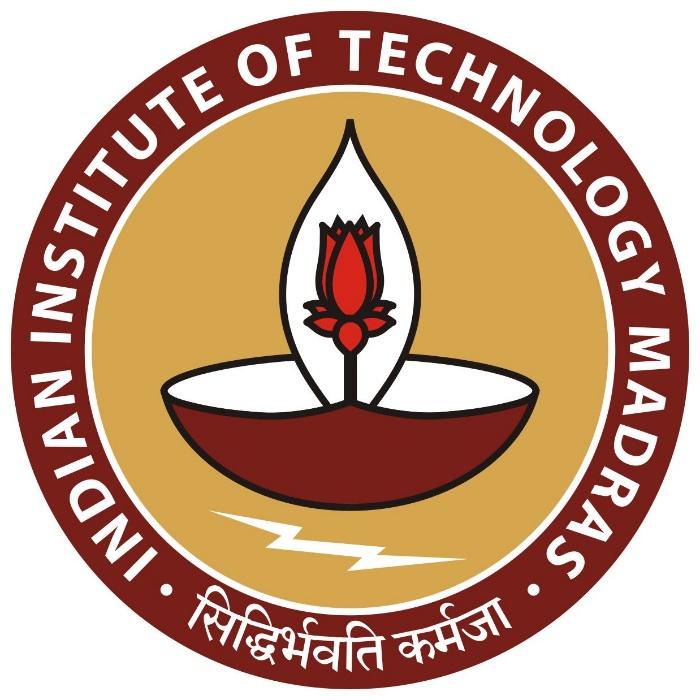
## Software Engineering Project

## January 2025 Term

## Milestone 5



## Submitted by Team 22:

Members:

1. Indranil Bhattacharyya 21f1005840

2. Sayan Biswas 22f3001500

3. Amisha Mishra 22f1000938

4. Priyanka Narula 21f3002839

5. Dinkar R Sharma 21f3003042

6. Dhruv Subhash Rathi 22f3001413

7. Hardik Aggarwal 21f1005570

## Administrative APIs

## **Testing Admin API**

## **Endpoint: Create or Update Course Information**

* URL: http://127.0.0.1:5000/api/admin\_course
* Method: POST

1. Test Case: test\_update\_course\_info
   * Description: Tests updating course information with valid admin authentication.
   * Passed Inputs:
     + headers: Authentication-Token: [valid admin token]
     + data: {'course\_id': 1, 'course\_name': 'Updated Course Name', 'instructor\_id': 3}
   * Expected Output:
     + HTTP-Status Code: 200
     + Response: {'message': 'Course information updated successfully'}
   * Actual Output:
     + HTTP-Status Code: 200
     + Response: {'message': 'Course information updated successfully'}
   * Result: Passed
   * Pytest Code:

def test\_update\_course\_info(client):

headers = {"Authentication-Token": get\_admin\_auth\_token(client)}

updated\_course\_data = {

"course\_id": 1,

"course\_name": "Updated Course Name",

"instructor\_id": 3

}

response = client.post('/api/admin\_course',

headers=headers,

data=json.dumps(updated\_course\_data),

content\_type='application/json')

assert response.status\_code == 200

2. Test Case: test\_create\_course\_info\_invalid\_input

* Description: Tests creating or updating course information with invalid input.
* Passed Inputs:
  + headers: Authentication-Token: [valid admin token]
  + data: {'course\_id': 'invalid', 'course\_name': 'New Course Name'}
* Expected Output:
  + HTTP-Status Code: 400
  + Response: {'message': 'Invalid input'}
* Actual Output:
  + HTTP-Status Code: 400
  + Response: {'message': 'Invalid input'}
* Result: Passed
* Pytest Code:

def test\_create\_course\_info\_invalid\_input(client):

headers = {"Authentication-Token": get\_admin\_auth\_token(client)}

invalid\_course\_data = {

"course\_id": "invalid",

"course\_name": "New Course Name"

}

response = client.post('/api/admin\_course',

headers=headers,

data=json.dumps(invalid\_course\_data),

content\_type='application/json')

assert response.status\_code == 400

3. Test Case: test\_create\_course\_info\_unauthorized

* Description: Tests creating or updating course information without authentication.
* Passed Inputs:
  + data: {'course\_id': 1, 'course\_name': 'Updated Course Name'}
* Expected Output:
  + HTTP-Status Code: 401
  + Response: {'message': 'Unauthorized'}
* Actual Output:
  + HTTP-Status Code: 401
  + Response: {'message': 'Unauthorized'}
* Result: Passed
* Pytest Code:

def test\_create\_course\_info\_unauthorized(client):

updated\_course\_data = {

"course\_id": 1,

"course\_name": "Updated Course Name"

