Assignment 6: API Spec

CS 493: Cloud Application Development

Oregon State University

Change Log

Version	Change	Date
1.0	Initial version.	May 2, 2024
2.0	For endpoint 12 “Update enrollment in a course” the body of the response should be empty on success. Updated error in the “Success Response Examples” section for this endpoint to remove the body from the example response.	May 22, 2024
3.0	For endpoint 7 “Create a Course” fixed error in description of when to send back 400 status code	May 23, 2024
4.0	For endpoint 12 “Update enrollment in a course” updated the description of 409 status code to clarify that 401 and 403 take precedence over 409.	May 25, 2024

Summary of All Endpoints

	Functionality	Endpoint	Protection	Description
1.	User login	POST /users/login	Pre-created Auth0 users with username and password	Use Auth0 to issue JWTs.
2.	Get all users	GET /users	Admin only	Summary information of all 9 users. No info about avatar or courses.
3.	Get a user	GET /users/:id	Admin. Or user with JWT matching id	Detailed info about the user, including avatar (if any) and courses (for instructors and students)
4.	Create/update a user’s avatar	POST /users/:id/avatar	User with JWT matching id	Upload file to Google Cloud Storage
5.	Get a user’s avatar	GET /users/:id/avatar	User with JWT matching id	Read and return file from Google Cloud Storage
6.	Delete a user’s avatar	DELETE /users/:id/avatar	User with JWT matching id	Delete file from Google Cloud Storage.
7.	Create a course	POST /courses	Admin only	Create a course.
8.	Get all courses	GET /courses	Unprotected	Paginated using offset/limit. Page size is 3. Ordered by “subject.” Doesn’t return info on course enrollment.
9.	Get a course	GET /course/:id	Unprotected	Doesn’t return info on course enrollment.
10.	Update a course	PATCH /course/:id	Admin only	Partial update.
11.	Delete a course	DELETE /course/:id	Admin only	Delete course and delete enrollment info about the course.
12.	Update enrollment in a course	PATCH /courses/:id/students	Admin. Or instructor of the course.	Enroll or disenroll students from the course.
13.	Get enrollment for a course	GET /courses/:id/students	Admin. Or instructor of the course.	All students enrolled in the course.

Note on Error Message for Error Status Codes

For the following response status codes, the error message must always be as follows (regardless of the endpoint)

Response Status Code	**JSON Error Message**
400	{"Error": "The request body is invalid"}
401	{"Error": "Unauthorized"}
403	{"Error": "You don't have permission on this resource"}
404	{"Error": "Not found"}

1. User Login

Generate a JWT for a registered user of the app by sending a request to Auth0 domain created for the REST API to get a token.

Note: You need to pre-create 9 users in Auth0 as described in the assignment's section "Users"

```
POST /users/login
```

Request

Path Parameters

None

Request Body

Required

Request Body Format

JSON

Request JSON Attributes

Name	Description	Required?
username	Username	Yes
password	Password	Yes

Request Body Example

```
{
  "username": "admin@osu.com",
  "password": "Cheese1234!"
}
```

Response

Response Body Format

JSON

Response Statuses

Outcome	Status Code	Notes
Success	200	Only one property, token, whose value is the JWT
Failure	400	Request body is invalid
Failure	401	Username and/or password is incorrect

Success Response Examples

Status: 200

```
{
   "token":
"eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCIsImtpZCI6IllLT3g5S1NXNDdmOGlUZTJHSkxQOCJ9.eyJuaWNr
bmFtZSI6IndhbGxhY2UiLCJuYW1lIjoid2FsbGFjZUBjaGVlc2UuY29tIiwicGljdHVyZSI6Imh0dHBzOi8vcy
5ncmF2YXRhci5jb20vYXZhdGFyL2Q2NjBhMjE4ODNhNGNjYjM3OWFkMWNlMWIxOTg0M2U5P3M9NDgwJnI9cGcm
ZD1odHRwcyUzQSUyRiUyRmNkbi5hdXRoMC5jb20lMkZhdmF0YXJzJTJGd2EucG5nIiwidXBkYXRlZF9hdCI6Ij
IwMjQtMDUtMDdUMTU6MDc6NDIuMTM4WiIsImVtYWlsIjoid2FsbGFjZUBjaGVlc2UuY29tIiwiZW1haWxfdmVy
aWZpZWQiOmZhbHNlLCJpc3MiOiJodHRwczovLzQ5My0yNC1zcHJpbmcudXMuYXV0aDAuY29tLyIsImF1ZCI6Ik
hKN1hYMHZiQ01wVHdqVmxjNmtqeG1EbUJRc3NkRFNqIiwiaWF0IjoxNzE1MDk0NDYyLCJleHAiOjE3MTUxMzA0
NjIsInN1YiI6ImF1dGgwfDY2MzlhMGMyMTA0NWRkMDY5ZmJiZjJjMCJ9.TGejoA4LACP0d62GoLrNW8x08ahUk
YHKSGBMusgWwriwvxhaO7uIYgXRS-GRdFbiRhPnKIXyZqLGrXbsHa3LLS0FGCQlNLMlPX8Vw3yLL2OzGL-
ceoJ2cQXM9shVu81HTRsXvLfZnc-UjDNsVqPBsphKMv2NIBqxmBGdF5i8dEfGCpoZJB8IlKEN-
ElllWBisYOJa88dg2LGFzT0uLZRntab1vHv-O12ojhoRC-
QR34pjikn7o48x1P0D0kwkyWszCndXWc11sj0q8u16OmXqtBFIEUVJ6CRor0yWZD9b8mphW0SD1UebcEiP-
38KSkWJh8inRbcnynxfj0KGUgizA"
}
```

