CLP Developer Manual

API Documentation

❖ Lesson

Base URL: http://example.com/lessons

GET /lessons

- Method: GET
- Description: Retrieves all the lessons.
- Response: A Result object containing a list of Lesson objects.
- Response Status Codes:
    - 200 OK - Request successful.

PUT /lessons/{lessonId}

- Method: PUT
- Description: Updates a lesson by its ID.
- Parameters:
    - lessonId (String) - ID of the lesson to be updated.
    - updatedLesson (Lesson) - Updated Lesson object.
- Response: A Result object containing a message confirming the update.
- Response Status Codes:
    - 200 OK - Request successful.
    - 404 Not Found - If lesson is not found.

DELETE /lessons/{lessonId}

- Method: DELETE
- Description: Deletes a lesson by its ID.
- Parameters:
    - lessonId (String) - ID of the lesson to be deleted.
- Response: A Result object containing a message confirming the deletion.
- Response Status Codes:
    - 200 OK - Request successful.
    - 404 Not Found - If lesson is not found.

GET /lessons/{lessonId}

- Method: GET
- Description: Retrieves a lesson by its ID.
- Parameters:
    - lessonId (String) - ID of the lesson to be retrieved.
- Response: A Result object containing a Lesson object.
- Response Status Codes:
    - 200 OK - Request successful.
    - 404 Not Found - If lesson is not found.

POST /lessons

- Method: POST
- Description: Creates a new lesson.
- Parameters:
    - newLesson (Lesson) - Lesson object to be created.
- Response: A Result object containing a message confirming the creation.
- Response Status Codes:
    - 200 OK - Request successful.

POST /lessons/saveAllLessons

- Method: POST
- Description: Creates multiple lessons at once.
- Parameters:
    - lessons (List<Lesson>) - List of Lesson objects to be created.
- Response: A Result object containing a message confirming the creation.
- Response Status Codes:
    - 200 OK - Request successful.

POST /lessons/homework/{lid}

- Method: POST
- Description: Creates a predefined homework for a lesson.
- Parameters:
    - questions (List<GradedQuestion>) - List of graded questions to be added to the homework.
    - lid (String) - ID of the lesson for which the homework is to be created.
- Response: A Result object containing a message confirming the creation.
- Response Status Codes:

- 200 OK - Request successful.

POST /lessons/exam/{lid}

- Method: POST
- Description: Creates an exam for a lesson.
- Parameters:
  - questions (List<GradedQuestion>) - List of graded questions to be added to the exam.
  - lid (String) - ID of the lesson for which the exam is to be created.
- Response: A Result object containing a message confirming the creation.
- Response Status Codes:
  - 200 OK - Request successful.

POST /lessons/{lessonId}/grammars

- Method: POST
- Description: Adds grammars to a lesson.
- Parameters:
  - lessonId (String) - ID of the lesson to which the grammars are to be added.
  - grammars (List<Grammar>) - List of Grammar objects to be added to the lesson.
- Response: A Result object containing a message confirming the addition of grammars.
- Response Status Codes:
  - 200 OK - Request successful.

GET /lessons/{lessonId}/grammar-meanings

This method returns all the saved grammars for a particular lesson.

Request

- HTTP Method: GET
- URI: /lessons/{lessonId}/grammar-meanings
- URI Parameters:
  - {lessonId} (string, required): The ID of the lesson for which to retrieve grammars.
- Request Body: None
- Example URI: http://example.com/lessons/123/grammar-meanings

Response

- HTTP Status Code: 200 OK
- Response Body:

- flag (boolean): Whether the operation was successful or not.
- statusCode (integer): The HTTP status code for the response.
- message (string): A message describing the result of the operation.
- data (array of objects): An array of Grammar objects representing the saved grammars for the specified lesson. Each Grammar object has the following properties:
    - id (string): The unique identifier for the grammar.
    - chinese (string): The Chinese text of the grammar.
    - english (string): The English translation of the grammar.

