MileStone - 5 Team - 21

IIT Madras
BS Degree

SOFTWARE ENGINEERING PROJECT

SMART SEEK PORTAL

Prepared by :

Sayan Hrik
21f3002833@ds.study.iitm.ac.in

Avijeet Palit
21f1005675@ds.study.iitm.ac.in

Uroosha Rahat
21f1002968@ds.study.iitm.ac.in

Preetam Mukherjee
21f1002436@ds.study.iitm.ac.in

Ayush Singh Rana
21f1005671@ds.study.iitm.ac.in

Ramesh Kumar Chandran
21f2000549@ds.study.iitm.ac.in

Bodhisatwa Bhattacharya
21f1000270@ds.study.iitm.ac.in

Sachin Kumar
21f2000143@ds.study.iitm.ac.in

Contents

1. Test Cases and Descriptions for Components:

1.1 Student Component Testing:

Test Cases for Authentication

Test Case: Successful Registration of Student

API being tested: /student (POST method)

Inputs:

Student_email: “21f2000143@ds.study.iitm.ac.in”

Expected Output:

Status Code: 200 (OK)

JSON:


```
{
  "success": true,
  "token":
"eyJ2ZXIiOiI1IiwidWlkIjoiZGI5NGI5NTI5MGM4NGQ4NzkzOWVmZDJlMWU4OWM5NjQi
LCJzaWQiOjAsImV4cCI6MH0.ZrYPdA.4VUwnQOFuoyeC4eDW1aLDsbWFAUw",
  "user_data": {
    "email": "21f2000143@ds.study.iitm.ac.in",
    "first_name": "Sachin",
    "last_name": "Kumar"
  }
}
```


Actual Output:

Status Code: 200 (OK)

JSON:


```
{
  "success": true,
  "token":
"eyJ2ZXIiOiI1IiwidWlkIjoiZGI5NGI5NTI5MGM4NGQ4NzkzOWVmZDJlMWU4OWM5NjQi
LCJzaWQiOjAsImV4cCI6MH0.ZrYPdA.4VUwnQOFuoyeC4eDW1aLDsbWFAUw",
  "user_data": {
    "email": "21f2000143@ds.study.iitm.ac.in",
    "first_name": "Sachin",
    "last_name": "Kumar"
  }
}
```

Result: Success

Test Case: Bad Request - Invalid Input Data
API being tested: /student/<int:student_id> (POST method)
Inputs:

Expected Output:
Status Code: 400
JSON:
{
"error": "Invalid email",
"success": false
}

Actual Output:
Status Code: 400
JSON:
{
"error": "Invalid email",
"success": false
}
Result: Success

Test Cases for Notes

Test Case: Creating / adding notes to a module
API being tested: /notes (POST method)
Inputs:
module_id: "1"
title: "Programming in Python"
content: "This lecture covers the basics of Python programming. "

Expected Output:
Status Code: 201
JSON:
{

"id": 1,
"module_id": "CS101",
"title": "Introduction to Python",
"content": "This lecture covers the basics of Python programming.",
}

Actual Output:
Status Code: 201 (Created)
JSON:
{
"id": 1,
"module_id": "CS101",
"title": "Introduction to Python",
"content": "This lecture covers the basics of Python programming.",
}

Result: Success

Test Case: Bad Request for note creation

API being tested: /student (POST method)
Inputs:
module_id: (missing)
title: “Python Notes”
content: “This note has missing information.”

Expected Output:
Status Code: 400 **(**BAD Request**)**
JSON:
{
"message": "Invalid input data"
}

Actual Output:
Status Code: 400
JSON:

{
"message": "Invalid input data"
}

Result: Success

Test Case: Deleting notes
API being tested:/notes/<int:note_id> (DELETE method)
Inputs:
note_id: 1

Expected Output:
Status Code: 200 (OK)
JSON:
{'message': 'Note deleted'}

Actual Output:
Status Code: 200 (OK)
JSON:
{'message': 'Note deleted'}

Result: Success

Test Case: BAD Request (Deleting a note that does not exist)
API being tested:/notes/<int:note_id> (DELETE method)
Inputs:
note_id: 100 (invalid note id)
Expected Output:
Status Code: 404 (Not Found)
JSON:
{
"message": "Note not found"
}

Actual Output:

Status Code: 404 (Not Found)
JSON:
{
"message": "Note not found"
}

Result:Success

Test Case For Courses

Test Cases for Course Management

Test Case: Get Details of a Course
API being tested: `/courses/<int:course_id>` (GET method)
Inputs:

`course_id`:1

Expected Output:

Status Code: 200 (OK)

JSON:

```
{
    "course_id": 1,
    "course_name": "Programming in Python",
    "description": "A foundational course in python for data science.",
    "modules": [
        {
            "module_id": "CS101",
            "module_name": "Introduction to Python",
            "description": "Basics of Python programming."
        },
           ]
}
```

Actual Output:

Status Code: 200 (OK)

JSON:


```
  {

    "course_id": 1,
    "course_name": "Programming in Python",
    "description": "A foundational course in python for data science.",
    "modules": [
        {
            "module_id": "CS101",
            "module_name": "Introduction to Python",
            "description": "Basics of Python programming."
        },
           ]
}
```


Result: Success

Test Case: Bad Request - Course Not Found

API being tested: `/courses/<int:course_id>` (GET method)

Inputs:

`course_id`: 100

Expected Output:

Status Code: 404 (Not Found)

JSON:


```
{
    "message": "Course not found"
}
```


Actual Output:

Status Code: 404 (Not Found)

JSON:

```
{
    "message": "Course not found"
}
```

Result: Success

Test Case: Add a Course

API being tested: `/courses` (POST method)

Inputs:

```
{
    "course_name": "Programming in Python",
    "description": "An in-depth course on python for data science"
}
```

Expected Output:

Status Code: 201 (Created)

JSON:

```
{
    "course_id": 1,
    "course_name": "Programming in Python",
    "description": "An in-depth course on python for data science."
}
```

Actual Output:

Status Code: 201 (Created)

JSON:

```
{
    "course_id": 1,
    "course_name": "Programming in Python",
    "description": "An in-depth course on python for data science."
}
```

Result: Success

Test Case: Update a Course
API being tested: `/courses/<int:course_id>` **(PUT method)**
Inputs:

```
{
    "course_name": "Python - 2",
    "description": "Changed the course name"
}
```

Expected Output:

Status Code: 200 (OK)

JSON:

```
{
    "course_id": 1,
    "course_name": "Python 2",
    "description": "Changed the course name"
}
```

Actual Output:

Status Code: 200 (OK)

JSON:

```
{
    "course_id": 1,
    "course_name": "Python 2",
    "description": "Changed the course name"
}
```

Result: Success

Test Cases for Modules

Test Case: Add a Module to a Course

API being tested: `/courses/<int:course_id>/modules` **(POST method)**

Inputs:

```
{
    "module_name": "OOPS",
    "description": "Basics of Object Oriented Programming.",
    "course_id": 102
}
```

Expected Output:

Status Code: 201 (Created)

JSON:

```
{
    "module_id": "1",
    "module_name": "OOPS",
    "description": "Basics of object oriented programming.",
    "course_id": 1
}
```

Actual Output:

Status Code: 201 (Created)

JSON:

```
{
    "module_id": "1",
    "module_name": "OOPS",
    "description": "Basics of object oriented programming.",
    "course_id": 1
}
```

Result: Success

Test Case: Delete a Module from a Course

API being tested: `/courses/<int:course_id>/modules/<int:module_id>`

(DELETE method)
Inputs:

`module_id`**:** 1

Expected Output:

Status Code: 200 (OK)

JSON:

```
{
    "message": "Module deleted successfully"
}
```

Actual Output:

Status Code: 200 (OK)

JSON:

```
{
    "message": "Module deleted successfully"
}
```

Result: Success

Test Case: Bad Request - Module Not Found
API being tested: `/courses/<int:course_id>/modules/<int:module_id>`
(DELETE method)
Inputs:

`module_id`: 100 (invalid module id)

Expected Output:

Status Code: 404 (Not Found)

JSON:

```
{
```

```
    "message": "Module not found"
}
```

Actual Output:

Status Code: 404 (Not Found)

JSON:

```
{
    "message": "Module not found"
}
```

Result: Success

Test Cases for Lectures

Test Case: Adding a lecture

API being tested: /lectures (POST method)

Inputs:

module_id: "CS101",
title: "Programming in Python",
content: "https://youtu.be/8ndsDXohLMQ?"