2. Get all users

Return an array with all 9 pre-created users from the kind “users” in Datastore.

Each user in the response must have exactly 3 properties: “id”, “role” and “sub” properties

- If you designed the kind “users” to have additional properties, those properties must not be included in the response.

Protection

Only users with the role admin.

```
GET /users
```

Request

Header

The JWT as a Bearer token in the Authorization header

Path Parameters

None

Request Body

None

Request Body Format

Response

Response Body Format

JSON

Response Statuses

Outcome	Status Code	Notes
Success	200	
Failure	401	The JWT is missing or invalid
Failure	403	The JWT is valid but doesn’t belong to an admin.

Success Response Examples

Status: 200

```
[
    {
        "id": 5631671361601536,
        "role": "student",
        "sub": "auth0|664384d7829d72375c7a034d"
    },
    {
        "id": 5632499082330112,
        "role": "instructor",
        "sub": "auth0|664383f2ad88a0630023ab9b"
    },
    {
        "id": 5634161670881280,
        "role": "admin",
        "sub": "auth0|65850deddca1222f3d038b98"
    },
    {
        "id": 5636645067948032,
        "role": "student",
        "sub": "auth0|664384a84d1c357206d6e3c5"
    },
    {
        "id": 5642368648740864,
        "role": "student",
        "sub": "auth0|6583adc15f31ac05f02ea87a"
    },
    {
        "id": 5644004762845184,
        "role": "instructor",
        "sub": "auth0|6583ae12895d09a70ba1c7c5"
    },
    {
        "id": 5646488461901824,
        "role": "student",
        "sub": "auth0|664384ca829d72375c7a0337"
    },
    {
        "id": 5712837116690432,
        "role": "student",
        "sub": "auth0|664384f1cddc69d8a1196941"
    },
    {
        "id": 5714489739575296,
        "role": "student",
        "sub": "auth0|6643848dcddc69d8a11968d4"
    }
]
```

3. Get a user

Return the details of a user.

Protection

User with admin role. Or when JWT is owned by user_id in the path parameter.

GET /users/:user_id

Request

Header

The JWT as a Bearer token in the Authorization header

Path Parameters

Name	Description
user_id	ID of the user

Request Body

None

Request Body Format

Response

Response Body Format

JSON

The response will always include the 3 properties “id”, “role” and “sub”.

Regardless of their role, if the user has an avatar, the body must also include the property “avatar_url” which is the link to get the avatar for the user. However, if the user doesn’t have an avatar, then the body must not include the property “avatar_url.”

If the user role is either “instructor” or “student” the response must always include the property “courses”

- The value of this property must be an array
- If the users’ role is “instructor” then the arrays has links to all the courses the instructor is teaching.
    - The array will be empty if the instructor is not teaching any courses.
- If the user’s role is “student” then the array has links to all the courses the student is enrolled in.
    - The array will be empty if the student is not enrolled in any courses.
- If the user role is “admin”, then the response must not have the property “courses.”

Response Statuses

Outcome	Status Code	Notes
Success	200	
Failure	401	The JWT is missing or invalid, regardless of whether the user exists or not.
Failure	403	The JWT is valid, but the user doesn’t exist. The JWT is valid, and the user exists, but the JWT doesn’t

		belong to either an admin or to the user whose ID is in the path parameter.

