Clint’s Bike Rental: API Spec

CS 493: Cloud Application Development

Spring 2023

Oregon State University

Last Update: June 3:00 pm MST

GCP URL for Application: https://portfolio-lohrcl.uc.r.appspot.com

URL for account creation/login: https://portfolio-lohrcl.uc.r.appspot.com

Change Log

Version	Change	Date
1.0	Initial version.	June 01, 2023

Data Model

The app stores three types of entities in Datastore, Users, Bikes and Components.

Users

Property	Data Type	Notes	Required	Valid Values
id	Integer	The id of the User. This property is automatically generated by Google Datastore.	Yes (autogenerated)	5183261005316096
renter_id	String	The unique id representing the owner of the JWT. This property is automatically generated by Auth0 and is displayed when logging in/creating an Auth0 account.	Yes (autogenerated)	“auth0|647c15f094499d0c9ac67ca3”
nickname	String	The nickname of the user. This property is automatically generated by Auth0.	Yes (autogenerated)	“c”, “jim”
verified	Boolean	A boolean value representing the user’s verification status. This property is automatically generated by Auth0.	Yes (autogenerated)	“false”, “true”
email	String	The email address of the user. This property is automatically generated by Auth0.	Yes (autogenerated)	“c@cheese.com”
rental	Array	The id of the bike(s) the user is renting.	Yes (autogenerated)	`[ { "id": 5694034018304000, "self": "http://127.0.0.1:8080/bikes/5694034018304000" } ]`

Bikes

Property	Data Type	Notes	Required	Valid Values
id	Integer	The id of the bike. This property is automatically generated by Google Datastore.	Yes (autogenerated)	5685034249879552
manufacturer	String	The manufacturer of the bike.	Yes	“GT”, “Yeti”, “Evil”
type	String	The type of the bike.	Yes	“mountain”, “cruiser”, “bmx”
model_year	Integer	The year of the bike	Yes	1999, 2020, 2023
bike_size	String	The size of the bike. E.g. small, medium, large	Yes	“small”, “medium”, “large”
specs	Array	The components that the bike includes. Component data includes the id of that component as an integer and the name of that component as a string.	Yes (autogenerated)	`[ { "description": "tires", "id": 5685839489138688, "self": "http://127.0.0.1:8080/components/5685839489138688" } ]`
rentee	Integer	The id of the user that is renting the bike.	Yes (autogenerated)	5646633081503744

Components

Property	Data Type	Notes	Required	Valid Values
id	Integer	The id of the component. This property is automatically generated by Google Datastore.	Yes (autogenerated)	5685839489138688
manufacturer	String	The manufacturer of the component.	Yes	“SRAM”, “Shimano”, “Crank Brothers”
description	String	A description of the item. E.g. tires, rear suspension	Yes	“shifter”, “tires”, “cog”
condition	Integer	The condition of the component on a 1 to 5 scale (1 being “needs replaced”, 5 being “like new”)	Yes	1, 2, 3, 4, 5
carrier	Dictionary	The bike that is carrying the component. Component data includes the id of that bike as an integer and the manufacturer of the bike as a string.	Yes (autogenerated)	null, { "id": 5685034249879552, "manufacturer": "Cannondale", "self": "http://127.0.0.1:8080/bikes/5685034249879552" }

Non-user Entities Relationship

The non-user entities relationship is between the ‘bike’ entity and the ‘component’ entity. A component can be installed onto a bike and uninstalled from a bike. A component can only be installed on one bike at a time. When a component is installed on a bike, the ‘carrier’ property of the component is updated to show the id of the bike that the component is installed on, the manufacturer of the bike, and a self link that points to the most canonical representation of that bike entity instance. If a component is not installed on a bike, the ‘carrier’ property is ‘null’.

A bike can have many components installed on it at one time. When a bike has a component(s) installed onto it, the ‘specs’ property of the bike is updated to show the id of the component, a description of the component, and a self link that points to the most canonical representation of that component entity instance. If a bike has no components installed on it, the ‘specs’ property is an empty array e.g. [ ].

If a component is modified while installed on a bike, the ‘specs’ property on the bike is updated to represent any changes. If a component is deleted, the component is removed from the ‘specs’ property of the bike entity.

If a bike is deleted, the ‘carrier’ property of the component entity is updated and the value of ‘carrier’ is reset to ‘null’.

User Entity Model Description

The ‘User’ entity in this application is generated when a user of the application logs in or creates an Auth0 account. The information generated from Auth0 is used to create the property of a user entity.

User entities have two unique identifiers, the first identifier is the property ‘id’ which is the unique id generated by Datastore. This id is used in the urls to make calls to endpoints that require a user id.

The second identifier is the renter_id which is the unique id representing the owner of the JWT. This property is automatically generated by Auth0 and is displayed when logging in/creating an Auth0 account. This id is used to verify the owner of an entity when making protected requests.

When a user is making a request that requires the user’s id, the user will use the ‘id’ property of the user entity properties. This property is represented in Postman by either ‘user_id1’ or ‘user_id2’.