}

response = client.post('/api/admin\_course',

data=json.dumps(updated\_course\_data),

content\_type='application/json')

assert response.status\_code == 401

## Instructor APIs

## **Testing Instructor API**

## **Endpoint: Get Assigned Courses for an Instructor**

* URL: http://127.0.0.1:5000/api/instructor\_assigned\_course
* Method: GET

1. Test Case: test\_get\_assigned\_courses
   * Description: Tests retrieval of all courses assigned to an instructor with valid authentication.
   * Passed Inputs:
     + headers: Authentication-Token: [valid instructor token]
   * Expected Output:
     + HTTP-Status Code: 200
     + Response: [{'course\_id': 1, 'course\_name': 'Course Name'}]
   * Actual Output:
     + HTTP-Status Code: 200
     + Response: [{'course\_id': 1, 'course\_name': 'Course Name'}]
   * Result: Passed
   * Pytest Code:

def test\_get\_assigned\_courses(client):

headers = {"Authentication-Token": get\_instructor\_auth\_token(client)}

response = client.get('/api/instructor\_assigned\_course', headers=headers)

assert response.status\_code == 200

5. Test Case: test\_get\_assigned\_courses\_unauthorized

* Description: Tests retrieving assigned courses without authentication.
* Passed Inputs: None
* Expected Output:
  + HTTP-Status Code: 401
  + Response: {'message': 'Unauthorized'}
* Actual Output:
  + HTTP-Status Code: 401
  + Response: {'message': 'Unauthorized'}
* Result: Passed
* Pytest Code:

def test\_get\_assigned\_courses\_unauthorized(client):

response = client.get('/api/instructor\_assigned\_course')

assert response.status\_code == 401

## **Endpoint: Upload Course Content**

* URL: http://127.0.0.1:5000/api/course\_content/{course\_id}
* Method: POST

1. Test Case: test\_upload\_course\_content
   * Description: Tests uploading new course content with valid instructor authentication.
   * Passed Inputs:
     + headers: Authentication-Token: [valid instructor token]
     + course\_id: 1
     + data: {'content': 'New course content'}
   * Expected Output:
     + HTTP-Status Code: 200
     + Response: {'message': 'Course content uploaded successfully'}
   * Actual Output:
     + HTTP-Status Code: 200
     + Response: {'message': 'Course content uploaded successfully'}
   * Result: Passed
   * Pytest Code:

def test\_upload\_course\_content(client):

headers = {"Authentication-Token": get\_instructor\_auth\_token(client)}

course\_id = 1

new\_content = {'content': 'New course content'}

response = client.post(f'/api/course\_content/{course\_id}',

headers=headers,

data=json.dumps(new\_content),

content\_type='application/json')

assert response.status\_code == 200

7. Test Case: test\_upload\_course\_content\_invalid\_input

* Description: Tests uploading course content with invalid input.
* Passed Inputs:
  + headers: Authentication-Token: [valid instructor token]
  + course\_id: 1
  + data: {'invalid\_key': 'Invalid content'}
* Expected Output:
  + HTTP-Status Code: 400
  + Response: {'message': 'Invalid input'}
* Actual Output:
  + HTTP-Status Code: 400
  + Response: {'message': 'Invalid input'}
* Result: Passed
* Pytest Code:

def test\_upload\_course\_content\_invalid\_input(client):

headers = {"Authentication-Token": get\_instructor\_auth\_token(client)}

course\_id = 1

invalid\_content = {'invalid\_key': 'Invalid content'}

response = client.post(f'/api/course\_content/{course\_id}',

headers=headers,

data=json.dumps(invalid\_content),

content\_type='application/json')

assert response.status\_code == 400

## **Endpoint: Create New Assignment**

* URL: http://127.0.0.1:5000/api/assignments
* Method: POST

1. Test Case: test\_create\_assignment
   * Description: Tests creating a new assignment with valid instructor authentication.
   * Passed Inputs:
     + headers: Authentication-Token: [valid instructor token]
     + data: {'assignment\_name': 'New Assignment', 'course\_id': 1, 'due\_date': '2024-01-01'}
   * Expected Output:
     + HTTP-Status Code: 201
     + Response: {'assignment\_id': 1, 'assignment\_name': 'New Assignment'}
   * Actual Output:
     + HTTP-Status Code: 201
     + Response: {'assignment\_id': 1, 'assignment\_name': 'New Assignment'}
   * Result: Passed
   * Pytest Code:

def test\_create\_assignment(client):

headers = {"Authentication-Token": get\_instructor\_auth\_token(client)}

new\_assignment\_data = {

"assignment\_name": "New Assignment",

"course\_id": 1,

"due\_date": "2024-01-01"

}

response = client.post('/api/assignments',

headers=headers,

data=json.dumps(new\_assignment\_data),

content\_type='application/json')

assert response.status\_code == 201