Success Response Examples

(instructor with an avatar, teaching 2 courses)
Status: 200

```
{
    "avatar_url": "http://localhost:8080/users/5644004762845184/avatar",
    "courses": [
        "http://localhost:8080/courses/5744039651442688",
        "http://localhost:8080/courses/5759318150348800"
    ],
    "id": 5644004762845184,
    "role": "instructor",
    "sub": "auth0|6583ae12895d09a70ba1c7c5"
}
```

(instructor without an avatar, teaching 0 courses)
Status: 200

```
{
    "courses": [
    ],
    "id": 5632499082330112,
    "role": "instructor",
    "sub": "auth0|664383f2ad88a0630023ab9b"
}
```

(admin user without an avatar)
Status 200

```
{
    "id": 5634161670881280,
    "role": "admin",
    "sub": "auth0|65850deddca1222f3d038b98"
}
```

4. Create/update a user’s avatar

Upload the .png in the request as the avatar of the user’s avatar. If there is already an avatar for the user, it gets updated with the new file. The file must be uploaded to Google Cloud Storage.

Protection

JWT is owned by user_id in the path parameter.

POST /users/:user_id/avatar

Request

Header

The JWT as a Bearer token in the Authorization header

Path Parameters

Name	**Description**
user_id	ID of the user

Request Body

Required

Request Body Format

Form-data with one required key “file”

- You can assume that if there is a file with the request, its extension will be “.png”

Response

Response Body Format

JSON

Response Statuses

Outcome	**Status Code**	**Notes**
Success	200	
Failure	400	The request doesn’t include the key “file.” 400 status code takes precedence over 401 and 403, i.e., if the request doesn’t include the key “file” return 400 regardless of whether the JWT is missing or valid.
Failure	401	The JWT is missing or invalid.
Failure	403	The JWT is valid but doesn’t belong to the user whose ID is in the path parameter.

Success Response Examples

Status: 200

```
{
    "avatar_url": "http://localhost:8080/users/5644004762845184/avatar"
}
```

5. Get a user’s avatar

Return the file stored in Google Cloud Storage as the user’s avatar.

Protection

JWT is owned by user_id in the path parameter.

```
GET /users/:user_id/avatar
```

Request

Header

The JWT as a Bearer token in the Authorization header

Path Parameters

Name	**Description**
user_id	ID of the user

Request Body

None

Response

On success, the file which is this user’s avatar.

Response Statuses

Outcome	**Status Code**	**Notes**
Success	200	
Failure	401	The JWT is missing or invalid.
Failure	403	The JWT is valid but doesn’t belong to the user whose ID is in the path parameter.
Failure	404	The JWT is valid, belongs to the user whose ID is in the path parameter, but the user doesn’t have an avatar.

Success Response Examples

```
Status: 200
Note: The body has the file.
```

6. Delete a user’s avatar

Delete the file stored in Google Cloud Storage as the user’s avatar.

Protection

JWT is owned by user_id in the path parameter.

DELETE /users/:user_id/avatar

Request

Header

The JWT as a Bearer token in the Authorization header

Path Parameters

Name	**Description**
user_id	ID of the user

Request Body

None

Response

None on success. JSON on failure.

Response Statuses

Outcome	**Status Code**	**Notes**
Success	204	
Failure	401	The JWT is missing or invalid.
Failure	403	The JWT is valid but doesn’t belong to the user whose ID is in the path parameter.
Failure	404	The JWT is valid, belongs to the user whose ID is in the path parameter, but the user doesn’t have an avatar.

Success Response Examples

Status: 204 *No body*

7. Create a course

Create a course.

- The “instructor_id” in the request body must match an instructor in the “users” kind in Datastore.

Protection

Only users with the role admin.

```
POST /courses
```

Request

Header

The JWT as a Bearer token in the Authorization header

Path Parameters

None

Request Body

Required

Request Body Format

JSON

Request JSON Attributes

Name	Description	Required?
subject	String. Subject code Up to 4 characters	Yes
number	Integer	Yes
title	String. Course title. Up to 50 characters.	Yes
term	String. Up to 10 characters.	Yes
instructor_id	Integer. The instructor assigned to teach the course.	Yes

Request Body Example

```
{
  "subject": "CS",
  "number": "493",
  "title": "Cloud Application Development",
  "term": "fall-24",
  "instructor_id": 5644004762845184
}
```

Response

Response Body Format

JSON

Response Statuses

Outcome	Status Code	Notes
Success	201	
Failure	400	If the request is missing any of the attributes or the value

		of instructor_id doesn't correspond to the id of an instructor, the course must not be created, and 400 status code must be returned. You don't need to validate the values of the attributes (other than instructor_id) and can assume that if the request contains any of the listed attributes, then that attribute's value is valid. You can also assume that the request will not contain any extraneous attribute (i.e., the request JSON will never contain any attribute that is not listed above) Check the body only when the JWT is valid and the JWT belongs to an admin ~~and a course with this ID exists~~, i.e., 401 and 403 status code take precedence over 400.
Failure	401	The JWT is missing or invalid.
Failure	403	The JWT is valid but doesn't belong to an admin.

