



partnered with



School of Computing

KH5041CMD

The Internet and Web Technologies

[Project Name]

[Student Name] / [TKH ID] / [Coventry ID]

Ethical Hacking and Cybersecurity

School of Computing

Coventry University – The Knowledge Hub Universities

[Submission Date]

Module ML/ Dr. Amir Tarek

Module AL(s)/ Eng. Mariam Abdelati and Eng. Nour Awany

Required Components

Frontend GitHub Link:

<https://github.com/yassin202300080/Internet-and-web-technologies-frontend-Yassin-Elkholy-20230080->

Backend GitHub Link:

<https://github.com/yassin202300080/Internet-and-web-technologies-CW-Yassin-Elkholy-20230080->

Table of Contents

Required Components	ii
Table of Contents	iii
Table of Figures	iv
Table of Tables	v
1 Introduction.....	1
2 Background and Motivation	1
3 Comparison with Similar Applications	1
4 Project Goals	1
4.1 User Goals.....	1
4.2 Developer Goals.....	2
5 Application Modelling.....	3
5.1 Use Case Diagram.....	3
5.2 Use Case Scenarios	4
6 Technology Stack Selection	6
7 Project Architecture	6
8 Implementation.....	7
8.1 API Endpoints.....	7
8.1.1 Authentication Endpoints.....	7
8.1.2 User Management Endpoints.....	7
8.1.3 Core Features Endpoints	8
8.2 Classroom Management.....	8
8.3 Assignments & Submissions.....	9
8.4 Database Schema Design.....	10
8.4.1 Tables/Collections Design	10
8.5 Users	10
8.6 Classroom	10
8.7 Enrolments	11
8.8 Assignment table	11
8.9 Submission table	12
8.9.1 Relationships and constraints	13
9 Security Features Implementation.....	13

9.1 Secure Session Management 13

9.2 Password hashing..... 14

10 Postman Collections 15

10.1 Starting server..... 15

10.2 Authentication 16

10.3 User Authentication 17

10.4 Classroom Creation 18

10.5 Student Joining Classroom..... 18

10.6 Create assignments 19

10.7 Submissions 19

10.8 View submissions..... 20

10.9 Grade submissions 20

10.10 verify Deadlines..... 21

10.11 Submitting after Deadline: 23

10.12 Archiving 24

References..... 25

Table of Figures

Figure 1. [use case diagram]..... 3

Table of Tables

Table 1. User Application Goals..... 1

Table 2. Use Case Scenario for. 4

1 Introduction

The proposed web application is a secure classroom management system named FlashEdu designed to manage the interaction between staff and students. The system will provide teachers to manage their classrooms, post assignments and grade student submissions.

Students will be able to join classes and submit their assignments this makes the system an organized educational platform. The system provides security and performance and uses node.js backend with an Sq lite database.

The features include role-based access control which help maintains the systems security and the secure session management only uses the Http Only cookies

2 Background and Motivation

Many educational platforms depend on emails for communication, physical handing out assignments and various calendars applications for scheduling; that results in a slow workflow, many deadlines could be missed and having some communication gaps. Flash Edu a unified platform that brings these essential functions together in one secure, accessible location.

3 Comparison with Similar Applications

Flashedu is designed for speed and security, but similar application like Moodle and google classroom runs on a lightweight Node.js flash edu does not require cloud storage and uses a structured SQL database for all data persistence.

4 Project Goals

4.1 User Goals

Table 1. User Application Goals.

Feature	Description
User Authentication	Students and staff can register and log in to access their dashboard.
Classroom Management	Staff can create classes and generate class codes so students can join.
Assignment	Staff can create assignments and add deadlines and students can view them.

Submission	Students can submit assignments and upload their text answers before the deadlines.
Grading	Staff can view student answers, grade them and give feedback.

