.

PATCH /loads/:load_id

Request

Path Parameters

Name	Description
load_id	ID of the load

Request Headers

Content-type: application/json

Accept: application/json

Request Body

Required

Request Body Format

JSON

Request JSON Attributes

(At least one attribute is required)

Name	Description	Required?
item	Description of the load.	No
volume	Volume of the load.	No
creation_date	Date the load was created.	No

Request Body Example

```
{
   "creation_date": "1/18/2022"
}
```

Response

Response Headers

Content-Type: application/json

Response Body Format

JSON

Response Statuses

Outcome	Status Code	Notes
Success	200 OK	
Failure	404 Not Found	No load with this load_id exists.
Failure	406 Not Acceptable	The Accept header in the request must be application/json.

Response Examples

- The self attribute is a link to the boat resource, and is generated dynamically from information in the request. The self attribute is not stored in Datastore.
- Any attributes not included in the request body will remain unchanged.
- Any extraneous attributes included by the user will be ignored.

Success

Status: 200 OK

```
{
  "id": 123,
  "volume": 15,
  "carrier": null,
  "item": "LEGO Blocks",
  "creation_date": "1/18/2022"
  "self": "http://127.0.0.1:8080/loads/123"
}
```

Failure

Status: 404 Not Found

```
{
"Error":  "No load with this load_id exists"
}
```

Status: 406 Not Acceptable

```
{
"Error":  "Client must accept JSON response"
}
```

Delete a Load

Allows you to delete a load. If the load being deleted is on a boat, the load is removed from the list of loads on that boat.

Unprotected endpoint.

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

Request

Path Parameters

Name	Description
load_id	ID of the load

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 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"
}
```

Assign a Load to a Boat

A load is loaded on a boat.

Unprotected endpoint.

PUT /boats/:boat_id/loads/:load_id

Request

Path Parameters

Name	Description
boat_id	ID of the boat
load_id	ID of the load

Request Body

None

Response

No body

Response Body Format

Success: No body

Failure: JSON

Response Statuses

Outcome	Status Code	Notes
Success	204 No Content	Succeeds only if a boat exists with this boat_id, a load exist with this load_id and the load is not already assigned to a boat.
Failure	403 Forbidden	The load is already loaded on another boat.
Failure	404 Not Found	The specified boat and/or load does not exist

Response Examples

Success

Status: 204 No Content

Failure

Status: 403 Forbidden

```
{
"Error": "The load is already loaded on another boat”
}
```

Status: 404 Not Found

```
{
"Error": "The specified boat and/or load does not exist"
}
```

Comment

- A load can be assigned to only one boat.

Remove a Load from a Boat

Load is no longer on the boat.

Unprotected endpoint.

```
DELETE /boats/:boat_id/loads/:load_id
```

Request

Path Parameters

Name	Description
boat_id	ID of the boat
load_id	ID of the load

Request Body

None

Response

No body

Response Body Format

Success: No body

Failure: JSON

Response Statuses

Outcome	Status Code	Notes
Success	204 No Content	Succeeds only if a boat exists with this boat_id, a load exist with this load_id and this load is on this boat.
Failure	404 Not Found	No boat with this boat_id is loaded with the load with this load_id. This could be because no boat with this boat_id exists, or because no load with load_id exists, or even if bo boat_id and load_id are valid, the boat with this boat_id is n loaded with the load with this load_id

Response Examples

Success

```
Status: 204 No Content
```

Failure

```
Status: 404 Not Found
{
"Error": " No boat with this boat_id is loaded with the load with this load_id”
}
```