Portfolio Assignment: API Spec

CS 493: Cloud Application Development

Fall 2022

Oregon State University

Deployed URL: https://final-370717.uc.r.appspot.com

Account/Creation URL: https://final-370717.uc.r.appspot.com

Data Model

The app stores two kinds of entities in Datastore, Boats and Loads.

Boats

Property	Data Type	Notes
id	Integer	The id of the boat. Datastore automatically generates it. Don't add it yourself as a property of the entity.
name	String	Name of the boat.
type	String	Type of the boat. E.g., Sailboat, Catamaran, etc.
length	Integer	The length of the boat in feet.
loads	array	An array of load entities.
self	String	URL string to the Boat's info of itself.

Loads

Property	Data Type	Notes
id	Integer	The id of the load. Datastore automatically generates it.
weight	Integer	The volume of the load, an integer
carrier	string	The current boat id.
content	String	A description of what the load is.
self	String	URL string to the load's info of itself.

Users

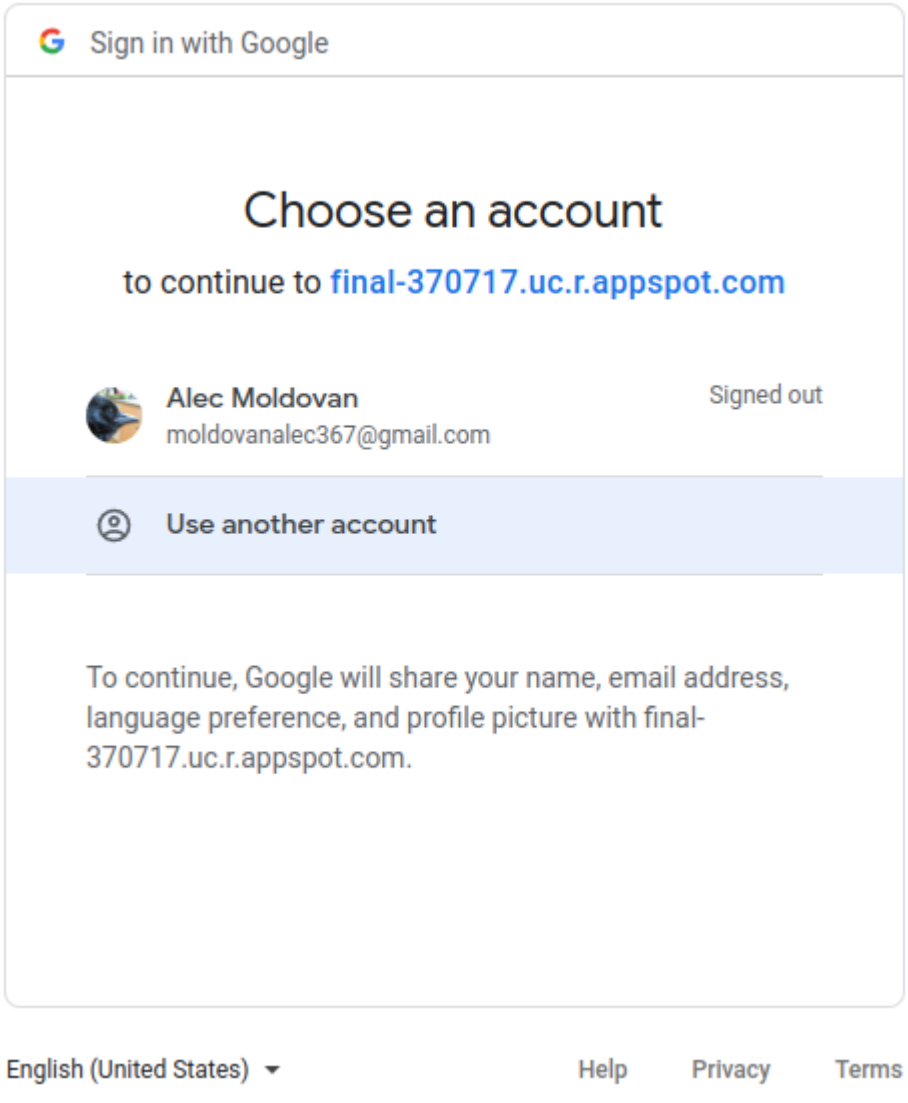
Property	Data Type	Notes
id	Integer	The user id generated by google oauth server.
name	string	Name of logged in user.