Expected Output:

Status Code: 201 (Created)

JSON:

```
{
  "id": 1,
  "module_id": "CS101",
  "title": "Programming in Python",
  "content": "https://youtu.be/8ndsDXohLMQ?"
}
```

Actual Output:

Status Code: 201 (Created)

JSON:

```
{
  "id": 1,
  "module_id": "CS101",
  "title": "Programming in Python",
```

"content": "https://youtu.be/8ndsDXohLMQ?"
}

Result: Success

Test Case: Successfully deleting a lecture
API being tested: /lectures/<int:lecture_id> (DELETE method)
Inputs:
Expected Output:
Status Code: 200 (OK)
JSON:
{'message': 'Lecture deleted'}

Actual Output:
Status Code: 200 (OK)
JSON:
{'message': 'Lecture deleted'}

Result: Success

Test Case: Bad Request (Deleting a lecture with lecture id that does not exist)
API being tested:/lectures/<int:lecture_id> (DELETE method)
Inputs:
lecture_id : 100

Expected Output:
Status Code: 404 (Not Found)
JSON:
{
"message": "Lecture not found"
}

Actual Output:
Status Code: 404 (Not Found)
JSON:
{

```
  "message": "Lecture not found"
}
```


Result: Success

Test Cases for Assignments

Test Case: Accessing an assignment.

API being tested: /assignments/<int:assignment_id> (GET method)

Inputs:

assignment_id: 1

Expected Output:

Status Code: 200

JSON:


```
{
  "id": "1",
  "title": "Sample Assignment",
  "description": "This is a sample assignment",
  "due_date": "2024-08-30T00:00:00",
  "created_at": "2024-08-09T10:00:00",
  "updated_at": "2024-08-09T10:00:00"
}
```


Actual Output:

Status Code: 200

JSON:


```
{
  "id": "1",
  "title": "Sample Assignment",
  "description": "This is a sample assignment",
  "due_date": "2024-08-30T00:00:00",
  "created_at": "2024-08-09T10:00:00",
  "updated_at": "2024-08-09T10:00:00"
}
```


Result: Success

Test Case: Successfully creating a new assignment.

API being tested: /assignments (POST method)

Inputs:

title: "New Assignment",
description: "This is a new assignment"

Expected Output:

Status Code: 200

JSON:


```
{
   "id": "1",
   "title": "New Assignment",
   "description": "This is a new assignment",
   "due_date": null,
   "created_at": "2024-08-09T10:15:00",
   "updated_at": "2024-08-09T10:15:00"
}
```


Actual Output:

Status Code: 200

JSON:


```
{
   "id": "1",
   "title": "New Assignment",
   "description": "This is a new assignment",
   "due_date": null,
   "created_at": "2024-08-09T10:15:00",
   "updated_at": "2024-08-09T10:15:00"
}
```


Result: Success

Test Case: Successfully updating an assignment.

API being tested: /assignments/<int:assignment_id> (PUT method)

Inputs:

assignment_id:1
title: "Updated Assignment",
description: "This assignment has been updated"

Expected Output:
Status Code: 200
JSON:

```
{
   "id": "1",
   "title": "Updated Assignment",
   "description": "This assignment has been updated",
   "due_date": "2024-08-30T00:00:00",
   "created_at": "2024-08-09T10:00:00",
   "updated_at": "2024-08-09T10:20:00"
}
```


Actual Output:
Status Code: 200
JSON:

```
{
   "id": "1",
   "title": "Updated Assignment",
   "description": "This assignment has been updated",
   "due_date": "2024-08-30T00:00:00",
   "created_at": "2024-08-09T10:00:00",
   "updated_at": "2024-08-09T10:20:00"
}
```

Result: Success

Test Cases for Programming Assignments

Test Case: Successfully getting a programming assignment.

APIbeing tested:
/programming_assignments/<int:programming_assignment_id> (GET method)
Inputs:
programming_assignment_id: 1

Expected Output:
Status Code: 200
JSON:

```
{
   "id": "1",
   "assignment_id": "2",
   "editor_content": "print('Hello, World!')",
   "sandbox_environment": "Python 3.8",
   "help_button_enabled": true,
   "created_at": "2024-08-09T10:30:00",
   "updated_at": "2024-08-09T10:30:00"
}
```

Actual Output:
Status Code: 200
JSON:

```
{
   "id": "1",
   "assignment_id": "2",
   "editor_content": "print('Hello, World!')",
   "sandbox_environment": "Python 3.8",
   "help_button_enabled": true,
   "created_at": "2024-08-09T10:30:00",
   "updated_at": "2024-08-09T10:30:00"
}
```