Success Response Examples

Success

Status: 201 Created

```
{
    "id": 5710353417633792,
    "instructor_id": 5644004762845184,
    "number": "493",
    "self": "http://localhost:8080/courses/5710353417633792",
    "subject": "CS",
    "term": "fall-24",
    "title": "Cloud Application Development"
}
```

8. Get all courses

Paginated list of the courses.

- This list should be paginated using limit/offset based paging with page size of 3.
- The list must be sorted by the “subject” property. Within a subject, sorting is not required.
- The courses returned should not contain the list of students enrolled in the course.

Protection

Unprotected.

```
GET /courses
GET /courses?offset=3&limit=3
…
```

Request

Path Parameters

None

Query Parameters

- 2 optional query parameters. Note that the order of query parameters doesn’t matter.
- You can assume the following:
  - A request will either have both parameters or neither.
  - The value of both will be an integer. You don’t need to validate the value.

Name	Description
offset	Offset.
limit	Limit. Value will be 3

Request Body

None

Response

Response Body Format

JSON

- This endpoint implements pagination with a page size of 3.
- The property “courses” is an array with the courses for this page.
- The property “next” is a URL with query parameters offset and limit set to correct values.
  - Optional: If a page has fewer than 3 entries then it is certainly the last page and it should not have the “next” link (this is optional to implement and we are not going to deduct points for it)

Response Statuses

Outcome	Status Code	Notes
Success	200	

Success Response Examples

Status: 200 OK

```
{
    "courses": [
        {
            "id": 5633378543992832,
            "instructor_id": 5644004762845184,
            "number": 493,
            "self": "http://localhost:8080/courses/5633378543992832",
            "subject": "CS",
            "term": "fall-24",
            "title": "Cloud Application Development"
        },
        {
            "id": 5640825748848640,
            "instructor_id": 5644004762845184,
            "number": 492,
            "self": "http://localhost:8080/courses/5640825748848640",
            "subject": "CS",
            "term": "fall-24",
            "title": "Mobile App Development"
        },
        {
            "id": 5706627130851328,
            "instructor_id": 5632499082330112,
            "number": 344,
            "self": "http://localhost:8080/courses/5706627130851328",
            "subject": "CS",
            "term": "fall-24",
            "title": "OS 1"
        }
    ],
    "next": "http://localhost:8080/courses?limit=3&offset=3"
}
```

9. Get a course

Allows you to get an existing course. The response must not include the list of students enrolled in the course.

Protection

Unprotected.

GET /courses/:course_id

Request

Path Parameters

Name	**Description**
course_id	ID of the course

Request Body

None

Response

Response Body Format

JSON

Response Statuses

Outcome	**Status Code**	**Notes**
Success	200	
Failure	404	No course with this ID exists.

Success Response Examples

Status: 200

```
{
    "id": 5667525748588544,
    "instructor_id": 5644004762845184,
    "number": 493,
    "self": "http://localhost:8080/courses/5667525748588544",
    "subject": "CS",
    "term": "fall-24",
    "title": "Cloud Application Development"
}
```

10. Update a course

Performs a partial update on the course.

- Student enrollment cannot be modified via this endpoint.

Protection

Only users with the role admin.

```
PATCH /courses/:course_id
```

Request

Header

The JWT as a Bearer token in the Authorization header

Path Parameters

Name	**Description**
course_id	ID of the course

Request Body

Required

Request Body Format

JSON

Request JSON Attributes

Name	**Description**	**Required?**
subject	String. Subject code Up to 4 characters	No
number	Integer	No
title	String. Course title. Up to 50 characters.	No
term	String. Up to 10 characters.	No
instructor_id	Integer.	No

Request Body Example

```
{
  "instructor_id": 5644004762845184
}
```

- Any property which is not specified in the request must remain unchanged.
- A request with a JSON object with no properties is valid. In this case, nothing in the course gets updated.

Response

Response Body Format

JSON

Success: The course with the updated values.

Failure: Error message.