❖ Homework

Base URL

/homework

Endpoints

Save Homework

POST /homework/{sid}/{lid}

Method to save a homework for a lesson of a section.

Request Parameters

Parameter	Type	Required	Description
sid	string	yes	The ID of the section that the lesson belongs to
lid	string	yes	The ID of the lesson

Request Body

This endpoint expects a JSON object containing a list of GradedQuestion objects.

Response Body

The response body is a Result object that contains a flag, status code, and message.

Find Homework By ID

GET /homework/{homeworkId}

Method to find a homework by its ID.

Request Parameters

Parameter	Type	Required	Description
homeworkId	string	yes	The ID of the homework to find

Response Body

The response body is a Result object that contains a flag, status code, message, and the found homework.

Find Homework By Section and Lesson ID

GET /homework/{sid}/{lid}

Method to find a homework for a lesson of a section.

Request Parameters

Parameter	Type	Required	Description
sid	string	yes	The ID of the section that the lesson belongs to

lid	string	yes	The ID of the lesson

Response Body

The response body is a Result object that contains a flag, status code, message, and the found homework.

Find All Homeworks

GET /homework

Method to find all homeworks.

Response Body

The response body is a Result object that contains a flag, status code, message, and a list of all found homeworks.

Update Homework

PUT /homework/{homeworkId}

Method to update a homework.

Request Parameters

Parameter	Type	Required	Description
homeworkId	string	yes	The ID of the homework to update

Request Body

This endpoint expects a JSON object containing an updated Homework object.

Response Body

The response body is a Result object that contains a flag, status code, and message.

Delete Homework

DELETE /homework/{homeworkId}

Method to delete a homework.

Request Parameters

Parameter	Type	Required	Description
homeworkId	string	yes	The ID of the homework to delete

Response Body

The response body is a Result object that contains a flag, status code, and message.

GradedQuestion

Base URL

The base URL for all endpoints is: /graded-question

Endpoints

Save Text Answer

Save a text answer for a question.

POST /answer/{username}/{id}

Parameters:

Name	Type	Description
username	string	Required. Username of the user who submitted the answer.
id	string	Required. ID of the question for which the answer is being submitted.
answer	Answer	Required. The answer object to be saved.

Response:

Name	Type	Description
flag	boolean	Indicates if the operation was successful.
statusCode	int	HTTP status code.
message	string	Description of the result of the operation.

Find All Answers

Find all answers for a question.

GET /answers/{id}

Parameters:

Name	Type	Description
id	string	Required. ID of the question for which the answers are being retrieved.

Response:

Name	Type	Description
flag	boolean	Indicates if the operation was successful.
statusCode	int	HTTP status code.
message	string	Description of the result of the operation.
data	List<Answer>	List of all answers for the question.

Save Comment

Save a comment on an answer.

POST /comment/{username}/{id}/{comment}

Parameters:

Name	Type	Description
username	string	Required. Username of the user who submitted the comment.
id	string	Required. ID of the answer for which the comment is being submitted.
comment	string	Required. The comment text to be saved.

Response:

Name	Type	Description
flag	boolean	Indicates if the operation was successful.
statusCode	int	HTTP status code.

message	string	Description of the result of the operation.

Find All Comments

Find all comments for an answer.

GET /comments/{id}

Parameters:

Name	Type	Description
id	string	Required. ID of the answer for which the comments are being retrieved.

Response:

Name	Type	Description
flag	boolean	Indicates if the operation was successful.
statusCode	int	HTTP status code.

message	string	Description of the result of the operation.
data	List<String>	List of all comments for the answer.

Error Responses

The following error responses may be returned by the API:

Status Code	Message	Description
400	Bad Request	The request was malformed or missing required parameters.

❖ Section

SectionController

This controller handles the operations related to sections, including managing sections, assigning students to sections, managing exams, and managing exam submissions.

Base URL

/sections

Endpoints

1. Get Section by ID

- URL: /sections/{sectionId}
- Method: GET
- Path Parameters:
  - sectionId: The ID of the section to be retrieved.
- Response:
  - Status code: 200 OK
  - Body:
    - flag: true for a successful operation.
    - status: StatusCode.SUCCESS for a successful operation.
    - message: "Find course by id success" for a successful operation.
    - data: The found section object.