Result: Success

Test Case: Successfully creating a programming assignment
API being tested: /programming_assignments (POST method)
Inputs:

assignment_id: 2,
language: "Python"

Expected Output:
Status Code: 201 (Created)
JSON:

```
{
   "id": "2",
   "assignment_id": "2",
   "editor_content": null,
   "sandbox_environment": "Python 3.8",
   "help_button_enabled": false,
   "created_at": "2024-08-09T10:35:00",
   "updated_at": "2024-08-09T10:35:00"
}
```

Actual Output:
Status Code: 201 (Created)
JSON:

```
{
   "id": "2",
   "assignment_id": "2",
   "editor_content": null,
   "sandbox_environment": "Python 3.8",
   "help_button_enabled": false,
   "created_at": "2024-08-09T10:35:00",
   "updated_at": "2024-08-09T10:35:00"
}
```


Result: Success

Test Case: Updating a Programming Assignment.
API being tested:
/programming_assignments/<int:programming_assignment_id> (PUT method)
Inputs:

programming_assignment_id: 1,
assignment_id: 2,
language: "Python 3"

Expected Output:
Status Code: 200
JSON:

```
{
   "id": "1",
   "assignment_id": "2",
   "editor_content": "print('Updated Content')",
   "sandbox_environment": "Python 3.9",
   "help_button_enabled": true,
   "created_at": "2024-08-09T10:30:00",
   "updated_at": "2024-08-09T10:40:00"
}
```


Actual Output:
Status Code: 200
JSON:

```
{
   "id": "1",
   "assignment_id": "2",
   "editor_content": "print('Updated Content')",
   "sandbox_environment": "Python 3.9",
   "help_button_enabled": true,
   "created_at": "2024-08-09T10:30:00",
   "updated_at": "2024-08-09T10:40:00"
}
```

Result: Success

Test Cases for test cases of Programming Assignments and Submissions

Test Case: Successfully getting a Test Case.

API being tested:/test_cases/<int:test_case_id> (GET method)

Inputs:

test_case_id: 1

Expected Output:

Status Code: 200

JSON:


```
{
 "id": "1",
 "assignment_id": "1",
 "input": "4",
 "expected_output": "16",
 "created_at": "2024-08-09T11:00:00",
 "updated_at": "2024-08-09T11:00:00"
}
```


Actual Output:

Status Code: 200

JSON:


```
{
 "id": "1",
 "assignment_id": "1",
 "input": "4",
 "expected_output": "16",
 "created_at": "2024-08-09T11:00:00",
 "updated_at": "2024-08-09T11:00:00"
}
```


Result: Success

Test Case: Successfully creating a Test Case.

API being tested: /test_cases (POST method)

Inputs:


```
{
 "assignment_id": 1001,
```

```
 "input": "5",
 "output": "25"
}
```


Expected Output:
Status Code: 200
JSON:

```
{
 "id": "2",
 "assignment_id": "1",
 "input": "5",
 "expected_output": "25",
 "created_at": "2024-08-09T11:05:00",
 "updated_at": "2024-08-09T11:05:00"
}
```


Actual Output:
Status Code: 200
JSON:

```
{
 "id": "2",
 "assignment_id": "1",
 "input": "5",
 "expected_output": "25",
 "created_at": "2024-08-09T11:05:00",
 "updated_at": "2024-08-09T11:05:00"
}
```


Result: Success

Test Case: Bad Request for test cases

API being tested: /test_cases/<int:test_case_id> (POST method)
Inputs:

```
{
```

"assignment_id": 1,
"input": "?",
"output": "error"
}

Expected Output:
Status Code: 400 **(**BAD Request**)**
JSON:
{
"message": "Bad Request: Input must be a valid string."
}

Actual Output:
Status Code: 400 **(**BAD Request**)**
JSON:
{
"message": "Bad Request: Input must be a valid string."
}

Result: Success

Test Case: Successfully getting a Submission.
API being tested:/submissions/<int:submission_id> (GET method)
Inputs:
submission_id: 1

Expected Output:
Status Code: 200
JSON:
{
"id": 1,
"assignment_id": "1",
"student_id": "2001",
"content": "print(4 ** 2)",
"submission_date": "2024-08-09T11:10:00",

```
  "grade": null,
  "feedback": null,
  "created_at": "2024-08-09T11:10:00",
  "updated_at": "2024-08-09T11:10:00"
}
```