4.2 Developer Goals

- Secure users' passwords using bcrypt hashing
- Prevent cross site scripting
- Prevent Session Hijacking
- Use http only and same site cookies
- Ensure data integrity using foreign key constraints in SQLite.

5 Application Modelling

5.1 Use Case Diagram

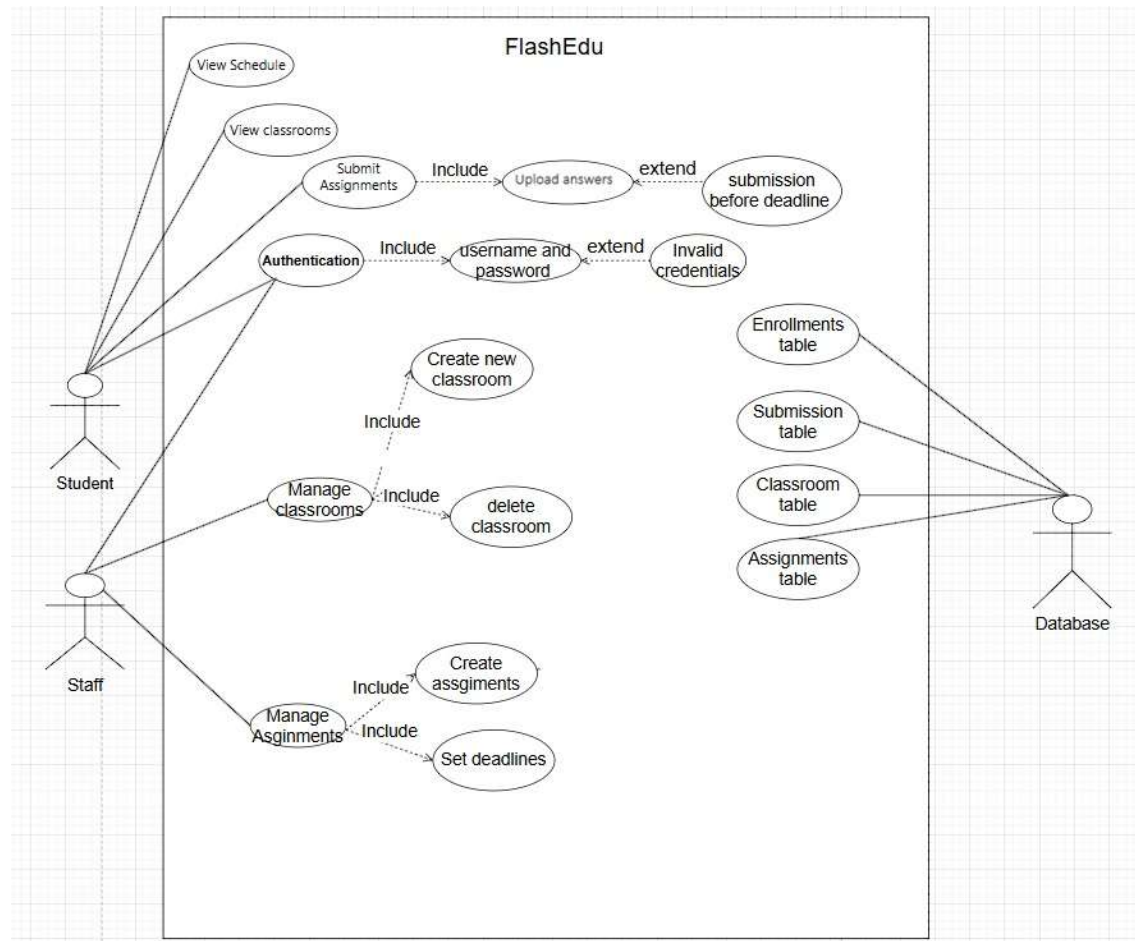


Figure 1. [use case diagram]