Login
GET /login

Description
Logs in a user by redirecting them to a Google consent dialog where they can grant permission to the application to access their profile information.

Response
- 302: Redirects the user to the consent dialog.
- 500: Returns an error object if an internal server error occurs.

Example response
Redirects the user to the Google consent dialog. Here the user can sign in with an existing account or create a new account and sign in with that.



Authenticate with Google

GET /auth/google: Authenticates a user using Google.

Description

Authenticates a user using Google by exchanging the `code` query parameter for access and ID tokens.

Query parameters

- code: The code returned by the Google consent dialog (string, required)

Response

- 302: Redirects the user to the `/profile` page with the ID token set in a cookie and the response header.
- 500: Returns an error object if an internal server error occurs.

Example query parameters

?code=4/4wA4ZK-6ZW8_...

Example response

Redirects the user to the `/profile` page with the ID token set in a cookie and the response header.

Log out

GET /logout: Logs out a user.

Description

Logs out a user by clearing the ID token cookie and redirecting them to the `/` page.

Response

- 302: Redirects the user to the `/` page.

Example response

Redirects the user to the `/` page.

Create a Boat

POST /boats

Description

Creates a new boat. Authentication required. The function first extracts the user ID from the request object, then calls the post_boat method on the Boat object, passing in the request body parameters and the user ID. The post_boat method checks if the required fields are present in the request object, and if so, saves the new boat to the datastore. It then waits for the datastore to save the boat, generates a self link for the boat using the request information, and updates the boat with the self link. If the operation is successful, the newly created boat object is returned.

Body parameters

- name: The name of the boat (string, required)
- type: The type of the boat (string, required)
- length: The length of the boat in meters (number, required)

Response Codes

- 201: Returns the newly created boat object.
- 400: Returns an error object if the request body is missing required parameters or contains invalid data.
- 500: Returns an error object if an internal server error occurs.

Example request body

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

Example response

```
{
  "id": "5752329273868288",
  "name": "Sea Witch",
```

"type": "Catamaran",

"length": 28,

"owner": "111348345198937810183",

"loads": [],

"self": "https://final-370717.uc.r.appspot.com/boats/5752329273868288"

}

Get a Boat by boat id

GET /boats/:id

Description

Returns the boat with the specified ID if the boat is owned by the authenticated user.

Query parameters

- id: The ID of the boat to retrieve (number, required)

Response

- 200: Returns the boat with the specified ID if it exists and is owned by the user.
- 404: Returns an error object if no boat with the specified ID exists.
- 403: Returns an error object if the authenticated user is not authorized to access the boat.
- 500: Returns an error object if an internal server error occurs.

Example path parameters

/boats/1

Example response

{

"type": "Catamaran",

"id": "5647577303220224",

"self": "https://final-370717.uc.r.appspot.com/boats/5647577303220224",

"length": 28,

"owner": "111348345198937810183",

"loads": [],

"name": "Sea Witch"

}

Update a Boat

PUT /boats/:id

Description

Updates a boat with the specified ID.

Path parameters

- id: The ID of the boat to be updated (number, required)

Body parameters

- name: The new name of the boat (string, required)
- type: The new type of the boat (string, required)
- length: The new length of the boat in meters (number, required)

Response

- 200: Returns the updated boat object.
- 400: Returns an error object if the request body is missing required parameters or contains invalid data.
- 403: Returns an error object if the user is unauthorized to access the boat.
- 404: Returns an error object if the boat with the specified ID does not exist.
- 500: Returns an error object if an internal server error occurs.