The application maps the supplied JWT to the identifier of a user by comparing the ‘sub’ property of the JWT against the ‘renter_id’ of the entity to which the request was made. If the ‘renter_id’ does not match the ‘sub’ property of the JWT, the request is rejected and the application responds with an error message.

User Entity and Non-user Entity Relationship

The user entity and non-user entity relationship is between the ‘User’ entity and the ‘Bike’ entity, respectively. A bike can be rented to a user and returned by a user. A bike can only be rented to one user at a time. When a bike is rented by a user, the ‘rentee’ property of the bike entity is updated to show the id of the user renting the bike. If the bike is not currently rented by a user the value of ‘rentee’ is ‘null’.

A user can rent multiple bikes at one time. When a user rents a bike(s), the ‘rental’ property of the user is updated to show the id of the bike(s) that are being rented and a self link that points to the most canonical representation of that bike entity instance. If a user is not currently renting any bikes, the value of the ‘rental’ property is an empty array e.g. [ ].

If a bike is deleted, the bike entity is removed from the ‘rental’ property of the user.

Directions for Use

To use this application, begin by following the link provided at the top of this document to login or create an Auth0 account.

Once logged in to the Auth0 account, copy the ‘ID Token’ that is displayed and paste it into the Postman environment ‘Portfolio-Project’ as the value for the variable ‘jwt1’.

Once the ‘jwt1’ variable is set, return to the Auth0 webpage where the ID Token was copied and click the “logout” link.

Once logged out of the first Auth0 account, login or create a second Auth0 account that is different from the first.

Once logged in to the Auth0 account, copy the ‘ID Token’ that is displayed and paste it into the Postman environment ‘Portfolio-Project’ as the value for the variable ‘jwt2’ and save the environment.

Next, run the first request in the Postman collection ‘Portfolio-Project’ titled ‘Get All Users’. Copy the value of the property ‘id’ for each of the users into the Postman environment as the value for ‘user_id1’ and user_id2’.

NOTE: The ‘id’ for ‘user_id1’ must be the id of the first user created using Auth0 and the ‘id’ for ‘user_id2’ must be the id of the second user created using Auth0.

View all Users

List all the users.

```
GET /users
```

Protected:

No: A valid JWT is not required to view all users.

Request

Path Parameters

None

Request Body

None

Response

Response Body Format

JSON

Response MIME Type

application/json

Response Statuses

Outcome	Status Code	Notes
Success	200 OK	

Response Examples

- The value of the attribute `rental` is the id of the bike currently being rented by the user. If there is no bike currently being rented by the user, the value of `rental` is an empty array E.g. [ ]
- The attribute `self` is added to the JSON response body but is not stored in Datastore. The value of this attribute is a self link that directs users to the location of that resource.
- The property `total_items` is added to the JSON response body but is not stored in Datastore. The value of this property is the total number of users in the '/users' collection.

Success

```
Status: 200 OK
{
  "users": [
    {
      "renter_id": "auth0|647c15f094499d0c9ac67ca3",
      "nickname": "c",
      "verified": false,
      "email": "c@cheese.com",
      "rental": [],
      "id": 5183261005316096,
      "self": "http://127.0.0.1:8080/users/5183261005316096"
    },
    {
      "renter_id": "auth0|647bed1afc60c55ea864616e",
      "nickname": "jim",
      "verified": false,
      "email": "jim@cheese.com",
      "rental": [],
            "id": 5190813503979520,
            "self": "http://127.0.0.1:8080/users/5190813503979520"
        }
    ],
    "total_items": 2
```

View all Bikes

List all the bikes.

```
GET /bikes
```

Protected:

Yes: A valid JWT is required to view bikes associated with the renter.

Request

Path Parameters

None

Request Body

None

Response

Response Body Format

JSON

Response MIME Type

application/json

Response Statuses

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

Response Examples

- The value of the attribute `specs` is the id of the component currently installed on the bike. If there is no component on the bike, the value of `specs` is an empty array E.g. [ ]
- The attribute `self` is added to the JSON response body but is not stored in Datastore. The value of this attribute is a self link that directs users to the location of that resource.
- The attribute `next` is added to the JSON response body but is not stored in Datastore. The value of this attribute is a link to the next page of results for bikes.
- The property `total_items` is added to the JSON response body but is not stored in Datastore. The value of this property is the total number of bikes in the '/bikes' collection.

Success

```
Status: 200 OK
{
    "bikes": [
        {
            "specs": [],
            "rentee": 5183261005316096,
            "model_year": 2024,
            "type": "mountain",
            "bike_size": "large",
            "manufacturer": "Yeti",
            "id": 5201568437633024,
            "self": "http://127.0.0.1:8080/bikes/5201568437633024"
        },
        {
            "specs": [],
            "rentee": 5183261005316096,
            "model_year": 2020,
            "type": "cruiser",
            "bike_size": "large",
            "manufacturer": "GT",
            "id": 5707975213711360,
            "self": "http://127.0.0.1:8080/bikes/5707975213711360"
        }
    ],
    "total_items": 2
}
```

View all Components

List all the components.

```
GET /components
```

Protected:

No: A valid JWT is not required to view all components.