2. Get Sections by Teacher

- URL: /sections/{username}/all
- Method: GET
- Path Parameters:
  - username: The username of the teacher whose sections are to be retrieved.
- Response:
  - Status code: 200 OK
  - Body:
    - flag: true for a successful operation.
    - status: StatusCode.SUCCESS for a successful operation.
    - message: "Find course by teacher success" for a successful operation.
    - data: The list of sections associated with the teacher.

3. Get Students in a Section

- URL: /sections/{sectionId}/students
- Method: GET
- Path Parameters:
  - sectionId: The ID of the section for which to retrieve the students.
- Response:
  - Status code: 200 OK
  - Body:
    - flag: true for a successful operation.
    - status: StatusCode.SUCCESS for a successful operation.
    - message: "Find students by course id success" for a successful operation.
    - data: The list of students in the section.

4. Save a Section

- URL: /sections/{username}
- Method: POST
- Path Parameters:
  - username: The username of the teacher who created the section.

- Request Body:
    - newSection: A JSON object representing the new section to be saved.
- Response:
    - Status code: 200 OK
    - Body:
        - flag: true for a successful operation.
        - status: StatusCode.SUCCESS for a successful operation.
        - message: "Save section success" for a successful operation.

5. Assign a Student to a Section

- URL: /sections/{sectionId}/{studentId}
- Method: PUT
- Path Parameters:
    - sectionId: The ID of the section to which the student should be assigned.
    - studentId: The ID of the student to be assigned to the section.
- Response:
    - Status code: 200 OK
    - Body:
        - flag: true for a successful operation.
        - status: StatusCode.SUCCESS for a successful operation.
        - message: "Student joined success" for a successful operation.

6. Save an Exam

- URL: /sections/exam/{sid}/{lid}/{start}/{day}/{length}
- Method: POST
- Path Parameters:
    - sid: The section ID.
    - lid: The lesson ID.
    - start: The start time of the exam.
    - day: The day of the exam.
    - length: The length of the exam.
    - Request Body:
        - questions: A JSON array of GradedQuestion objects.
    - Response:
        - Status code: 200 OK
        - Body:
            - flag: true for a successful operation.
            - status: StatusCode.SUCCESS for a successful operation.
            - message: "Save exam success" for a successful operation.

7. Get Exam by Section and Lesson ID

- URL: /sections/exam/{sid}/{lid}
- Method: GET
- Path Parameters:
    - sid: The section ID.

        - lid: The lesson ID.
- Response:
    - Status code: 200 OK
    - Body:
        - flag: true for a successful operation.
        - status: StatusCode.SUCCESS for a successful operation.
        - message: "Find exam by section success" for a successful operation.
        - data: The found exam object.

8. Save Exam Submission

- URL: /sections/submission/{sid}/{lid}
- Method: POST
- Path Parameters:
    - sid: The section ID.
    - lid: The lesson ID.
- Request Body:
    - answers: A JSON array of ExamAnswer objects.
- Response:
    - Status code: 200 OK
    - Body:
        - flag: true for a successful operation.
        - status: StatusCode.SUCCESS for a successful operation.
        - message: "Save exam submission success" for a successful operation.

9. Grade Exam Submission

- URL: /sections/submission/{sid}/{lid}/{idx}
- Method: PUT
- Path Parameters:
    - sid: The section ID.
    - lid: The lesson ID.
    - idx: An index representing the submission to be graded.
- Request Body:
    - answers: A JSON array of ExamAnswer objects.
- Response:
    - Status code: 200 OK
    - Body:
        - flag: true for a successful operation.
        - status: StatusCode.SUCCESS for a successful operation.
        - message: "Grade exam submission success" for a successful operation.

❖ TopicAnswer

This controller handles the operations related to topic answers, including adding answers to topics.