5.2 Use Case Scenarios

Table 2. Submitting assignments

Use Case #1: Submit assignments		
Goal in Context	Student submits text-based answers to assignments before the deadline.	
Scope & Level	Primary Task, System Level	
Preconditions	1. User is logged in as student 2. Assignment exists and deadline has not passed	
Success End Condition	Submission is saved to database with timestamp	
Failed End Condition	System rejects late submission	
Primary actor	Student	
Trigger	Student clicks Submit button on assignment page.	
Description	Step	Action
	1	Student navigates to the assignment page
	2	System retrieves assignment details
	3	Student enters their answer as text
	4	Student clicks submit
	5	System validates that the text is not empty
	6	System compares current time with due date
	7	System saves the submission to the database
Extensions	Step	Branching Action
	5a	If empty submission system returns request error
	6a	If deadline passed system returns deadline passed error
Sub-Variations	List	Branching Action
Resubmission		Can resubmit if deadline has not passed

Table 3. Manage Classrooms

Use Case #2: Manage Classrooms		
Goal in Context	Staff create a new digital classroom to organize students and assignments.	
Scope & Level	Primary Task, System Level	
Preconditions	1. User is logged in. 2. User has role as a staff	
Success End Condition	Classroom is created, saved to the database, and a unique Class Code is generated.	
Failed End Condition	Fails to create a classroom	
Primary actor	Staff	
Trigger	Click create classroom button	
Description	Step	Action
	1	Staff navigates to the classroom dashboard
	2	Staff creates new classroom
	3	enter the Classroom Name and description.
	4	System validates classroom not empty
	5	System generates
	6	System creates a Class Code
	7	New classroom saved to the database with status active
Extensions	Step	Branching Action
	5a	Missing name gives a request error and prompts user to add class name
	7b	If system cannot save the data system returns server error
Sub-Variations	List	Branching Action
Archiving Classrooms		Staff can update classroom status to be archived

6 Technology Stack Selection

Frontend uses HTML, CSS and JavaScript Backend uses Node.js and Express.js

SQL manages manage data for supporting optimized queries while express will handle secure authentication and session management.

Database uses sql lite and security implemented uses bcryptjs for hashing, jsontoken for authentication and cookie parser for session management.

7 Project Architecture

The model view controller MVC used

1. Models: Handles database interaction uses sql queries.
2. Controllers: systems logic
3. Routes: the API endpoints.
4. Middleware: verifies the authentication tokens before they reach controllers

8 Implementation

8.1 API Endpoints

8.1.1 Authentication Endpoints

Description	Endpoint	http method	Request	Response
Register new user	api/auth/register	Post	<ol style="list-style-type: none"> 1. Email 2. Password 3. First name 4. Last name 5. role 	<p>201 Create:</p> <ol style="list-style-type: none"> 1. JSON: User info & Token 2. Header: Set-Cookie (HttpOnly)
User login	api/auth/login	Post	Email and password	<p>200 Successful login</p> <ol style="list-style-type: none"> 1. JSON: User info & Token 2. Header: Set-Cookie (HttpOnly)
User logout	api/auth/logout	Post	No request	Header: Clear-Cookie

8.1.2 User Management Endpoints

Description	Endpoint	http method	Access	parameters
current user profile	api/users/me	Get	Authenticated	No parameters
Update profile details	api/users/me	Put	Authenticated	<ol style="list-style-type: none"> 1. First name 2. Last name

8.1.3 Core Features Endpoints

8.2 Classroom Management

Description	Endpoint	HTTP Method	Access	parameters
List user's classrooms	api/classrooms	Get	authenticated	Uses User ID from Token
Create new classroom	api/classrooms	Post	Staff	<ol style="list-style-type: none"> 1. Name 2. Description 3. Schedule
Join Classroom	/classrooms/join	Post	Student	Class code
Get classroom details	/classrooms/id	Get	Both	Id path
Update classroom	/classrooms/id	Put	Staff	<ol style="list-style-type: none"> 1. Name 2. Description 3. Schedule
Archive Classroom	/classrooms/id /Archive	Patch	Staff	Archived
Delete classroom	/classrooms/id	Delete	Staff	Id path