Request

Path Parameters

None

Request Body

None

Response

Response Body Format

JSON

Response MIME Type

application/json

Response Statuses

Outcome	Status Code	Notes
Success	200 OK	

Response Examples

- The value of the attribute `carrier` is the id of the bike that the component is currently installed on. If the component is not currently installed on a bike, the value of `carrier` is an empty array E.g. [ ]
- The attribute `self` is added to the JSON response body but is not stored in Datastore. The value of this attribute is a self link that directs users to the location of that resource.
- The attribute `next` is added to the JSON response body but is not stored in Datastore. The value of this attribute is a link to the next page of results for components.
- The property `total_items` is added to the JSON response body but is not stored in Datastore. The value of this property is the total number of components in the '/components' collection.

Success

```
Status: 200 OK
{
    "components": [
        {
            "condition": 5,
            "carrier": {
                "id": 5685034249879552,
                "manufacturer": "Cannondale",
                "self": "http://127.0.0.1:8080/bikes/5685034249879552"
            },
            "description": "shifter",
            "manufacturer": "SRAM",
            "id": 5122084296458240,
            "self": "http://127.0.0.1:8080/components/5122084296458240"
        },
        {
            "condition": 5,
            "carrier": {
                "id": 5664969639067648,
                "manufacturer": "Specialized",
                "self": "http://127.0.0.1:8080/bikes/5664969639067648"
            },
            "description": "shifter",
            "manufacturer": "SRAM",
            "id": 5644784165191680,
            "self": "http://127.0.0.1:8080/components/5644784165191680"
        }
    ],
    "total_items": 2
```

Create a Bike

Allows you to create a new bike.

```
POST /bikes
```

Protected:

Yes: A valid JWT is required to create a bike entity.

Request

Path Parameters

None

Request Body

Required

Request Body Format

JSON

Request JSON Attributes

Name	Description	Required
manufacturer	The manufacturer of the bike.	Yes
type	The type of the bike. E.g. Mountain, Cruiser, etc.	Yes
model_year	The year of the bike	Yes
bike_size	The size of the bike. E.g. small, medium, large	Yes

Request Body Example: Successful

```
{
   "manufacturer": "Cannondale",
   "type": "mountain",
   "model_year": 2020,
   "bike_size": "large"
}
```

Request Body Examples: Failure

```
{
   "manufacturer": "Cannondale",
   "type": "mountain",
   "model_year": 2020
}
```

```
{
   "manufacturer": "Cannondale",
   "type": "mountain",
   "model_year": 2020,
   "bike_size": "large"
    “specs”: []
}
```

```
{
   "manufacturer": "Cannondale",
   "type": "mountain",
   "model_year": “2020”,
   "bike_size": "large"
}
```

```
<table>
            <tr>
                        <th>manufacturer</th>
                        <td>Cannondale</td>
            </tr>
            <tr>
                        <th>type</th>
                        <td>mountain</td>
            </tr>
            <tr>
                        <th>model_year</th>
                        <td>2020</td>
            </tr>
</table>
```

Response

Response Body Format
JSON

Response MIME Type
application/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 bike must not be created, and 400 status code must be returned.
Failure	405 Method Not Allowed	If the request method is not one of the allowed request methods (POST, GET), the request will not be fulfilled, and 405 status code is returned.
Failure	406 Not Acceptable	If the request only accepts MIME-types that are not supported by the server, the bike is not created, and 406 status code is returned. Accepted content-type for this request method is 'application/json'.
Failure	415 Unsupported Media Type	If the request is a content-type that is not supported by the server, the boat is not created, and 415 status code is returned. Accepted content-type for this request method is 'application/json'.

Response Examples

- Datastore will automatically generate an ID and store it with the entity being created. This value will be sent in the response body as shown in the example.
- The value of the attribute `specs` is the id of the component currently installed on the bike. If there is no component on the bike, the value of `specs` is an empty array E.g. [ ]
- The value of the attribute `rentee` is the id of the user that is currently renting the bike. If there is no rentee for the bike, the value of `rentee` is an empty array E.g. [ ]
- The attribute `self` is added to the JSON response body but is not stored in Datastore. The value of this attribute is a self link that directs users to the location of that resource.

Success

```
Status: 201 Created
{
    "manufacturer": "Cannondale",
    "type": "mountain",
    "model_year": 2020,
    "bike_size": "large",
    "specs": [],
    "rentee": null,
    "id": 5685034249879552,
    "self": "http://127.0.0.1:8080/bikes/5685034249879552"
}
```

Failure

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

```
Status: 405 Method Not Allowed
<!doctype html>
<html lang=en>
<title>405 Method Not Allowed</title>
<h1>Method Not Allowed</h1>
<p>The method is not allowed for the requested URL.</p>
```

```
Status: 406 Not Acceptable
{
    "code": "Not Acceptable",
    "description": "The chosen media type is not supported for this request"
}
```

```
Status: 415 Unsupported Media Type
{
    "code": "Unsupported Media Type",
    "description": "Content type for this request must be application/json"
}
```

View a Bike

Allows you to get an existing bike

```
GET /bikes/:bike_id
```

Protected:

Yes: A valid JWT is required to view a bike entity.

