URL: /artworks
HTTP Method: GET
Description:
Searches artworks on the database.
Supported Query Parameters:

- name (string) – an artwork will match if the name property contains the given string
- artist name (string) - artwork will match if the artist name property contains the given string
- category keyword (string) - artwork will match if the category property contains the given keyword string
- limit (number) - the number of results to return (default 10, minimum 1, maximum 20)

Expected Request Body Data:
None
Supported Response Data Types:

- text/html
- application/json

Expected Response (Success):

- Status code of 200.
- If text/html is requested: An HTML page containing a list of up to *limit* matching artworks. Each of the artworks in the list will have a hyperlink to redirect to that artworks page.
- If application/json is requested: An array containing the object representation for each matching artwork.

URL: /artworks/{artID}
HTTP Method: GET
Description:
Retrieves the artwork with unique ID of artID if it exists.
URL Parameters:

- artID – the unique artwork ID to get

Expected Request Body Data:
None
Supported Response Data Types:

- text/html
- application/json

Expected Response (Success):

- Status code of 200.
- If text/html is requested: An HTML page showing all the artworks info: name, artist, year, category, medium description, and image. As well as the reviews and number of likes the matching artwork has received
- If application/json is requested: An object of the matching artwork.

URL: /artworks

HTTP Method: POST

Description:

Creates a new artwork using request data

Expected Request Body Data:

JSON string with the structure {“name”:“Artwork name”, “artist”: “Artist name”, “year”: “year the artwork was created”, “category”: “photograph, sculpture, etc.”, “medium”: “wood, silk, etc.”, “description”: “information about the artwork.”, “image”: “link to image”, “reviews”: “user entered reviews”, “likes”: “number of likes the artwork received”}.

Supported Response Data Types:

- application/json

Expected Response (Success):

- Status code of 201 (Created).
- JSON string representing the newly created artwork object.

URL: /artworks/{artID}

HTTP Method: PUT

Description:

Updates the artwork with the given artwork ID, if they exist and the request data is valid. For eachproperty in the JSON string contained in the request that is valid, the artwork’s data associated with that property will be updated.

URL Parameters:

- artID – the unique artwork ID to get

Expected Request Body Data:

A JSON string containing at least 1 property that match properties of artwork objects in the database. An example of the structure would be: {“reviews”:“user entered review”}

Supported Response Data Types:

- application/json

Expected Response (Success):

- Status code of 200
- JSON string of the modified artwork object.

URL: /artworks/{artID}

HTTP Method: DELETE

Description:

Deletes the artwork with the given artwork ID if they exist.

URL Parameters:

- artID – the unique artwork ID to get

Expected Response (Success):

- Status code of 200

URL: /users

HTTP Method: POST

Description:

Creates a new user using request data if the username given is unique

Expected Request Body Data:

JSON string with the structure {“username” :“username”, “password” : “password”, “account type”: “patron”}.

This is then used for creation of username and password for the new user patron.

Supported Response Data Types:

- application/json

Expected Response (Success):

- Status code of 201 (Created).
- JSON string representing the newly created user object.

URL:/users/{userID}

HTTP Method: GET

Description:

Gets the user with unique ID of userID if they exist.

URL Parameters:

- userID – the unique user ID to get

Expected Request Body Data:

None

Supported Response Data Types:

- text/html
- application/json

Expected Response (Success):

- Status code of 200.
- If text/html is requested: An HTML page for the given user, formatting of page depending on if the user is a patron or an artist.
- If application/json is requested: An object of the given user.

URL: /user/{userID}
HTTP Method: PUT
Description:
Updates the user with the given user ID, if they exist and the request data is valid. For each property in the JSON string contained in the request that is valid, the user's data associated with that property will be updated.
URL Parameters:

- userID – the unique user ID to get

Expected Request Body Data:
A JSON string containing at least 1 property that match properties of user objects in the database. An example of the structure would be: {"account type":"type"}. Once a user is upgraded to an artist we will add property's to their user object such as {"workshops": {"name": "name of workshop", "enrolled": {list of enrolled users}}(as an example will add more propertys) and {"artworks": {list of artworks}}.
Supported Response Data Types:

- application/json

Expected Response (Success):

- Status code of 200
- JSON string of the modified user object.

URL: /users/{userID}
HTTP Method: DELETE
Description:
Deletes the user with the given user ID if they exist.
URL Parameters:

- userID – the unique user ID to get

Expected Response (Success):

- Status code of 200

URL:/users/{userID}/add
artwork

HTTP Method: GET
Description:
An HTML page to allow the artist user to input data to add an artwork object
URL Parameters:

- userID – the unique user ID to get

Expected Request Body Data:
None
Supported Response Data Types:

- text/html

Expected Response (Success):

- Status code of 200.
- Render of the add an artwork HTML page

URL:/users/{userID}/add
workshop

HTTP Method: GET
Description:
An HTML page to allow the artist user to input data to add a workshop object
URL Parameters:

- userID – the unique user ID to get

Expected Request Body Data:
None
Supported Response Data Types:

- text/html

Expected Response (Success):

- Status code of 200.
- Render of the add a workshop HTML page

URL: /workshop
HTTP Method: POST
Description:
Creates a new workshop using request data
Expected Request Body Data:
JSON string with the structure {“name”:“title of workshop”}
Supported Response Data Types:

- application/json

Expected Response (Success):

- Status code of 201 (Created).
- JSON string representing the newly created workshop object.

URL: /workshop/{workID}
HTTP Method: DELETE
Description:
Deletes the workshop with the given workshop ID if they exist.
URL Parameters:

- workID – the unique workshop ID to get

Expected Response (Success):

- Status code of 200

URL: /workshop/{userID}
HTTP Method: GET
Description:
Retrieves all the workshops with artist’s user ID if it exists
URL Parameters:

- userID – the unique artwork ID to use to get list of workshops

Expected Request Body Data:
None
Supported Response Data Types:
- text/html

Expected Response (Success):
- Status code of 200.
- An HTML page showing all the workshops of the given artist

URL: /workshop/{workID}
HTTP Method: GET
Description:
Retrieves the workshop with unique ID of work ID if it exists.
URL Parameters:
- workID – the unique workshop ID to get

Expected Request Body Data:
None
Supported Response Data Types:
- text/html
- application/json

Expected Response (Success):
- Status code of 200.
- If text/html is requested: An HTML page showing all the workshops info: name and list of enrolled users
- If application/json is requested: An object of the matching workshop.

URL:
/workshop/{workID}
HTTP Method: PUT
Description:
Updates the workshop with the given work ID, if they exist and the request data is valid. For each property in the JSON string contained in the request that is valid, the workshop's data associated with that property will be updated.
URL Parameters:
- workID – the unique artwork ID to get

Expected Request Body Data:
A JSON string containing at least 1 property that match properties of artwork objects in the database. An example of the structure would be: {"name":"title of workshop"}
Supported Response Data Types:
- application/json

Expected Response (Success):
- Status code of 200
- JSON string of the modified workshop object.