8.3 Assignments & Submissions

Description	Endpoint	HTTP Method	Access	parameters
Create assignment	api/assignments	Post	staff	<ol style="list-style-type: none"> 1. Classroom Id 2. Title 3. Description 4. Due Date 5. max points
List assignments	api/classrooms/id/assignments	Get	both	Id path
Get assignment details	/assignments/id	Get	both	Id path
Update assignment	/assignments/id	Put	staff	<ol style="list-style-type: none"> 1. title 2. description 3. due Date
Submit assignment	/assignments/id/Submit	Post	student	Text submission
View submissions	/assignments/id/submissions	Get	Staff	Id path
Grade submission	/submissions/id/grade	Put	Staff	Grade feed back

8.4 Database Schema Design

8.4.1 Tables/Collections Design

8.5 Users

Name	Key type	Description	Constraints
User id	Primary	Unique user identifier	Auto Increment
email		User email address	Unique not null
Role		Student or student	not null
First name		User's first name	not null
Last name		User's last name	not null
Time created		Account creation time	Default current

8.6 Classroom

name	Key type	Description	Constraints
Classroom id	Primary	Unique classroom identifier	Auto Increment
Teacher id	Foreign	Reference user from user id	Not null
Name		Classroom name	Not null
Description		Brief description	
Class code		Code for students to join	Unique Not null
Schedule		schedule data	

Status		Active or archived	Default active
Time created		Creation timestamp	Default current

8.7 Enrolments

Name	Key Type	Description	constraints
Enrolment id	primary	Unique enrolment ID	Auto Increment
Classroom id	Foreign	References classrooms	Not null
Student id	Foreign	References users	Not null
Enrolled time		Time of enrolment	Default current
Combined key	Unique	Prevents duplicate students in one class	Unique Classroom id and student id

8.8 Assignment table

Name	Key Type	Description	constraints
Assignment id	Primary	Unique assignment ID	Auto increment
Classroom id	Foreign	References classrooms	Not null
Title		Assignment title	Not null
description		Assignment instructions	
Due date		Submission deadline	Not null
Max points		Maximum possible points	Default 100 points
Created at		Creation timestamp	Default current

8.9 Submission table

Name	Key Type	Description	constraints
Submission id	Primary	Unique submissions id	Not null
Assignment id	Foreign	References assignments	Not null
Student id	Foreign	References users	Not null
Submission text		Student's text answer	
status		Submitted or graded	Default submitted
Grade		Points awarded	
feedback		Teacher feedback comments	
Submitted at		Submission timestamp	Default current

8.9.1 Relationships and constraints

- The Staff and Classroom: It's a one-to-many relationship because a single staff member will create and manage multiple classrooms.
- Students and classrooms: it's a many to many relationships that is implemented by the enrolments table to allow students join multiple classes and the classrooms will have multiple students.
- Classroom and assignments: one to many relationship classroom can have multiple assignments, but an assignment belongs to one classroom.
- Assignments and submissions: every classroom has a one-to-many relationship with assignments. Each assignment is connected to submission is one to many.

9 Security Features Implementation

9.1 Secure Session Management

To prevent Cross Site Scripting attacks uses secure HTTP cookies and doesn't store web tokens. Hhttp only flag ensures cookie is not accessed by the client side.

Samesite prevents requests attacks and the cookie is only sent for requests from the same website.