Base URL

/topicanswer

Endpoints

1. Create an Answer for a Topic
    - URL: /topicanswer/{topicId}/answer
    - Method: POST
    - Path Parameters:
        - topicId: The ID of the topic to which the answer should be added.
    - Request Body:
        - answer: A JSON object representing the answer to be added to the topic.
    - Response:
        - Status code: 200 OK
        - Body:
            - flag: true for a successful operation.
            - status: StatusCode.SUCCESS for a successful operation.
            - message: "Answer added to topic" for a successful operation.

❖ Topic

This controller handles the operations related to forum topics, including finding topics, creating, updating, and deleting topics.

Base URL

/topics

Endpoints

1. Find All Topics
    - URL: /topics
    - Method: GET
    - Response:
        - Status code: 200 OK

- Body:
    - flag: true for a successful operation.
    - status: StatusCode.SUCCESS for a successful operation.
    - message: "Find all topic!" for a successful operation.
    - data: A list of all forum topics.

2. Find Topic by ID
    - URL: /topics/{topicId}
    - Method: GET
    - Path Parameters:
        - topicId: The ID of the topic to be retrieved.
    - Response:
        - Status code: 200 OK
        - Body:
            - flag: true for a successful operation.
            - status: StatusCode.SUCCESS for a successful operation.
            - message: "Find by id success" for a successful operation.
            - data: The found topic object.
3. Create a Topic
    - URL: /topics/{sectionId}
    - Method: POST
    - Path Parameters:
        - sectionId: The ID of the section where the topic should be created.
    - Request Body:
        - newTopic: A JSON object representing the new forum topic to be created.
    - Response:
        - Status code: 200 OK
        - Body:
            - flag: true for a successful operation.
            - status: StatusCode.SUCCESS for a successful operation.
            - message: "Save success" for a successful operation.
4. Update a Topic
    - URL: /topics/{topicId}
    - Method: PUT
    - Path Parameters:
        - topicId: The ID of the topic to be updated.
    - Request Body:
        - updatedTopic: A JSON object representing the updated forum topic.
    - Response:
        - Status code: 200 OK
        - Body:
            - flag: true for a successful operation.
            - status: StatusCode.SUCCESS for a successful operation.
            - message: "updated" for a successful operation.
5. Delete a Topic
    - URL: /topics/{topicId}
    - Method: DELETE
    - Path Parameters:
        - topicId: The ID of the topic to be deleted.
    - Response:
        - Status code: 200 OK
        - Body:
            - flag: true for a successful operation.
            - status: StatusCode.SUCCESS for a successful operation.
            - message: "Delete success" for a successful operation.

❖ User

This controller handles the operations related to users, including finding users, creating, updating, and deleting users.

Base URL
/users

Endpoints

\Find User by Username

- URL: /users/{username}
- Method: GET
- Path Parameters:
  - username: The username of the user to be retrieved.
- Response:
  - Status code: 200 OK
  - Body:
    - flag: true for a successful operation.
    - status: StatusCode.SUCCESS for a successful operation.
    - message: "Find by user by username success" for a successful operation.
    - data: The found user object.

2. Find Users by Requested Role

- URL: /users/all/{requestedRole}
- Method: GET
- Path Parameters:
  - requestedRole: The requested role to filter users by.
- Response:
  - Status code: 200 OK
  - Body:
    - flag: true for a successful operation.
    - status: StatusCode.SUCCESS for a successful operation.
    - message: "Find by user by requested Role success" for a successful operation.
    - data: A list of users with the requested role.

3. Find All Users

- URL: /users
- Method: GET
- Response:
  - Status code: 200 OK
  - Body:
    - flag: true for a successful operation.
    - status: StatusCode.SUCCESS for a successful operation.
    - message: "Find All users" for a successful operation.
    - data: A list of all users.

