API Documentation

/[course]/create

Type: Form Data

Method: POST

Description: Creates a new assignment for a course. Only users with permission to create assignments for the course can perform this action. Validates the provided course ID and ensures all necessary assignment fields are included.

Parameters

- **courseId** UUID: The course ID to which the assignment is being created.
- **name** string: The name of the assignment.
- **description** string: The description of the assignment.
- **active** boolean: Whether the assignment is active or not.
- **dueDate** string: The due date for the assignment in a valid date string format.

Returns

- A success message if the assignment is created successfully, or a fail response with an error message.

Throws

- 400 - If the course ID or assignment fields are invalid.
- 403 - If the user does not have permission to create an assignment.
- 500 - If there is a database insertion error.

/[course]/delete

Type: Form Data

Method: POST

Description: Deletes an assignment from a course. Only users with permission to delete assignments for the course can perform this action. Validates the provided course ID and assignment ID before deletion.

Parameters

- **courseId** UUID: The course ID to which the assignment belongs.
- **assignmentId** UUID: The assignment ID to be deleted.

Returns

- A success message if the assignment is deleted successfully, or a fail response with an error message.

Throws

- 400 - If the course ID or assignment ID is invalid.
- 403 - If the user does not have permission to delete the assignment.
- 500 - If there is a database deletion error.

/[course]/update

Type: Form Data

Method: POST

Description: Updates fields for an existing assignment. Only users with permission to update assignments for the course can perform this action. Validates the provided assignment ID and ensures that at least one field is being updated.

Parameters

- **assignmentId** UUID: The assignment ID to be updated.
- **name** string: The new name of the assignment (optional).
- **description** string: The new description of the assignment (optional).
- **dueDate** string: The new due date for the assignment in a valid date string format (optional).
- **active** boolean: Whether the assignment is active or not (optional).

Returns

- A success message if the assignment is updated successfully, or a fail response with an error message.

Throws

- 400 - If the assignment ID is invalid or no fields to update are provided.
- 403 - If the user does not have permission to update the assignment.
- 500 - If there is a database update error.

/[course]/assignment/[assignment]/[student_assignment]/problem_description

Type: Form Data

Method: POST

Description: Updates the problem description. Only users with permission to update assignments for the course can perform this action. Validates the provided problem ID and description before updating the problem description in the database.

Parameters

- **course** UUID: The ID of the course to which the assignment belongs.
- **assignment** UUID: The ID of the assignment that the problem is part of.
- **student_assignment** UUID: The ID of the student assignment, if necessary for context.
- **problemId** UUID: The ID of the problem to update.
- **description** string: The new description for the problem.

Returns

- A success message if the problem description is updated successfully, or a fail response with an error message.

Throws

- 400 - If the problem ID is invalid or the description is missing.
- 403 - If the user does not have permission to update the problem description.
- 500 - If there is a database update error.

/[course]/assignment/[assignment]/[student_assignment]/problem_name

Type: Form Data

Method: POST

Description: Creates a new problem. Only users with permission to update assignments for the course can perform this action. Validates the provided problem name before inserting a new problem into the database.

Parameters

- **course** UUID: The ID of the course to which the assignment belongs.
- **assignment** UUID: The ID of the assignment that the problem is part of.
- **student_assignment** UUID: The ID of the student assignment, if necessary for context.
- **problemName** string: The name of the new problem.

Returns

- A success message if the new problem is created successfully, or a fail response with an error message.

Throws

- 400 - If the problem name is missing.
- 403 - If the user does not have permission to create a problem.
- 500 - If there is a database insertion error.

/[course]/assignment/[assignment]/[student_assignment]/delete_problem

Type: Form Data

Method: POST

Description: Deletes a problem. Only users with permission to update assignments for the course can perform this action. Validates the provided problem ID before deleting the problem from the database.

Parameters

- **course** UUID: The ID of the course to which the assignment belongs.
- **assignment** UUID: The ID of the assignment that the problem is part of.
- **student_assignment** UUID: The ID of the student assignment, if necessary for context.
- **problemId** UUID: The ID of the problem to delete.

Returns

- A success message if the problem is deleted successfully, or a fail response with an error message.

Throws

- 400 - If the problem ID is invalid or missing.
- 403 - If the user does not have permission to delete the problem.
- 500 - If there is a database deletion error.

/[course]/assignment/[assignment]/[student_assignment]/save_problem

Type: Form Data

Method: POST

Description: Saves a problem's content to a file. Only users with permission to update assignments for the course can perform this action. Validates the provided problem ID and content before saving the content to a file.

Parameters

- **course** UUID: The ID of the course to which the assignment belongs.
- **assignment** UUID: The ID of the assignment that the problem is part of.
- **student_assignment** UUID: The ID of the student assignment, if necessary for context.
- **problemId** UUID: The ID of the problem to save content for.
- **content** string: The content to be saved for the problem.