Request

Path Parameters

Name	Description
bike_id	ID of the bike

Request Body

None

Response

Response Body Format

JSON

Response MIME Type

application/json

Response Statuses

Outcome	Status Code	Notes
Success	200 OK	
Failure	401 Unauthorized	If the bike to be viewed is not rented by the user. You must rent this bike before making any requests
Failure	403 Forbidden	If a bike is rented out to another user. You cannot view a bike that you aren't renting
Failure	404 Not Found	No bike with this bike_id exists
Failure	405 Method Not Allowed	If the request method is not one of the allowed request methods (PUT, PATCH, DELETE, GET), the request will not be fulfilled, and 405 status code is returned.

Response Examples

- The value of the attribute `specs` is the id of the component currently being carried by the bike. If there is no component on the bike, the value of `specs` is an empty array E.g. [ ]
- The attribute `self` is added to the JSON response body but is not stored in Datastore. The value of this attribute is a self link that directs users to the location of that resource.
- The attribute `rentee` is the id of the user currently renting this bike. If no user is renting the bike the value is 'null' The value of this attribute is a self link that directs users to the location of that resource.

Success

```
Status: 200 OK
{
   "specs": [],
   "rentee": 5646633081503744,
   "model_year": 2020,
   "type": "mountain",
   "bike_size": "large",
   "manufacturer": "Cannondale",
   "id": 5705076043677696,
   "self": "http://127.0.0.1:8080/bikes/5705076043677696"
}
```

Failure

```
Status: 401 Unauthorized
{
   "code": "Unauthorized",
   "description": "You must rent this bike before making any requests"
}
```

```
Status: 403 Forbidden
{
   "code": “Forbidden",
   "description": "You cannot view a bike that you aren't renting"
}
```

```
Status: 404 Not Found
{
   "code": “Not Found",
   "description": "No bike with this bike_id exists"
}
```

```
Status: 405 Method Not Allowed
<!doctype html>
<html lang=en>
<title>405 Method Not Allowed</title>
<h1>Method Not Allowed</h1>
<p>The method is not allowed for the requested URL.</p>
```

Modify a Bike (PUT)

Allows you to edit all attributes of a bike.

```
PUT /bikes/:bike_id
```

Protected:

Yes: A valid JWT is required to modify a bike entity.

Request

Path Parameters

Name	Description
bike_id	ID of the bike

Request Body

Required

Request Body Format

JSON

Request JSON Attributes

Name	Description	Required
manufacturer	The manufacturer of the bike.	Yes
type	The type of the bike. E.g. Mountain, Cruiser, etc.	Yes
model_year	The year of the bike	Yes
bike_size	The size of the bike. E.g. small, medium, large	Yes

Request Body Example: Successful

```
{
   "manufacturer": "Cannondale",
   "type": "mountain",
   "model_year": 2023,
   "bike_size": "small"
}
```

Request Body Examples: Failure

```
{
   "manufacturer": "Cannondale",
   "type": "mountain",
   "model_year": 2020
}
```

```
{
   "manufacturer": "Cannondale",
   "type": "mountain",
   "model_year": 2020,
   "bike_size": "large"
    “specs”: []
}
```

```
{
   "manufacturer": "Cannondale",
   "type": "mountain",
   "model_year": “2020”,
   "bike_size": "large"
}
```

```
<table>
        <tr>
                <th>manufacturer</th>
                <td>Cannondale</td>
        </tr>
        <tr>
                <th>type</th>
                <td>mountain</td>
        </tr>
        <tr>
                <th>model_year</th>
                <td>2020</td>
        </tr>
</table>
```

Response

Response Body Format

Success: No Body

Failure: JSON

Response MIME Type

application/json

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 bike must not be created, and 400 status code must be returned.
Failure	401 Unauthorized	If the bike to be viewed is not rented by the user. You must rent this bike before making any requests
Failure	403 Forbidden	If a bike is rented out to another user. You cannot modify a bike that you aren't renting
Failure	404 Not Found	No bike with this bike_id exists
Failure	405 Method Not Allowed	If the request method is not one of the allowed request methods (PUT, PATCH, DELETE, GET), the request will not be fulfilled, and 405 status code is returned.
Failure	406 Not Acceptable	If the request only accepts MIME-types that are not supported by the server, the bike is not created, and 406 status code is returned. Accepted content-type for this request method is 'application/json'.
Failure	415 Unsupported Media Type	If the request is a content-type that is not supported by the server, the boat is not created, and 415 status code is returned. Accepted content-type for this request method is 'application/json'.

Response Examples

Success

```
Status: 204 No Content
```

Failure

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

```
Status: 401 Unauthorized
{
   "code": "Unauthorized",
   "description": "You must rent this bike before making any requests"
}
```

```
Status: 403 Forbidden
{
   "code": “Forbidden",
   "description": "You cannot modify a bike that you aren't renting"
}
```

```
Status: 404 Not Found
{
   "code": “Not Found",
   "description": "No bike with this bike_id exists"
}
```

```
Status: 405 Method Not Allowed
<!doctype html>
<html lang=en>
<title>405 Method Not Allowed</title>
<h1>Method Not Allowed</h1>
<p>The method is not allowed for the requested URL.</p>
```

```
Status: 406 Not Acceptable
{
   "code": "Not Acceptable",
   "description": "The chosen media type is not supported for this request"
}
```

```
Status: 415 Unsupported Media Type
{
   "code": "Unsupported Media Type",
   "description": "Content type for this request must be application/json"
}
```

Modify a bike (PATCH)

Allows you to edit selected attributes of a bike.

```
Patch /bikes/:bike_id
```

Protected:

Yes: A valid JWT is required to modify a bike entity.