Example request body

{

"name": "Sea Cucumber",

"type": "Yacht",

"length": 100

}

Example response


```
{
   "length": 100,
   "id": "5660930088108032",
   "type": "Yacht",
   "self": "https://final-370717.uc.r.appspot.com/boats/5660930088108032",
   "name": "Sea Cucumber",
   "loads": [],
   "owner": "111348345198937810183"
}
```


List all Boats

GET /boatspublic

Description

Retrieves a paginated list of boats.

Query parameters

- cursor: The cursor for the starting point of the query (string, optional)

Example query parameters

/?cursor=cD0yMDAx...

Response Codes

- 200: Returns a paginated list of boats.
- 404: Returns an error object if no boats are found.
- 406: Returns an error message if the request does not accept JSON.
- 500: Returns an error object if an internal server error occurs.

Example response (200)


```
{
  "items": [
```

```
    {
      "id": 1,
      "name": "Boaty McBoatface",
      "type": "Yacht",
      "length": 20,
      "owner": 5,
      "loads": [],
      "self": "http://localhost:8080/boats/1"
    },
    {
      "id": 2,
      "name": "Sailor Moon",
      "type": "Sailboat",
      "length": 15,
      "owner": 5,
      "loads": [],
      "self": "http://localhost:8080/boats/2"
    }
  ],
  "next": "http://localhost:8080/boats/?cursor=cD0yMDAx..."
}
```


Example response (404)


```
{
  "message": "No boats found.",
  "status": 404
}
```


Example response (406)


```
"Not Acceptable"
```

Example response (500)


```
{
  "message": "An internal server error occurred.",
  "status": 500
}
```


List all Boats for Logged in User

GET /boats

Description

Retrieves a list of boats owned by the user, with pagination of 5 boats per page.

Response Codes

- 200: Successfully fetched boats for user
- 401: User not authenticated
- 404: User does not have any boats
- 500: Internal server error

Example Response (200)


```
{
    "items": [
        {
            "id": 1,
            "name": "Sailboat",
            "type": "Sail",
            "length": 24
        },
        {
            "id": 2,
            "name": "Motor Yacht",
            "type": "Power",
            "length": 40
        }
```

```
    ],
    "next": "http://localhost:3000/boats/?cursor=12345"
}
```


Example Response (401)


```
{
    "message": "User not authenticated"
}
```


Example Response (404)


```
{
    "message": "User does not have any boats"
}
```


Example Response (500)


```
{
    "message": "Internal server error"
}
```


Delete a Boat

DELETE /boats/:id

Description

Deletes a boat with the provided boat id and only deletes if the logged in user owns the boat.

Parameters

- id: The ID of the boat to be deleted

Headers

- Authorization: [ID Token] (required)

Response Codes

- 200: The boat was successfully deleted
- 401: User is not authenticated
- 403: User is not authorized to delete this boat
- 404: No boat with the specified ID exists
- 500: Internal server error

Example Response (200):

```
{
  "message": "Boat successfully deleted"
}
```

Example Response (401):

```
{
  "message": "User not authenticated"
}
```

Example Response (403):

```
{
  "message": "Unauthorized to access this boat"
}
```

Example Response (404):

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

Example Response (500):

```
{
  "message": "Internal server error"
}
```

Create a Load

POST /loads

Description

This route is used to create a new load in the datastore.

Headers

- Authorization: The JSON Web Token (JWT) used to authenticate the user.

Request Body

- weight (required): The weight of the load in pounds.
- content (required): The content of the load.
- carrier: The carrier transporting the load.

Response Codes

- 201: The load was successfully created and saved in the datastore.
- 400: The request object is missing at least one of the required attributes.
- 401: The user is not authenticated.
- 404: The specified load could not be found.
- 500: There was an error authenticating the user or saving the load in the datastore.