Actual Output:
Status Code: 200
JSON:

```
{
  "id": 1,
  "assignment_id": "1",
  "student_id": "2001",
  "content": "print(4 ** 2)",
  "submission_date": "2024-08-09T11:10:00",
  "grade": null,
  "feedback": null,
  "created_at": "2024-08-09T11:10:00",
  "updated_at": "2024-08-09T11:10:00"
}
```


Result: Success

Test Case: Creating a submission

API being tested: /post/submissions (POST method)
Inputs:

```
{
  "assignment_id": 1,
  "student_id": "2001",
  "content": "print(5 ** 2)"
}
```


Expected Output:
Status Code: 200
JSON:

{
 "id": 2,
 "assignment_id": "1",
 "student_id": "2001",
 "content": "print(5 ** 2)",
 "submission_date": "2024-08-09T11:15:00",
 "grade": null,
 "feedback": null,
 "created_at": "2024-08-09T11:15:00",
 "updated_at": "2024-08-09T10:15:00"
}

Actual Output:

Status Code: 200

JSON:

{
 "id": 2,
 "assignment_id": "1",
 "student_id": "2001",
 "content": "print(5 ** 2)",
 "submission_date": "2024-08-09T11:15:00",
 "grade": null,
 "feedback": null,
 "created_at": "2024-08-09T11:15:00",
 "updated_at": "2024-08-09T11:15:00"
}

Result: Success

Test Cases for Gen AI Features

Test Case: Successfully getting a Virtual Instructor Query.

API being tested:
/virtual_instructor_queries/<string:course_id>/<string:student_id> (GET method)
Inputs:
{ "course_id": "course123", "student_id": "456" }
Expected Output:
Status Code: 200
JSON:

[
  {
    "id": "1",
    "student_id": "456",
    "course_id": "1",
    "gen_query": "Explain the concept of linear regression.",
    len("response") >0
  }
]

Actual Output:
Status Code: 200
JSON:

{
  "id": "1",
  "student_id": "456",
  "course_id": "1",
  "gen_query": "Explain the concept of linear regression.",
  "response": "> Linear regression is a fundamental statistical technique used in data science to model the relationship between a dependent variable (the one you want to predict) and one or more independent variables (the ones you use to make the prediction). \n> \n> Imagine you want to predict the price of a house based on its size. Linear regression assumes a linear relationship: as the size increases, the price increases proportionally. It finds the \"best-fit\" line that minimizes the distance between the actual house prices and the line's predictions.\n> \n> Mathematically, this line is represented by an equation: y = mx + c, where 'y' is the predicted price, 'x' is the size, 'm' is the slope (how much the price changes per unit

increase in size), and 'c' is the intercept (the price when the size is zero). \n> \n> Linear regression helps us understand the relationship between variables, make predictions, and identify important factors influencing a phenomenon. \n",
  "created_at": "Fri, 09 Aug 2024 23:36:35 -0000"
}
Result: Success

Test Case: Successfully posting a Virtual Instructor Query.
API being tested: /virtual_instructor_queries (POST method)
Inputs:
JSON
{
  "student_id": "456",
  "course_id": "1",
  "gen_query": "Explain the concept of linear regression."
}

Expected Output:
Status Code: 201
JSON:
{
  "id": "1",
  "student_id": "456",
  "course_id": "1",
  "gen_query": "Explain the concept of linear regression.",
  len("response") > 0
}
Actual Output:
Status Code: 201
JSON:
{
  "id": "1",
  "student_id": "456",
  "course_id": "1",
  "gen_query": "Explain the concept of linear regression.",

"response": "> Linear regression is a fundamental statistical technique used in data science to model the relationship between a dependent variable (the one you want to predict) and one or more independent variables (the ones you use to make the prediction). \n> \n> Imagine you want to predict the price of a house based on its size. Linear regression assumes a linear relationship: as the size increases, the price increases proportionally. It finds the \"best-fit\" line that minimizes the distance between the actual house prices and the line's predictions.\n> \n> Mathematically, this line is represented by an equation: y = mx + c, where 'y' is the predicted price, 'x' is the size, 'm' is the slope (how much the price changes per unit increase in size), and 'c' is the intercept (the price when the size is zero). \n> \n> Linear regression helps us understand the relationship between variables, make predictions, and identify important factors influencing a phenomenon. \n",
"created_at": "Fri, 09 Aug 2024 23:36:35 -0000"
}

Result: Success

Inputs:

JSON

{
"student_id": "456",
"course_id": "12",
"gen_query": "Explain the concept of linear regression."
}

Expected Output:

Status Code: 201

JSON:

{
"id": "1",
"student_id": "456",
"course_id": "12",
"gen_query": "Explain the concept of linear regression.",
len("response") > 0
}

Actual Output:
Status Code: 404
JSON:
{
"message": "Course not found"
}

Result: Success

Test Case: Successfully deleting a Virtual Instructor Query.
API being tested: /virtual_instructor_queries/<int:query_id> (DELETE method)
Inputs:
{ "course_id": "course123", "student_id": "student456" }
Expected Output:
Status Code: 200
JSON:
{
"message": "All queries deleted"
}

Actual Output:
Status Code: 200
JSON:
{
"message": "All queries deleted"
}

Result: Success
Inputs:
{ "course_id": "course123", "student_id": "student456" }
Expected Output:
Status Code: 200
JSON:
{

"message": "All queries deleted"
}

Actual Output:
Status Code: 200
JSON:
{
"message": "All queries deleted"
}

Result: Success

Test Case: Successfully getting a Student Planner.
API being tested: /student_planners/<int:planner_id> (GET method)
Inputs:
{ "student_id": "1", "content_type": "question" }
Expected Output:
Status Code: 200
JSON:
[{
"id": 1,
"student_id": "1",
len("response")>0
"level": "intermediate",
"content_type": "question",
"gen_query": "How to become a data scientist."
}]
Actual Output:
Status Code: 200
JSON:
[{
"id": 1,
"student_id": "1",
len("response")>0
"level": "intermediate",

"content_type": "question",
"gen_query": "How to become a data scientist."
}]

Result: Success

Test Case: Successfully posting a Student Planner query.

API being tested: /student_planners (POST method)

Inputs:

{
"student_id": "1",
"level": "intermediate",
"content_type": "question",
"gen_query": "How to become a data scientist."
}

Expected Output:

Status Code: 201

JSON:

{
"id": 1,
"student_id": "1",
len("response")>0
"level": "intermediate",
"content_type": "question",
"gen_query": "How to become a data scientist."
}

Actual Output:

Status Code: 201

JSON:

{
"id": 1,
"student_id": "1",
len("response")>0
"level": "intermediate",
"content_type": "question",

"gen_query": "How to become a data scientist."
}
Result: Success

Inputs:
{
"student_id": "2",
"level": "intermediate",
"content_type": "question",
"gen_query": "How to become a data scientist."
}

Expected Output:
Status Code: 404
JSON:
{
"message": "No enrolled courses found for the student"
}
Actual Output:
Status Code: 404
JSON:
{
"message": "No enrolled courses found for the student"
}
Result: Success

Test Case: Successfully deleting a Student Planner query.
API being tested: /student_planners/<int:query_id> (DELETE method)
Inputs:
{ "query_id": "1" }

Expected Output:
Status Code: 200
JSON:

{
"message": "Curation deleted successfully"
}

Actual Output:

Status Code: 200

JSON:

{
"message": "Curation deleted successfully"
}

Result: Success

Instructor Component Testing:

Test Case: Successfully getting an Instructor Content Planner response.

API being tested: /instructor_content_planners/<int:query_id> (GET method)

Inputs:

{ "instructor_id": "instructor789", "course_id": "1", "content_type": "resource" }

Expected Output:

Status Code: 200

JSON:

[
{
"id": "1",
"instructor_id": "instructor789",
"course_id": "1",
"content_type": "resource",
len("gen_ai_resources") > 0
"gen_ai_questions": null,
"gen_ai_answers": null,
"gen_ai_question_detail": null,
"difficulty_level": "intermediate"
}
]

Actual Output:
Status Code: 200
JSON:
[
{
"id": "1",
"instructor_id": "instructor789",
"course_id": "1",
"content_type": "resource",
len("gen_ai_resources") > 0
"gen_ai_questions": null,
"gen_ai_answers": null,
"gen_ai_question_detail": null,
"difficulty_level": "intermediate"
}
]
Result: Success

Test Case: Successfully submitting an Instructor Content Planner Query.
API being tested: /instructor_content_planners (POST method)
Inputs:
{
"instructor_id": "instructor789",
"course_id": "1",
"gen_query": "Introduction to Machine Learning",
"content_type": "resource",
"difficulty_level": "intermediate"
}