Returns

- A success message if the problem content is saved to a file successfully, or a fail response with an error message.

Throws

- 400 - If the problem ID or content is missing or invalid.
- 403 - If the user does not have permission to save the problem.
- 500 - If there is an error while saving the content to a file.

/[course]/gradebook/edit_grades

Type: Form Data

Method: POST

Description: Edits grades for assignments in a course. Only users with permission to change course grades can perform this action. Validates the provided student IDs, assignment IDs, and grades before updating the grades in the database.

Parameters

- **course** UUID: The course id for the grades that are being changed.
- **student_id** UUID[]: The array of student IDs whose grades need to be updated.
- **assignment_id** UUID[]: The array of assignment IDs for which grades need to be updated.
- **grade** number[]: The array of grades to be assigned to the students for the respective assignments.

Returns

- A success message if the grades are updated successfully, or a fail response with an error message.

Throws

- 400 - If the input data is mismatched or invalid (e.g., mismatched lengths of student IDs, assignment IDs, and grades).
- 403 - If the user does not have permission to edit the grades.
- 500 - If there is a database update error.

/[course]/statements/add

Type: Form Data

Method: POST

Description: Adds a new statement and uploads a file to the server. Validates the provided statement data and uploads the associated file to the server. A new record is added to the database for the statement.

Parameters

- **statement_name** string: The name of the statement to be added.
- **statement_type** string: The type of the statement to be added.
- **statement_description** string: The description of the statement to be added.
- **statement_category** string: The category of the statement to be added.
- **statement_file** string: The file content of the statement to be uploaded.

Returns

- A success message if the statement is added and file is uploaded successfully, or a fail response with an error message.

Throws

- 400 - If any required fields (statement_name, statement_type, statement_description, statement_category, or statement_file) are missing.
- 500 - If there is a database insertion error or file upload error.

/[course]/statements/remove

Type: Form Data

Method: POST

Description: Removes a statement from the database and deletes the associated file. Validates the provided statement ID, deletes the associated record from the database, and removes the file from the server.

Parameters

- **statement_id** string: The ID of the statement to be removed.

Returns

- A success message if the statement is removed and the file is deleted successfully, or a fail response with an error message.

Throws

- 400 - If the statement ID is missing.
- 500 - If there is a database deletion error or file removal error.

/[course]/users/add

Type: Form Data

Method: POST

Description: Adds a user to the course and assigns a role. Validates the provided user ID and role, and adds the user to the course with the specified role. If the user is assigned the "student" role, their assignments for the course are also created.

Parameters

- **user_id** string: The ID of the user to be added to the course.
- **role** string: The role to be assigned to the user (e.g., "student", "instructor").
- **courseId** UUID: The ID of the course to which the user is being added.

Returns

- A success message if the user is successfully added to the course and assigned a role, or a fail response with an error message.

Throws

- 400 - If the user ID or role is missing or if the user is already assigned a role.
- 403 - If the user does not have permission to update course users or the role is not valid for the user.
- 500 - If there is a database insertion error or error adding student assignments.

/[course]/users/remove

Type: Form Data

Method: POST

Description: Removes a user from the course and deletes their assignments. Validates the provided user ID and role, and removes the user from the course, deleting their assignments for the course if they are a student.

Parameters

- **user_id** string: The ID of the user to be removed from the course.
- **role** string: The role of the user to be removed (e.g., "student", "instructor").
- **courseId** UUID: The ID of the course from which the user is being removed.

Returns

- A success message if the user is successfully removed from the course and their assignments are deleted, or a fail response with an error message.

Throws

- 400 - If the user ID or role is missing.
- 403 - If the user does not have permission to update course users or the role is not valid for the user.
- 500 - If there is a database deletion error or error deleting student assignments.

/login/default

Type: Form Data

Method: POST

Description: Handles the user login process. Validates the username and password, checks for the existence of the user in the database, and compares the provided password with the stored password hash. If the credentials are correct, it creates a new session and sets a session cookie.

Parameters

- **username** string: The username entered by the user.
- **password** string: The password entered by the user.
- **cookies** Cookies: The cookies object for setting the session cookie.

Returns

- A fail response with an error message if the credentials are invalid or if there are any errors during the process.

Throws

- 400 - If the username or password is invalid.
- 500 - If there is an error querying the user in the database or during password verification.

/admin/add

Type: Form Data

Method: POST

Description: Adds a new user to the system. Only admin users can add new users. Validates username and password, ensures required fields are present, and handles unique constraint violations.

Parameters

- **username** string: The username of the new user.
- **password** string: The password for the new user.
- **first_name** string: The first name of the new user.
- **last_name** string: The last name of the new user.
- **email** string: The email address of the new user (optional).
- **is_admin** string: Whether the user should have admin privileges (optional, default: 'no').

Returns

- A success message if the user is added, or a fail response with an error message.