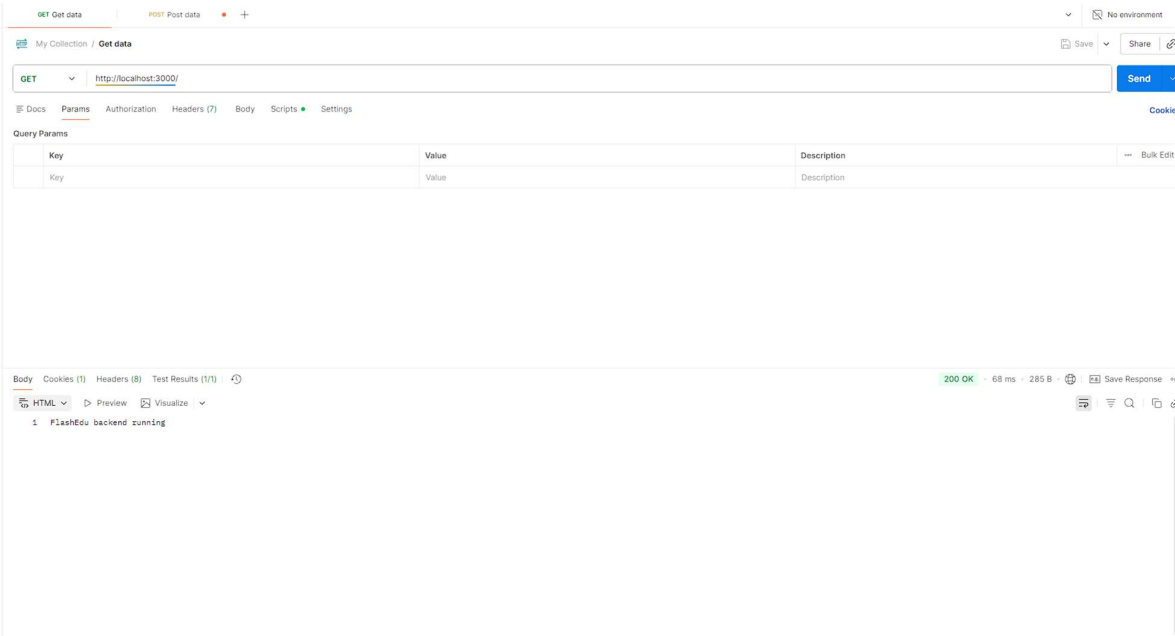
```
// set cookie
res.cookie('token', token, {
  httpOnly: true,
  secure: process.env.NODE_ENV === 'production',
  sameSite: 'strict',
  maxAge: 24 * 60 * 60 * 1000
});
```

9.2 Password hashing

```
bcrypt.hash(password, 10, (err, hash) => {  
  if (err) {  
    return res.status(500).json({ error: "Error hashing password" });  
  }  
})
```

10 Postman Collections

10.1 Starting server



10.2 Authentication

The screenshot shows the Swagger UI for the endpoint `POST /api/v1/auth/register`. The 'Body' tab is active, showing a JSON payload for a user registration. The payload is as follows:

```

1 {
2   "first_name": "Yassin",
3   "last_name": "Teacher",
4   "email": "yassin.elkholy@gmail.com",
5   "password": "ye202300000",
6   "role": "staff"
7 }

```

The status bar at the bottom indicates a `201 Created` response.

Staff:

Student:

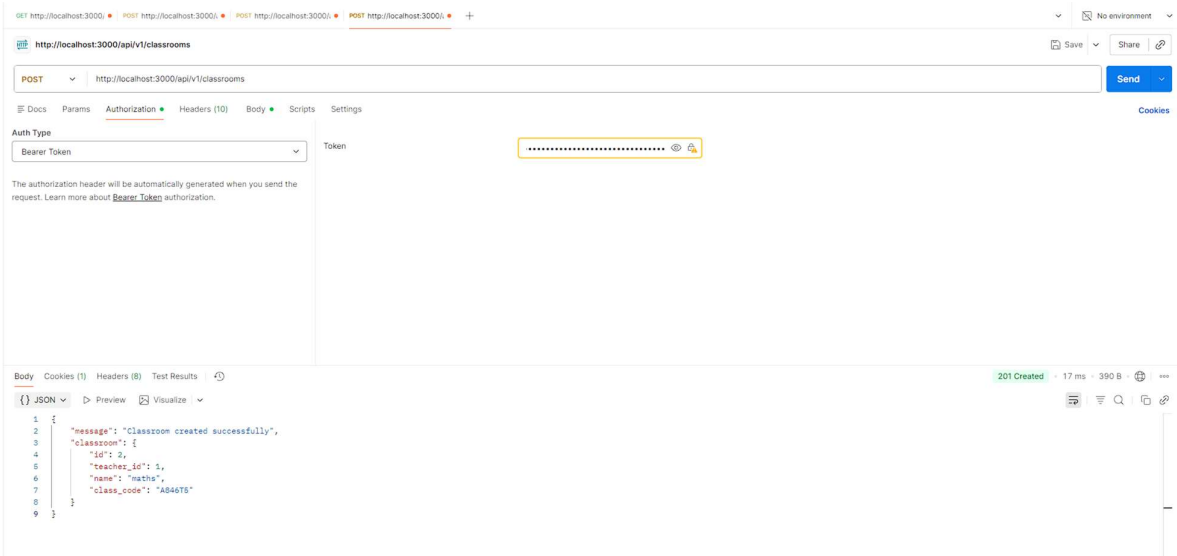
The screenshot shows the Postman interface with a POST request to `http://localhost:3000/api/v1/auth/register`. The request body is a JSON object:

```
{
  "first_name": "Ali",
  "last_name": "Student",
  "email": "ali@gmail.com",
  "password": "password021",
  "role": "student"
}
```

The response is a 201 Created status with a response time of 112 ms and a body size of 837 B. The response body is a JSON object:

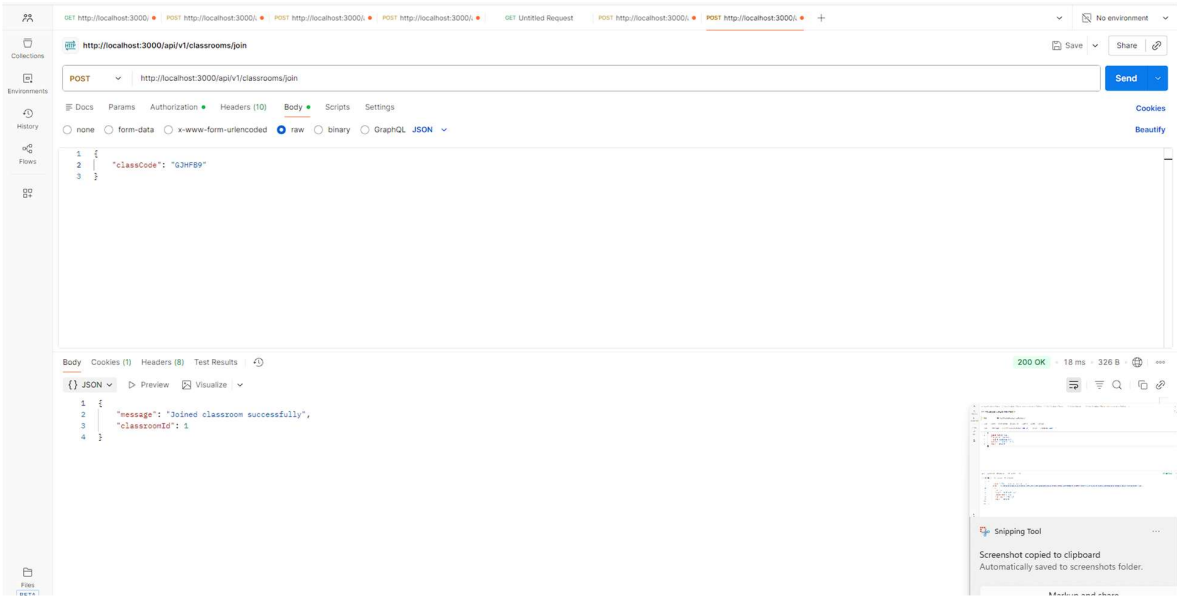
```
{
  "message": "User registered successfully",
  "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MjEsIm5hbWUiOiJ1bWU0NTIiLCJyb290IjoiIn0.20n0wGred83T0nMGL8787P1c5-upuQLnEifuy_1Ia",
  "user": {
    "id": 2,
    "email": "ali@gmail.com",
    "first_name": "Ali",
    "last_name": "Student",
    "role": "student"
  }
}
```