Response Statuses

Outcome	**Status Code**	**Notes**
Success	200	
Failure	400	If the request contains the property instructor_id, but the value of instructor_id doesn't correspond to the id of an instructor, the course must not be updated, and 400 status code must be returned. You don't need to validate the values of the attributes (other than instructor_id) and can assume that if the request contains any of the listed attributes, then that attribute's value is valid. You can also assume that the request will not contain any extraneous attribute (i.e., the request JSON will never contain any attribute that is not listed above) Check the body only when the JWT is valid, the JWT belongs to an admin and a course with this ID exists, i.e., 401 and 403 status code take precedence over 400.
Failure	401	The JWT is missing or invalid, regardless of whether a course with this ID exists or not.
Failure	403	The JWT is valid, but the course doesn't exist. The JWT is valid, and the course exists, but the JWT doesn't belong to an admin.

Success Response Examples

Status: 200

```
{
    "id": 5710353417633792,
    "instructor_id": 5644004762845184,
    "number": "493",
    "self": "http://localhost:8080/courses/5710353417633792",
    "subject": "CS",
    "term": "fall-24",
    "title": "Cloud Application Development"
}
```

11. Delete a course

Delete a course. Also deletes enrollment of all students that were enrolled in the course.

Protection

Only users with the role admin.

```
DELETE /courses/:course_id
```

Request

Header

The JWT as a Bearer token in the Authorization header

Path Parameters

Name	Description
course_id	ID of the course

Request Body

None

Response

No body

Response Body Format

Success: No body

Failure: JSON

Response Statuses

Outcome	Status Code	Notes
Success	204	
Failure	401	The JWT is missing or invalid, regardless of whether a course with this ID exists or not.
Failure	403	The JWT is valid, but the course doesn't exist. The JWT is valid, and the course exists, but the JWT doesn't belong to an admin.

Success Response Examples

Status: 204

No body

12. Update enrollment in a course

Enroll and/or disenroll students from a course.

Protection

User with admin role. Or when JWT is owned by the instructor of this course.

PATCH /courses/:course_id/students

Request

Header

The JWT as a Bearer token in the Authorization header

Path Parameters

Name	Description
course_id	ID of the course

Request Body

Required

Request Body Format

JSON

Request JSON Attributes

Name	Description	Required?
add	An array, possibly empty, containing student IDs for students to enroll in the course.	Yes
remove	An array, possibly empty, containing student IDs for students to be removed from the course.	Yes

- You can assume that
  - The request will always have these 2 properties
  - The values of these 2 properties will always be arrays.
  - Any elements in the array will be integers.
  - At least one of the arrays will be non-empty.

Request Body Examples

```
{
  "add": [
    5642368648740864,
    5714489739575296,
    5636645067948032,
    5646488461901824
   ],
   "remove": [
   ]
}
```

```
{
  "add": [
   ],
   "remove": [
    5714489739575296,
    5636645067948032,
    5646488461901824
   ]
}
```

```
{
  "add": [
    5642368648740864
   ],
   "remove": [
    5714489739575296,
    5636645067948032,
    5646488461901824
   ]
}
```

Response

Response Body Format

Success: Empty body.

Failure: Error message.

Response Statuses

Outcome	Status Code	Notes
Success	200	
Failure	401	The JWT is missing or invalid, regardless of whether a course with this ID exists or not.
Failure	403	The JWT is valid, but the course doesn't exist. The JWT is valid, and the course exists, but the JWT doesn't belong to either an admin or to the instructor of the course.
Failure	409	The enrollment data in the arrays "add" and/or "remove" is invalid. Enrollment data is valid if both of the following conditions is true: i) There is no common value between the arrays "add" and "remove" ii) All values in the array "add" and "remove" correspond to the ID of a user with the role "student" in the kind "users" If a student in the "add" array is already enrolled in the course, skip them. If a student in the "remove" array is not enrolled in the course, skip them. Neither of these

		cases, by themselves, make the enrollment data invalid. Check the body only when the JWT is valid, the course exists, and the JWT belongs to an admin and to the instructor of the course, i.e., 401 and 403 status code take precedence over 409.

Success Response Examples

Status: 200

Failure Response for 409

Status: 409

```
{"Error": "Enrollment data is invalid"}
```

13. Get enrollment for a course

Get the list of students enrolled in a course.

Protection

User with admin role. Or when JWT is owned by the instructor of this course.

```
GET /courses/:course_id/students
```

Request

Header

The JWT as a Bearer token in the Authorization header

Path Parameters

Name	Description
course_id	ID of the course

Request Body

None

Response Body Format

JSON

Response Statuses

Outcome	Status Code	Notes
Success	200	
Failure	401	The JWT is missing or invalid, regardless of whether a course with this ID exists or not.
Failure	403	The JWT is valid, but the course doesn't exist. The JWT is valid, and the course exists, but the JWT doesn't belong to either an admin or to the instructor of the course.

Success Response Examples

Status: 200

```
[
    5646488461901824,
    5631671361601536,
    5642368648740864
]
```