Example Response (201)

{

"id": 1,

"weight": 100,

"content": "Furniture",

"carrier": "ABC Carrier",

"self": "https://localhost:3000/loads/1"

}

Example Response (400)

{
"message": "The request object is missing at least one of the required attributes",
"status": 400
}

Example Response (401)

"User not authenticated"

Example Response (404)

{
"message": "The specified load could not be found",
"status": 404
}

Example Response (500)

{
"message": "Error authenticating user: Invalid JWT",
"status": 500
}

Get a Load

GET /loads/:id

Description
This route is used to retrieve a load by its id. User must be authenticated. The route returns a JSON object with the load information if the request is successful.

Parameters
- id: The id of the load to be retrieved. Required.

Request Example
GET /loads/12345 HTTP/1.1

Response Codes:
- 200: The load was successfully retrieved.
- 401: The user is not authenticated.
- 404: No load with the specified id was found.
- 500: An error occurred while trying to retrieve the load.

Example Response (200)

```
{
    "weight": 5,
    "self": "https://final-370717.uc.r.appspot.com/loads/5720442564247552",
    "id": "5720442564247552",
    "carrier": null,
    "content": "LEGO Blocks"
}
```


Example Response (401)
"User not authenticated"

Example Response (404)

```
{
    "message": "No load with this load_id exists",
    "status": 404
}
```


Example Response (500)
“Internal Server Error”

Get all Loads on a Boat

GET /boats/:boat_id/loads

Description

This route gets all loads for a specific boat with the given boat_id. The user must be authenticated in order to access this route.

Parameters

boat_id (required): The id of the boat for which to get the loads

Request Example

GET /boats/12345/loads

Response Codes

200: Success - Returns all loads for the specified boat.

404: Failure – No boat exists with that boat id.

500: Failure – Internal Server Error

Response Example (200)


```
{
   "loads": [
      {
         "content": "Stuffed Animals",
         "self": "https://final-370717.uc.r.appspot.com/loads/5732778582736896",
         "carrier": "5732000589676544",
         "weight": 10,
         "id": "5732778582736896"
      },
      {
         "id": "5714920645591040",
         "content": "Food",
         "weight": 3,
         "self": "https://final-370717.uc.r.appspot.com/loads/5714920645591040",
```

```
            "carrier": "5732000589676544"
        },
        {
            "self": "https://final-370717.uc.r.appspot.com/loads/5712466340216832",
            "content": "Furnitures",
            "id": "5712466340216832",
            "weight": 7,
            "carrier": "5732000589676544"
        },
        {
            "self": "https://final-370717.uc.r.appspot.com/loads/5757464578359296",
            "weight": 4,
            "content": "Toys",
            "carrier": "5732000589676544",
            "id": "5757464578359296"
        }
    ],
    "type": "Yacht",
    "name": "Lucifer",
    "self": "https://final-370717.uc.r.appspot.com/boats/5732000589676544",
    "id": "5732000589676544",
    "length": 15,
    "owner": "111348345198937810183"
}
```


Response Example (404)


```
{
message: "No boat with this boat_id exists",
status: 404
}
```


Response Example (500)

“Internal Server Error”

List all Loads

GET /loads

Description

This route retrieves all loads and paginates by 5. The cursor parameter can be included in the query to specify the next page of results.

Parameters

cursor (optional): The cursor value for the next page of results.

Request Example

GET /loads?cursor=<cursor_value>

Response Codes

- 200: Successfully retrieved loads.
- 500: Internal server error occurred.