4. Create a User
    - URL: /users
    - Method: POST
    - Request Body:
        - newUser: A JSON object representing the new user to be created.
    - Response:
        - Status code: 200 OK
        - Body:
            - flag: true for a successful operation.
            - status: StatusCode.SUCCESS for a successful operation.
            - message: "Create user success" for a successful operation.
5. Update a User
    - URL: /users/{username}
    - Method: PUT
    - Path Parameters:
        - username: The username of the user to be updated.
    - Request Body:
        - updatedUser: A JSON object representing the updated user.
    - Response:
        - Status code: 200 OK
        - Body:
            - flag: true for a successful operation.
            - status: StatusCode.SUCCESS for a successful operation.
            - message: "Update user success" for a successful operation.
6. Delete a User
    - URL: /users/{username}
    - Method: DELETE
    - Path Parameters:
        - username: The username of the user to be deleted.
    - Response:
        - Status code: 200 OK
        - Body:
            - flag: true for a successful operation.
            - status: StatusCode.SUCCESS for a successful operation.
            - message: "Delete success" for a successful operation.

❖ Vocab

VocabController

This controller handles operations related to vocabulary, including finding, creating, updating, and deleting vocabulary entries.

Base URL

/vocabsEndpoints

1. Find Vocab by ID
    - URL: /vocabs/{vocabId}

- Method: GET
- Path Parameters:
  - vocabId: The ID of the vocabulary entry to be retrieved.
- Response:
  - Status code: 200 OK
  - Body:
    - flag: true for a successful operation.
    - status: StatusCode.SUCCESS for a successful operation.
    - message: "Find vocab by id success" for a successful operation.
    - data: The found vocabulary entry object.

2. Find All Vocabulary Entries by Lesson
   - URL: /vocabs/vocab-lesson/{lessonId}
   - Method: GET
   - Path Parameters:
     - lessonId: The ID of the lesson to retrieve vocabulary entries for.
   - Response:
     - Status code: 200 OK
     - Body:
       - flag: true for a successful operation.
       - status: StatusCode.SUCCESS for a successful operation.
       - message: "list of all vocabs for that lesson success" for a successful operation.
       - data: A list of vocabulary entries for the specified lesson.
3. Save a Vocabulary Entry
   - URL: /vocabs/save/{lessonId}
   - Method: POST
   - Path Parameters:
     - lessonId: The ID of the lesson the vocabulary entry belongs to.
   - Request Body:
     - newVocab: A JSON object representing the new vocabulary entry to be created.
   - Response:
     - Status code: 200 OK
     - Body:
       - flag: true for a successful operation.
       - status: StatusCode.SUCCESS for a successful operation.
       - message: "Save vocab success" for a successful operation.
4. Save a List of Vocabulary Entries
   - URL: /vocabs/{lessonId}
   - Method: POST
   - Path Parameters:
     - lessonId: The ID of the lesson the vocabulary entries belong to.
   - Request Body:
     - vocabs: A JSON array of vocabulary entry objects to be created.
   - Response:
     - Status code: 200 OK
     - Body:
       - flag: true for a successful operation.
       - status: StatusCode.SUCCESS for a successful operation.
       - message: "Save vocabList success" for a successful operation.
5. Delete a Vocabulary Entry
   - URL: /vocabs/delete/{lessonId}/{vocabId}
   - Method: DELETE
   - Path Parameters:
     - lessonId: The ID of the lesson the vocabulary entry belongs to.
     - vocabId: The ID of the vocabulary entry to be deleted.

- Response:
    - Status code: 200 OK
    - Body:
        - flag: true for a successful operation.
        - status: StatusCode.SUCCESS for a successful operation.
        - message: "Delete vocab success!" for a successful operation.

6. Update a Vocabulary Entry

- URL: /vocabs/update/{lessonId}/{vocabId}
- Method: PUT
- Path Parameters:
    - lessonId: The ID of the lesson the vocabulary entry belongs to.
    - vocabId: The ID of the vocabulary entry to be updated.
- Request Body:
    - updatedVocab: A JSON object representing the updated vocabulary entry.
- Response:
    - Status code: 200 OK
    - Body:
        - flag: true for a successful operation.
        - status: StatusCode.SUCCESS for a successful operation.
        - message: "Update vocab success!" for a successful operation.