Expected Output:
Status Code: 201
JSON:
{
"id": "1",

```
  "instructor_id": "instructor789",
  "course_id": "1",
  "content_type": "resource",
  len("gen_ai_resources") > 0
  "gen_ai_questions": null,
  "gen_ai_answers": null,
  "gen_ai_question_detail": null,
  "difficulty_level": "intermediate"
}
```


Actual Output:
Status Code: 201
JSON:


```
{
  "id": "1",
  "instructor_id": "instructor789",
  "course_id": "1",
  "content_type": "resource",
  len("gen_ai_resources") > 0
  "gen_ai_questions": null,
  "gen_ai_answers": null,
  "gen_ai_question_detail": null,
  "difficulty_level": "intermediate"
}
```


Result: Success

Test Case: Successfully deleting an Instructor Content Planner Query.
API being tested: /instructor_content_planners/<int:planner_id> (DELETE method)
Inputs:

```
{ "query_id": "1" }
```


Expected Output:
Status Code: 200

JSON:
{
"message": "Content deleted successfully"
}

Actual Output:
Status Code: 200
JSON:
{
"message": "Content deleted successfully"
}
Result: Success

2. Test Summary and Pytest Screenshots:

2.1 Configuration Testing Component:

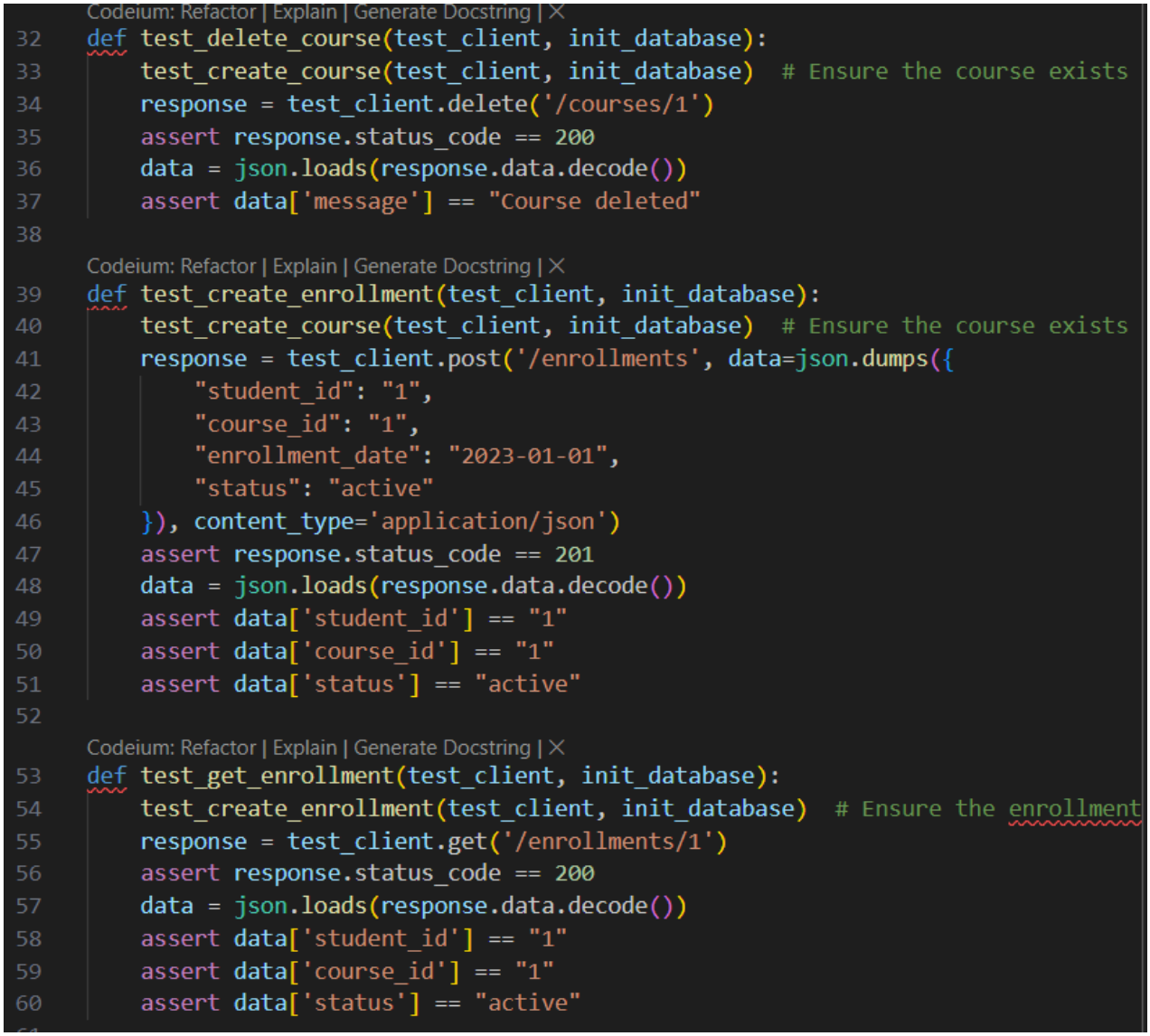
(a) <u>Screenshot 1:</u>

```python
# tests/functional/conftest.py
import pytest
from main import create_app
from application.config import TestingConfig
from application.database import db
from application.models import User, Course

Codeium: Refactor | Explain | Generate Docstring | ×
@pytest.fixture(scope='module')
def test_client():
    app, _ = create_app()
    app.config.from_object(TestingConfig)
    testing_client = app.test_client()

    with app.app_context():
        db.create_all()
        yield testing_client
        db.drop_all()

Codeium: Refactor | Explain | Generate Docstring | ×
@pytest.fixture(scope='module')
def init_database():
    app, _ = create_app()
    app.config.from_object(TestingConfig)

    with app.app_context():
        db.create_all()

        # Create test data
        user1 = User(email="test1@example.com", username="testuser1", password="password")
        user2 = User(email="test2@example.com", username="testuser2", password="password")
        db.session.add(user1)
        db.session.add(user2)
        db.session.commit()

        course1 = Course(title="Test Course 1", description="This is a test course.")
        db.session.add(course1)
        db.session.commit()

        yield db
        db.session.remove()
        db.drop_all()
```