Request

Path Parameters

Name	Description
bike_id	ID of the bike

Request Body

Required

Request Body Format

JSON

Request MIME Type

application/json

Request JSON Attributes

Name	Description	Required
manufacturer	The manufacturer of the bike.	No
type	The type of the bike. E.g. Mountain, Cruiser, etc.	No
model_year	The year of the bike	No
bike_size	The size of the bike. E.g. small, medium, large	No

Request Body Example: Successful

```
{
   "model_year": 2023
}
```

Request Body Examples: Failure

```
{
   "manufacturer": "Cannondale",
   "type": "mountain",
   "model_year": “2020”,
   "bike_size": "small"
}
```

```
{
   "model_year": “2022”
}
```

```
<table>
        <tr>
                <th>model_year</th>
                <td>2023</td>
        </tr>
</table>
```

Response

Response Body Format

Success: No Body

Failure: JSON

Response MIME Type

application/json

Response Statuses

Outcome	**Status Code**	**Notes**
Success	204 No Content	
Failure	401 Unauthorized	If the bike to be viewed is not rented by the user. You must rent this bike before making any requests
Failure	403 Forbidden	If a bike is rented out to another user. You cannot modify a bike that you aren't renting
Failure	404 Not Found	No bike with this bike_id exists
Failure	405 Method Not Allowed	If the request method is not one of the allowed request methods (PUT, PATCH, DELETE, GET), the request will not be fulfilled, and 405 status code is returned.
Failure	406 Not Acceptable	If the request only accepts MIME-types that are not supported by the server, the bike is not created, and 406 status code is returned. Accepted content-type for this request method is 'application/json'.
Failure	415 Unsupported Media Type	If the request is a content-type that is not supported by the server, the boat is not created, and 415 status code is returned. Accepted content-type for this request method is 'application/json'.

Success

```
Status: 204 No Content
```

Failure

```
Status: 401 Unauthorized
{
    "code": "Unauthorized",
    "description": "You must rent this bike before making any requests"
}
```

```
Status: 403 Forbidden
{
    "code": “Forbidden",
    "description": "You cannot modify a bike that you aren't renting"
}
```

```
Status: 404 Not Found
{
    "code": “Not Found",
    "description": "No bike with this bike_id exists"
}
```

```
Status: 405 Method Not Allowed
<!doctype html>
<html lang=en>
<title>405 Method Not Allowed</title>
<h1>Method Not Allowed</h1>
<p>The method is not allowed for the requested URL.</p>
```

```
Status: 406 Not Acceptable
{
    "code": "Not Acceptable",
    "description": "The chosen media type is not supported for this request"
}
```

```
Status: 415 Unsupported Media Type
{
    "code": "Unsupported Media Type",
    "description": "Content type for this request must be application/json"
}
```

Delete a Bike

Allows you to delete a bike.

```
DELETE /bikes/:bike_id
```

Protected:

Yes: A valid JWT is required to delete a bike entity.

Request

Path Parameters

Name	Description
bike_id	ID of the bike

Request Body

None

Response

No body

Response Body Format

Success: No body

Failure: JSON

Response MIME Type

application/json

Response Statuses

Outcome	Status Code	Notes
Success	204 No Content	
Failure	401 Unauthorized	If the bike to be deleted is not rented by the user. You must rent this bike before making any requests
Failure	403 Forbidden	If a bike is rented out to another user. You cannot delete a bike that you aren’t renting
Failure	404 Not Found	No bike with this bike_id exists
Failure	405 Method Not Allowed	If the request method is not one of the allowed request methods (PUT, PATCH, DELETE, GET), the request will not be fulfilled, and 405 status code is returned.

Response Examples

Success

```
Status: 204 No Content
```

Failure

```
Status: 401 Unauthorized
{
    "code": "Unauthorized",
    "description": "You must rent this bike before making any requests"
}
```

```
Status: 403 Forbidden
{
    "code": “Forbidden",
    "description": "You cannot modify a bike that you aren't renting"
}
```

```
Status: 404 Not Found
{
    "code": “Not Found",
    "description": "No bike with this bike_id exists"
}
```

```
Status: 405 Method Not Allowed
<!doctype html>
<html lang=en>
<title>405 Method Not Allowed</title>
<h1>Method Not Allowed</h1>
<p>The method is not allowed for the requested URL.</p>
```

Rent Bike to a User

Assigns a specified component to a specified bike

```
PUT /users/:user_id/bikes/:bike_id
```

Protected:

Yes: A valid JWT is required to rent a bike to a user.