10.4 Classroom Creation



10.5 Student Joining Classroom

Many to many relationship



10.6 Create assignments

GET http://localhost:3000/ POST http://localhost:3000/ POST http://localhost:3000/ POST http://localhost:3000/ GET Untitled Request POST http://localhost:3000/ POST http://localhost:3000/ POST http://localhost:3000/ +

http://localhost:3000/api/v1/assignments Save Share

POST http://localhost:3000/api/v1/assignments Send

Docs Params Authorization Headers (10) Body Scripts Settings

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL ☐ JSON

```
1 {
2   "classroomId": 1,
3   "title": "First Project",
4   "description": "Momentum",
5   "dueDate": "2020-12-31"
6 }
```

Body Cookies (1) Headers (8) Test Results

JSON Preview Visualize

```
1 {
2   "message": "Assignment created successfully",
3   "assignmentId": 1
4 }
```

201 Created - 19 ms - 334 B

10.7 Submissions

POST http://localhost:3000/api/v1/submissions/1 Send

Docs Params Authorization Headers (10) Body Scripts Settings

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL ☐ JSON

```
1 {
2   "submissionText": "p=mv Here is my answer to the project. "
3 }
```

Body Cookies (1) Headers (8) Test Results

JSON Preview Visualize

```
1 {
2   "message": "Assignment submitted successfully",
3   "submissionId": 1
4 }
```

201 Created - 17 ms - 336 B

10.8 View submissions

GET

http://localhost:3000/api/v1/submissions/assignment/1

Send

Docs Params Authorization Headers (8) Body Scripts Settings

Auth Type
Bearer Token

The authorization header will be automatically generated when you send the request. Learn more about [Bearer Token](#) authorization.

Token
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1b290IjoiMSIsImVudCI6ImF1dG8iLCJpYXQiOiIyMDI0LTA4LTA4In0.eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1b290IjoiMSIsImVudCI6ImF1dG8iLCJpYXQiOiIyMDI0LTA4LTA4In0

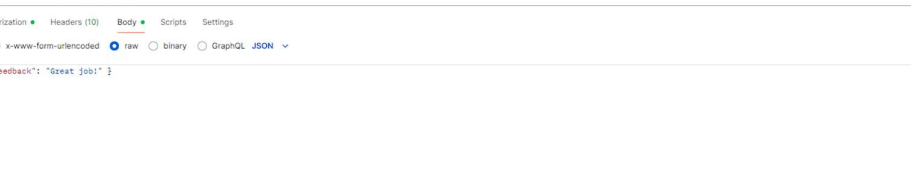
Body Cookies (1) Headers (8) Test Results

JSON Preview Visualize

```
1 {
2   "submissions": [
3     {
4       "submission_id": 1,
5       "assignment_id": 1,
6       "student_id": 2,
7       "submission_text": "pmv Here is my answer to the project. ",
8       "status": "submitted",
9       "grade": null,
10      "feedback": null,
11      "submitted_at": "2020-12-08 10:39:13",
12      "first_name": "Ali",
13      "last_name": "Student",
14      "email": "ali@gmail.com"
15    }
16  ]
17 }
```

200 OK 9 ms - 551 B


10.9 Grade submissions







The screenshot displays the Postman application interface. At the top, the URL bar shows `http://localhost:3000/api/v1/submissions/1/grade`. The method is set to `PUT`. The request body is a JSON object: `{ "grade": 95, "feedback": "Great job!" }`. The response is a `200 OK` status with a message: `"Submission graded successfully"`. The interface includes tabs for Docs, Params, Authorization, Headers (10), Body, Scripts, and Settings. The Body tab is selected, showing the raw JSON data. The response tab is also visible, showing the status and message.