Example Response (200):


```
{
   "items": [
      {
         "weight": 3,
         "self": "http://localhost:8080/loads/5088344509775872",
         "id": "5088344509775872",
         "content": "Food",
         "carrier": "5711226101301248"
      },
      {
         "content": "Stuffed Animals",
         "id": "5091190730915840",
         "weight": 10,
         "self": "http://localhost:8080/loads/5091190730915840",
         "carrier": null
      },
      {
         "self": "http://localhost:8080/loads/5106202648248320",
```

```
            "weight": 10,
            "content": "Stuffed Animals",
            "id": "5106202648248320"
        },
        {
            "id": "5113278539759616",
            "content": "Stuffed Animals",
            "weight": 10,
            "self": "http://localhost:8080/loads/5113278539759616"
        },
        {
            "id": "5146163359514624",
            "self": "http://localhost:8080/loads/5146163359514624",
            "carrier": null,
            "weight": 5,
            "content": "LEGO Blocks"
        }
    ],
    "next": "https://final-370717.uc.r.appspot.com/loads/?
cursor=CikSI2oOc35maW5hbC0zNzA3MTdyEQsSBGxvYWQYgICAuIiNkgkMGAAgAA=="
}
```


Example Response (500):
“Internal Server Error”

Put a Load on boat
PUT /boats/:boat_id/loads/:load_id

Description
This route allows a user to update a load on a boat by providing the boat_id and load_id in the URL parameters. The user must be authenticated to access this route. The weight and content of the load must be provided in the request body.

Query Parameters

- boat_id (required): The id of the boat the load is on
- load_id (required): The id of the load to be updated

Request Body Parameters

- weight (required): The weight of the load
- content (required): The content of the load

Request Example:

PUT /boats/1/loads/2

{

"weight": 500,

"content": "Furniture"

}

Response Codes

- 200: The load was updated successfully
- 400: Missing required fields in request body
- 401: User not authenticated.
- 404: No boat or load with the provided id exists
- 403: The load already has a carrier
- 500: An error occurred while updating the load

Example Response (200):


```
{
    "self": "https://final-370717.uc.r.appspot.com/loads/5679858948505600",
    "weight": 5,
    "carrier": "5726763346821120",
    "content": "LEGO Blocks",
    "id": "5679858948505600"
}
```


Example Response (400)


```
{
    "message": "The request object is missing at least one of the required attributes",
    "status": 400
}
```


Example Response (401)

"User not authenticated"

Example Response (404)


```
{
    "message": "No boat with this boat_id exists",
    "status": 404
}
```


OR


```
{
    "message": "No load with this load_id exists",
    "status": 404
}
```


Example Response (403)


```
{
    "message": "Load already has a carrier",
    "status": 403
}
```


Example Response (500)

“Internal Server Error”

Delete a Load

DELETE /loads/:id

Description

This route allows a user to delete a load by its id. The user must be authenticated in order to delete the load.

Query Parameters

- id (required): The id of the load to delete.

Request Example

DELETE /loads/12345

Response Codes

- 204: The load was successfully deleted.
- 400: The request was invalid.
- 401: The user is not authenticated.
- 404: The load does not exist or is not on the specified boat.
- 500: There was an error deleting the load.

Example Response (204)

No Content

Example Response (400)

{

"message": "Invalid request"

}

Example Response (401)

{

"message": "User not authenticated"

}

Example Response (404)

{

"message": "Load does not exist or is not on specified boat"

}

Example Response (500)

{

"message": "Error deleting load"

}

Remove Load from Boat

DELETE /boats/:boat_id/loads/:load_id

Description

This route removes a load from a boat by its boat_id and load_id. The user must be authenticated in order to use this route.

Query Parameters

- boat_id: The id of the boat to remove the load from (required)
- load_id: The id of the load to remove from the boat (required)

Request Example

DELETE /boats/123/loads/456

Response Codes

- 204: Successful removal of load from boat
- 400: Invalid boat_id or load_id provided
- 401: User not authenticated
- 403: User does not have permission to remove load from boat
- 404: No boat or load with provided ids exists

- 500: Server error

Example Response (204)
No Content

Example Response (400)
{

"message": "Invalid boat_id or load_id provided"

}

Example Response (401)
{

"message": "User not authenticated"

}

Example Response (403)
{

"message": "User does not have permission to remove load from boat"

}

Example Response (404)
{message: "No load with this load_id exists", status: 404}

OR

{message: "No boat with this boat_id exists", status: 404}

Example Response (500)
“Internal Server Error”