Request

Path Parameters

Name	Description
user_id	ID of the user
bike_id	ID of the bike

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 bike exists with this bike_id, a user exists with this user_id and the bike is not currently being rented to another user.
Failure	403 Forbidden	The bike is already rented to another user.
Failure	404 Not Found	The specified bike and/or user does not exist
Failure	405 Method Not Allowed	If the request method is not one of the allowed request methods (PUT, DELETE), the request will not be fulfilled, and 405 status code is returned.

Response Examples

Success

```
Status: 204 No Content
```

Failure

```
Status: 403 Forbidden
{
    "code": "Forbidden",
    "description": "This bike is currently rented out"
}
```

```
Status: 404 Not Found
{
    "code": “Not Found",
    "description": "The specified bike and/or user does not exist"
}
```

```
Status: 405 Method Not Allowed
<!doctype html>
<html lang=en>
<title>405 Method Not Allowed</title>
<h1>Method Not Allowed</h1>
<p>The method is not allowed for the requested URL.</p>
```

Return Bike from a User

A specified component is removed from the specified bike carrying it.

```
DELETE /users/:user_id/bikes/:bike_id
```

Protected:

Yes: A valid JWT is required to return a bike entity from a user.

Request

Path Parameters

Name	Description
user_id	ID of the user
bike_id	ID of the bike

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 bike exists with this bike_id, a user exists with this user_id and the bike is not currently being rented to another user.
Failure	401 Unauthorized	If a bike is rented out by another user. You cannot return a bike that is rented to another user
Failure	404 Not Found	The specified bike and/or user does not exist
Failure	405 Method Not Allowed	If the request method is not one of the allowed request methods (PUT, DELETE), the request will not be fulfilled, and 405 status code is returned.

Response Examples

Success

```
Status: 204 No Content
```

Failure

```
Status: 401 Forbidden
{
    "code": "Unauthorized",
    "description": "You cannot return a bike that is rented to another user"
}
```

```
Status: 404 Not Found
{
    "code": “Not Found",
    "description": "No bike with this bike_id is rented to a user with this user_id"
}
```

```
Status: 405 Method Not Allowed
<!doctype html>
<html lang=en>
<title>405 Method Not Allowed</title>
<h1>Method Not Allowed</h1>
<p>The method is not allowed for the requested URL.</p>
```

Create a Component

Allows you to create a new component.

```
POST /components
```

Protected:

No: A valid JWT is not required to create a component entity.

Request

Path Parameters

None

Request Body

Required

Request Body Format

JSON

Response MIME Type

application/json

Request JSON Attributes

Name	**Description**	**Required?**
manufacturer	The manufacturer of the component.	Yes
description	A description of the component. E.g., tires, pedals, etc.	Yes
condition	The condition of the component on a 1 to 5 scale.	Yes

Request Body Example: Successful

```
{
   "manufacturer": "SRAM",
   "description": "shifter",
   "condition": 5
}
```

Request Body Examples: Failure

```
{
   "manufacturer": "SRAM",
   "description": "shifter"
}
```

```
{
   "manufacturer": "SRAM",
   "description": "shifter",
   "condition": “five”
}
```

```
<table>
		<tr>
				<th>Manufacturer</th>
				<td>SRAM</td>
		</tr>
		<tr>
				<th>description</th>
				<td>shifter</td>
		</tr>
		<tr>
				<th>condition</th>
				<td>5</td>
		</tr>
</table>
```

Response

Response Body Format

Success: No body

Failure: 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 component will not be created, and 400 status code must be returned.
Failure	405 Method Not Allowed	If the request method is not one of the allowed request methods (POST, GET), the request will not be fulfilled, and 405 status code is returned.
Failure	406 Not Acceptable	If the request only accepts MIME-types that are not supported by the server, the component is not created, and 406 status code is returned. Accepted content-type for this request method is ‘application/json’.
Failure	415 Unsupported Media Type	If the request is a content-type that is not supported by the server, the component is not created, and 415 status code is returned. Accepted content-type for this request method is ‘application/json’.

Response Examples

- Datastore will automatically generate an ID and store it with the entity being created. This value will be sent in the response body as shown in the example.
- The value of the attribute `carrier` is the id of the bike currently carrying the component. If the component is not currently on a bike, the value of `carrier` is ‘null’.
- The attribute `self` is added to the JSON response body but is not stored in Datastore. The value of this attribute is a self link that directs users to the location of that resource.

Success

```
Status: 201 Created
{
    "manufacturer": "SRAM",
    "description": "shifter",
    "condition": 5,
    "carrier": null,
    "id": 5122084296458240,
    "self": "http://127.0.0.1:8080/components/5122084296458240"
}
```

Failure

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

```
Status: 405 Method Not Allowed
<!doctype html>
<html lang=en>
<title>405 Method Not Allowed</title>
<h1>Method Not Allowed</h1>
<p>The method is not allowed for the requested URL.</p>
```

```
Status: 406 Not Acceptable
{
    "code": "Not Acceptable",
    "description": "The chosen media type is not supported for this request"
}
```

```
Status: 415 Unsupported Media Type
{
    "code": "Unsupported Media Type",
    "description": "Content type for this request must be application/json"
}
```

View a Component

Allows you to get an existing component

```
GET /components/:component_id
```

Protected:

No: A valid JWT is not required to view a component entity.

