

SHARE2TEACH WEB APPLICATION API

© 2024 CodeCrafters. All Rights Reserved.



Septermber, 2024

CodeCrafters

This documentation is designed to assist developers in understanding and interacting with the RESTful API provided by the Share2Teach application. Built using **Spring Boot**, this API enables various operations like user registration, login, user management, and token-based authentication. The guide includes API request details, authentication mechanisms, and sample request formats for easy integration and usage.

**Abstract**

The API adheres to RESTful principles, meaning that it provides standard HTTP operations such as GET, POST, PUT and DELETE for interaction with resources. These operations allow the client to perform CRUD (Create, Read, Update, Delete) actions on resources like users. In this documentation, you will find:

|  |  |
| --- | --- |
| **Resource** | **NO.** |
| Section A - Details on making API requests using Postman | 3 |
| Section B - Explanation of key components of a REST API request and response | 4 |
| Section C - Instructions for JWT-based authentication | 5 |
| Section D - Error handling strategies | 6 |
| Section E - Sample code snippets demonstrating API usage | 6 |
| Conclusion | 8 |

**Section A**

**Making API Requests**

**Request URI**  
The Uniform Resource Identifier (URI) identifies the resource you want to interact with.

* Scheme: The protocol (e.g., https)
* Host: Domain or server hosting the API (e.g., https://api.share2teach.com)
* Path: Specific resource being addressed (e.g., /auth/users)
* Query Parameters: Optional parameters to filter or modify the request (e.g., ?filter=active)

**Sample code in bash**:

GET https://api.share2teach.com/auth/register?filter=active

**HTTP Method**  
This defines the action to be performed on the resource:

* GET – Retrieve data
* POST – Send data to create a resource
* PUT – Update an existing resource
* DELETE – Remove a resource

**Headers**  
This provides additional information such as Authorization tokens and content types.

**Request Body**  
Sent with POST or PUT requests in JSON format.

**Sample code in JSON**:

{

"name": "John King",

"email": "john@gmail.com",

“password": "pass”,

“role”: “ADMIN”

}

**Section B**

**Sample API Requests and Responses**

**GET Request**

To retrieve a list of users:

In bash:

curl -X GET "https://api.share2teach.com/auth/users" -H "Authorization: Bearer { token}"

This request returns a list of users in JSON format.

**POST Request**

To register a new user:

In bash:

curl -X POST "https://api.share2teach.com /auth/register" \

-H "Content-Type: application/json" \

-d '{

"name": "Susan King",

"email": "susan@gmail.com",

"password": "password",

"role": "USER"

}'

**PUT Request**

To update an existing user’s details:

In bash:

curl -X PUT "https://api.share2teach.com/auth/users/{id}" \

-H "Authorization: Bearer {token}" \

-H "Content-Type: application/json" \

-d '{"email": "john@outlook.com", "role": "ADMIN"}'

**DELETE Request**

To delete a user:

In bash:

curl -X DELETE "https://api.share2teach.com/auth/users/{id}" -H "Authorization: Bearer {token}"

**Section C**

**Authentication**

The API employs JWT (JSON Web Token) based authentication. Clients must first authenticate and obtain an access token, which must be included in the header of all subsequent requests.

To obtain a JWT token, use the /auth/login endpoint. Here’s an example of how to log in and obtain a token:

**Request** via Postman:

In bash

curl -X POST "https://api.share2teach.com/auth/login" \

-H "Content-Type: application/json" \

-d '{

"email": "user@example.com",

"password": "userpassword"

}'

**Response** code in JSON:

json

{

“statusCode”: 200,

"message": "Successfully Logged In"

"token": "eyJhbGciOiJIUzI1NiIsInR5...",

"refreshToken": "eyJhbGciOiJIUzI1NiIsInR5...",

"expirationTime": "24Hrs",

“role”: “USER”

}

**Using the Token**

Include the access token in the Authorization header for all protected routes:

In bash

-H "Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5..."

**Section D**

**Error Handling**

The API returns standard HTTP status codes to indicate the result of a request:

* 200 OK – Request was successful.
* 201 Created – A resource was successfully created.
* 400 Bad Request – The request was invalid or malformed.
* 401 Unauthorized – Authentication failed or token was not provided.
* 404 Not Found – The requested resource could not be found.
* 500 Internal Server Error – A server error occurred.

Sample Error Response in JSON:

{

"statusCode": 500,

"message": "Bad credentials"

}

**Section E**

**Sample Code – Using Spring Boot to Interact with the API**

Below is a Java-based Spring Boot controller that demonstrates how to handle API requests within the application:

import com.S2T.Share\_2\_Teach.dto.RequestResponse;

import com.S2T.Share\_2\_Teach.service.UsersManagementService;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.http.ResponseEntity;

import org.springframework.web.bind.annotation.\*;

@RestController

public class UserController {

@Autowired

private UsersManagementService usersManagementService;

// Registers a new user

@PostMapping("/auth/register")

public ResponseEntity<RequestResponse> register(@RequestBody RequestResponse reg) {

return ResponseEntity.ok(usersManagementService.register(reg));

}

// Handles user login requests

@PostMapping("/auth/login")

public ResponseEntity<RequestResponse> login(@RequestBody RequestResponse loginRequest) {

return ResponseEntity.ok(usersManagementService.login(loginRequest));

}

// Refreshes JWT tokens

@PostMapping("/auth/refresh")

public ResponseEntity<RequestResponse> refreshToken(@RequestBody RequestResponse refreshTokenRequest) {

return ResponseEntity.ok(usersManagementService.refreshToken(refreshTokenRequest));

}

// Retrieves all users

@GetMapping("/v1/users")

public ResponseEntity<RequestResponse> getAllUsers() {

return ResponseEntity.ok(usersManagementService.getAllUsers());

}

}

This sample demonstrates how various endpoints (like /auth/register, /auth/login) use service methods to handle user-related operations and return JSON responses.

**Conclusion**

This guide should give you a clear understanding of how to interact with the Share2Teach API. Whether you're registering users, logging in, or managing JWT tokens, the API's simplicity and adherence to REST principles ensure seamless integration with client applications. Utilize tools like Postman to test and interact with the API endpoints effectively.