Throws

- 500 - If there is a database insertion error.

/admin/remove

Type: Form Data

Method: POST

Description: Removes a user from the system. Only admin users can remove users. Validates the provided user ID.

Parameters

- **user_id** string: The unique identifier (UUID) of the user to be removed.

Returns

- A success message if the user is removed, or a fail response with an error message.

Throws

- 500 - If there is a database deletion error.

/admin/update_user

Type: Form Data

Method: POST

Description: Updates fields for an existing user. Only admin users can update user fields. Validates the provided user ID and ensures that at least one field is being updated. Prevents users from demoting themselves from admin status.

Parameters

- **first_name** string: The new first name of the user (optional).
- **last_name** string: The new last name of the user (optional).
- **email** string: The new email address of the user (optional).
- **username** string: The new username for the user (optional).
- **password** string: The new password for the user (optional).
- **admin** boolean: Whether the user should be an admin (optional).
- **user_id** string: The unique identifier (UUID) of the user to be updated.

Returns

- Nothing if successful, or a fail response with an error message.

Throws

- 500 - If there is a database update error.

/admin/add_course

Type: Form Data

Method: POST

Description: Adds a new course to the system. Only admin users can add courses. Ensures all fields are provided before insertion.

Parameters

- **course_number** string: The unique course number.
- **course_name** string: The name of the course.
- **status** string: The current status of the course (e.g., 'active', 'inactive').
- **start_date** string: The start date of the course in ISO format.
- **end_date** string: The end date of the course in ISO format.

Returns

- A success message if the course is added, or a fail response with an error message.

Throws

- 500 - If there is a database insertion error.

/admin/remove_course

Type: Form Data

Method: POST

Description: Removes a course from the system. Only admin users can remove courses. Validates the provided course ID.

Parameters

- **course_id** string: The unique identifier (UUID) of the course to be removed.

Returns

- A success message if the course is removed, or a fail response with an error message.

Throws

- 500 - If there is a database deletion error.

/admin/update_course

Type: Form Data

Method: POST

Description: Updates fields for an existing course. Only admin users can update course fields. Validates the provided course ID and ensures that at least one field is being updated.

Parameters

- **course_number** string: The new course number (optional).
- **course_name** string: The new course name (optional).
- **status** string: The new status of the course (optional).
- **start_date** string: The new start date of the course in ISO format (optional).
- **end_date** string: The new end date of the course in ISO format (optional).
- **course_id** string: The unique identifier (UUID) of the course to be updated.

Returns

- Nothing if successful, or a fail response with an error message.

Throws

- 500 - If there is a database update error.

/apiv2/complete_proof

Type: JSON

Method: POST

Description: Marks a student's proof as complete. Updates the `complete` status for a specific proof in the database. This also causes the trigger in database to update grade for the assignment.

Parameters

- **proofId** UUID: Proof id to change the complete status
- **val** boolean: The complete status

Returns

- A JSON response confirming the update.

Throws

- 500 - If the database update operation fails.

/apiv2/load_proof

Type: JSON

Method: POST

Description: Loads or creates a student's proof file for a given problem. - If `orig` is `true`, returns the original problem file content. - If no `proofId` is provided, creates a new student proof entry. - Attempts to load the student's proof file; if missing and editable, creates a blank file.

Parameters

- **proofId** UUID: Proof id of which file to load from. Possibly null if proof doesn't exist yet.
- **problemId** UUID: Problem id to load from if proof id is null
- **studentAssignmentId** UUID: Student assignment id to correspond which student assignment the proof is for
- **orig** boolean: Whether to always to pull from problem file

Returns

- A JSON response containing the file content and `proofId`.

Throws

- 400 - If proof creation fails due to a bad request.
- 500 - If reading or writing files fails, or if database queries fail.

/apiv2/save_problem

Type: JSON

Method: POST

Description: Updates the content of a problem file. Checks user permission to update assignments for the given course, verifies the problem exists, and saves the new content to the problem's file.

Parameters

- **courseId** UUID: Course id to check if have permission to save a problem
- **problemId** UUID: Problem id for what the content is save to
- **content** string: Content to save to the problem

Returns

- A JSON response confirming the proof was saved successfully.

Throws

- 400 - If the problem ID is missing, invalid, or the problem does not exist.
- 403 - If the user does not have permission to update assignments.
- 500 - If there is a failure checking permissions or saving the file.

/apiv2/save_proof

Type: JSON

Method: POST

Description: Saves a student's proof content. Validates the provided proof ID, checks that the proof exists in the database, and writes the provided content to the corresponding proof file.

Parameters

- **proofId** UUID: Proof id for what the content is save to
- **content** string: Content to save to the proof

Returns

- A JSON response confirming that the proof was saved successfully.

Throws

- 400 - If the proof ID is missing or invalid.
- 500 - If the proof does not exist in the database, or if writing to the file fails.