Request

Path Parameters

Name	**Description**
component_id	ID of the component

Request Body

None

Response

Response Body Format

JSON

Response Statuses

Outcome	**Status Code**	**Notes**
Success	200 OK	
Failure	404 Not Found	No component with this component_id exists
Failure	405 Method Not Allowed	If the request method is not one of the allowed request methods (PUT, PATCH, DELETE, GET), the request will not be fulfilled, and 405 status code is returned.

Response Examples

- The value of the attribute `carrier` is the id of the bike currently carrying the component. If the component is not currently on a bike, the value of `carrier` is ‘null’.
- The attribute `self` is added to the JSON response body but is not stored in Datastore. The value of this attribute is a self link that directs users to the location of that resource.

Success

```
Status: 200 OK
{
   "condition": 5,
   "carrier": null,
   "description": "shifter",
   "manufacturer": "SRAM",
   "id": 5122084296458240,
   "self": "http://127.0.0.1:8080/components/5122084296458240"
}
```

Failure

```
Status: 404 Not Found
{
   "code": "Not Found",
   "description": “No component with this component_id exists"
}
```

```
Status: 405 Method Not Allowed
<!doctype html>
<html lang=en>
<title>405 Method Not Allowed</title>
<h1>Method Not Allowed</h1>
<p>The method is not allowed for the requested URL.</p>
```

Modify a Component (PUT)

Allows you to edit all attributes of a component.

```
PUT /components/:component_id
```

Protected:

No: A valid JWT is not required to modify a component entity.

Request

Path Parameters

Name	Description
component_id	ID of the component

Request Body

Required

Request Body Format

JSON

Request MIME Type

application/json

Request JSON Attributes

Name	Description	Required?
manufacturer	The manufacturer of the component.	Yes
description	A description of the component. E.g., tires, pedals, etc.	Yes
condition	The condition of the component on a 1 to 5 scale.	Yes

Request Body Example: Successful

```
{
   "condition": 1
}
```

Request Body Examples: Failure

```
{
   "manufacturer": "SRAM",
   "description": "tires",
   "condition": “three”
}
```

```
{
}
```

```
<table>
        <tr>
                <th>manufacturer</th>
                <td>SRAM</td>
        </tr>
        <tr>
                <th>description</th>
                <td>tires</td>
        </tr>
</table>
```

Response

Response Body Format

Success: No body

Failure: JSON

Response MIME Type

application/json

Response Statuses

Outcome	**Status Code**	**Notes**
Success	204 No Content	
Failure	400 Bad Request	The request object is missing at least one of the required attributes.
Failure	404 Not Found	No component with this component_id exists.
Failure	405 Method Not Allowed	If the request method is not one of the allowed request methods (PUT, PATCH, DELETE, GET), the request will not be fulfilled, and 405 status code is returned.
Failure	406 Not Acceptable	If the request only accepts MIME-types that are not supported by the server, the component is not created, and 406 status code is returned. Accepted content-type for this request method is ‘application/json’.
Failure	415 Unsupported Media Type	If the request is a content-type that is not supported by the server, the component is not created, and 415 status code is returned. Accepted content-type for this request method is ‘application/json’.

Response Examples

- Datastore will automatically generate an ID and store it with the entity being created. This value will be sent in the response body as shown in the example.
- The value of the attribute `carrier` is the id of the bike currently carrying the component. If the component is not currently on a bike, the value of `carrier` is ‘null’.
- The attribute `self` is added to the JSON response body but is not stored in Datastore. The value of this attribute is a self link that directs users to the location of that resource.

Success

```
Status: 204 No Content
```

Failure

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

```
Status: 404 Not Found
{
    "code": "Bad Request",
    "description": "No component with this component_id exists"
}
```

```
Status: 405 Method Not Allowed
<!doctype html>
<html lang=en>
<title>405 Method Not Allowed</title>
<h1>Method Not Allowed</h1>
<p>The method is not allowed for the requested URL.</p>
```

```
Status: 406 Not Acceptable
{
    "code": "Not Acceptable",
    "description": "The chosen media type is not supported for this request"
}
```

```
Status: 415 Unsupported Media Type
{
    "code": "Unsupported Media Type",
    "description": "Content type for this request must be application/json"
}
```

Modify a Component (PATCH)

Allows you to edit selected attributes of a component.

```
Patch /components/:component_id
```

Protected:

No: A valid JWT is not required to modify a component entity.

Request

Path Parameters

Name	Description
component_id	ID of the component

Request Body

Required

Request Body Format

JSON

Request MIME Type

application/json

Request JSON Attributes

Name	Description	Required?
manufacturer	The manufacturer of the component.	No
description	A description of the component. E.g., tires, pedals, etc.	No
condition	The condition of the component on a 1 to 5 scale.	No

Request Body Example: Successful

```
{
   "manufacturer": "SRAM",
   "description": "tires",
   "condition": 3
}
```

Request Body Examples: Failure

```
{
    "manufacturer": "SRAM",
    "description": "tires",
    "condition": “three”
}
```

```
{
    "manufacturer": "SRAM",
    "description": "tires"
}
```

```
<table>
        <tr>
                <th>manufacturer</th>
                <td>SRAM</td>
        </tr>
        <tr>
                <th>description</th>
                <td>tires</td>
        </tr>
        <tr>
                <th>condition</th>
                <td>5</td>
        </tr>
</table>
```