10.10 verify Deadlines

Update an Assignment


 <http://localhost:3000/api/v1/assignments/1>






PUT  <http://localhost:3000/api/v1/assignments/1>

 Docs Params Authorization  Headers (10) Body  Scripts Settings

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL **JSON** 



```
1 { "title": "Course work project", "dueDate": "2023-01-01 23:59:59" }
```

Body Cookies (1) Headers (8) Test Results 

 **JSON**   Preview  Visualize 

```
1 {
2   |   "message": "Assignment updated successfully"
3 }
```

View :

 http://localhost:3000/api/v1/assignments/1 

GET

http://localhost:3000/api/v1/assignments/1

Docs

Params

Authorization

Headers (8)

Body

Scripts

Settings

Auth Type

Bearer Token

Token

.....

The authorization header will be automatically generated when you send the request. Learn more about [Bearer Token](#) authorization.

Body

Cookies (1)

Headers (8)

Test Results

200 OK

{}

JSON

Preview

Visualize

1 {

2 "assignment": {

3 "assignment_id": 1,

4 "classroom_id": 1,

5 "title": "Course work project",

6 "description": null,

7 "due_date": "2025-12-31 23:59:59",

8 "max_points": 100,

9 "created_at": "2025-12-13 11:06:55"

10 }

11 }

10.11 Submitting after Deadline:

The screenshot shows a REST client interface with the following components:

- URL Bar:** `http://localhost:3000/api/v1/submissions/1`
- Method:** `POST`
- Body Tab:** Selected, showing the request body as raw JSON:

```
1 {  
2   "submissionText": "math cw project "  
3 }
```
- Response Tab:** Selected, showing the response body as raw JSON:

```
1 {  
2   "error": "Deadline has passed. Submission rejected."  
3 }
```

The response indicates that the submission was rejected because the deadline has passed.

10.12 Archiving

http://localhost:3000/api/v1/classrooms/1/archive

PATCH

http://localhost:3000/api/v1/classrooms/1/archive

Send

Docs

Params

Authorization

Headers (9)

Body

Scripts

Settings

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

JSON

1

Ctrl+Alt+P to Ask AI

Body

Cookies (1)

Headers (8)

Test Results

200 OK

13 ms

312 B

JSON

Preview

Visualize

1

{

2

"message": "Classroom archived successfully"

3

}

check classroom status:

http://localhost:3000/api/v1/classrooms/1

GET

http://localhost:3000/api/v1/classrooms/1

Send

Docs

Params

Authorization

Headers (8)

Body

Scripts

Settings

Auth Type

Bearer Token

Token

.....

The authorization header will be automatically generated when you send the request. Learn more about [Bearer Token](#) authorization.

Body

Cookies (1)

Headers (8)

Test Results

200 OK

8 ms

498 B

JSON

Preview

Visualize

1

{

2

"classroom_id": 1,

3

"teacher_id": 1,

4

"name": "maths",

5

"description": "mechanics and statistics",

6

"class_code": "2H506X",

7

"schedule": "{\\"days\\": [\\"Mon\\",\\"Wed\\"],\\"time\\": \\"10:00 AM\\"}",

8

"status": "archived",

9

"created_at": "2025-12-13 11:04:28"

10

}

References

Firer, A., Kopeikin, A., & Khramov, I. (2025). *Features of automation of the university schedule management process through the “Class Schedule WEB application”*. ITM Web of Conferences, 72, 02006.

DOI:

https://www.itm-conferences.org/articles/itmconf/pdf/2025/03/itmconf_hmmocs-III2024_02006.pdf