Response

Response Body Format

Success: No body

Failure: JSON

Response MIME Type

application/json

Response Statuses

Outcome	**Status Code**	**Notes**
Success	204 No Content	
Failure	404 Bad Request	No component with this component_id exists.
Failure	405 Method Not Allowed	If the request method is not one of the allowed request methods (POST, GET), the request will not be fulfilled, and 405 status code is returned.
Failure	406 Not Acceptable	If the request only accepts MIME-types that are not supported by the server, the component is not created, and 406 status code is returned. Accepted content-type for this request method is ‘application/json’.
Failure	415 Unsupported Media Type	If the request is a content-type that is not supported by the server, the component is not created, and 415 status code is returned. Accepted content-type for this request method is ‘application/json’.

Response Examples

- Datastore will automatically generate an ID and store it with the entity being created. This value will be sent in the response body as shown in the example.
- The value of the attribute `carrier` is the id of the bike currently carrying the component. If the component is not currently on a bike, the value of `carrier` is 'null'.
- The attribute `self` is added to the JSON response body but is not stored in Datastore. The value of this attribute is a self link that directs users to the location of that resource.

Success

```
Status: 204 No Content
```

Failure

```
Status: 404 Not Found
{
   "code": "Bad Request",
   "description": "No component with this component_id exists"
}
```

```
Status: 405 Method Not Allowed
<!doctype html>
<html lang=en>
<title>405 Method Not Allowed</title>
<h1>Method Not Allowed</h1>
<p>The method is not allowed for the requested URL.</p>
```

```
Status: 406 Not Acceptable
{
   "code": "Not Acceptable",
   "description": "The chosen media type is not supported for this request"
}
```

```
Status: 415 Unsupported Media Type
{
   "code": "Unsupported Media Type",
   "description": "Content type for this request must be application/json"
}
```

Delete a Component

Allows you to delete a component.

```
DELETE /component/:component_id
```

Protected:

No: A valid JWT is not required to delete a component entity.

Request

Path Parameters

Name	Description
component_id	ID of the component

Request Body

None

Response

No body

Response Body Format

Success: No body

Failure: JSON

Response MIME Type

application/json

Response Statuses

Outcome	Status Code	Notes
Success	204 No Content	
Failure	404 Not Found	No component with this component_id exists

Response Examples

Success

```
Status: 204 No Content
```

Failure

```
Status: 404 Not Found
{
   "code": "Bad Request",
   "description": "No component with this component_id exists"
}
```

Add Component to a Bike

Assigns a specified component to a specified bike

```
PUT /bikes/:bike_id/components/:component_id
```

Protected:

Yes: A valid JWT is required to add a component to a bike entity.

Request

Path Parameters

Name	**Description**
component_id	ID of the component
bike_id	ID of the bike

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 bike exists with this bike_id, a component exists with this component_id and the component is not already installed on another bike.
Failure	403 Forbidden	The component is already installed on another bike.
Failure	404 Not Found	The specified bike and/or component does not exist
Failure	405 Method Not Allowed	If the request method is not one of the allowed request methods (PUT, DELETE), the request will not be fulfilled, and 405 status code is returned.

Response Examples

Success

```
Status: 204 No Content
```

Failure

```
Status: 403 Forbidden
{
    "code": "Forbidden",
    "description": "The components is already installed on another bike"
}
```

```
Status: 404 Not Found
{
    "code": “Not Found",
    "description": "The specified bike and/or component does not exist"
}
```

```
Status: 405 Method Not Allowed
<!doctype html>
<html lang=en>
<title>405 Method Not Allowed</title>
<h1>Method Not Allowed</h1>
<p>The method is not allowed for the requested URL.</p>
```

Remove Component from a Bike

Remove a component from the bike carrying it.

```
DELETE /bikes/:bike_id/components/:component_id
```

Protected:

Yes: A valid JWT is required to remove a component from a bike entity.

Request

Path Parameters

Name	Description
component_id	ID of the component
bike_id	ID of the bike

Request Body

None

Response

No body

Response Body Format

Success: No body

Failure: JSON

Response Body Format

Success: No body

Failure: JSON

Response Statuses

Outcome	Status Code	Notes
Success	204 No Content	Succeeds only if a bike exists with this bike_id, a component exists with this component_id and the component is currently being carried by the bike with bike_id.
Failure	404 Not Found	The specified bike and/or component does not exist
Failure	405 Method Not Allowed	If the request method is not one of the allowed request methods (PUT, DELETE), the request will not be fulfilled, and 405 status code is returned.

Response Examples

Success

```
Status: 204 No Content
```

Failure

```
Status: 404 Not Found
{
    "code": “Not Found",
    "description": "No bike with this bike_id or component with this component_id exists"
}
```

```
Status: 405 Method Not Allowed
<!doctype html>
<html lang=en>
<title>405 Method Not Allowed</title>
<h1>Method Not Allowed</h1>
<p>The method is not allowed for the requested URL.</p>
```