2.2 API Testing Component:

(b) <u>Screenshot 2:</u>

```
Codeium: Refactor | Explain | Generate Docstring | ×
def test_create_course(test_client, init_database):
    response = test_client.post('/courses', data=json.dumps({
        "title": "New Test Course",
        "description": "A new test course description"
    }), content_type='application/json')
    assert response.status_code == 201
    data = json.loads(response.data.decode())
    assert data['title'] == "New Test Course"
    assert data['description'] == "A new test course description"

Codeium: Refactor | Explain | Generate Docstring | ×
def test_get_course(test_client, init_database):
    test_create_course(test_client, init_database)  # Ensure the course exists
    response = test_client.get('/courses/1')
    assert response.status_code == 200
    data = json.loads(response.data.decode())
    assert data['title'] == "New Test Course"
    assert data['description'] == "A new test course description"

Codeium: Refactor | Explain | Generate Docstring | ×
def test_update_course(test_client, init_database):
    test_create_course(test_client, init_database)  # Ensure the course exists
    response = test_client.put('/courses/1', data=json.dumps({
        "title": "Updated Test Course",
        "description": "Updated test course description"
    }), content_type='application/json')
    assert response.status_code == 200
    data = json.loads(response.data.decode())
    assert data['title'] == "Updated Test Course"
    assert data['description'] == "Updated test course description"
```



```
Codeium: Refactor | Explain | Generate Docstring | ×
def test_delete_course(test_client, init_database):
    test_create_course(test_client, init_database)  # Ensure the course exists
    response = test_client.delete('/courses/1')
    assert response.status_code == 200
    data = json.loads(response.data.decode())
    assert data['message'] == "Course deleted"

Codeium: Refactor | Explain | Generate Docstring | ×
def test_create_enrollment(test_client, init_database):
    test_create_course(test_client, init_database)  # Ensure the course exists
    response = test_client.post('/enrollments', data=json.dumps({
        "student_id": "1",
        "course_id": "1",
        "enrollment_date": "2023-01-01",
        "status": "active"
    }), content_type='application/json')
    assert response.status_code == 201
    data = json.loads(response.data.decode())
    assert data['student_id'] == "1"
    assert data['course_id'] == "1"
    assert data['status'] == "active"

Codeium: Refactor | Explain | Generate Docstring | ×
def test_get_enrollment(test_client, init_database):
    test_create_enrollment(test_client, init_database)  # Ensure the enrollment
    response = test_client.get('/enrollments/1')
    assert response.status_code == 200
    data = json.loads(response.data.decode())
    assert data['student_id'] == "1"
    assert data['course_id'] == "1"
    assert data['status'] == "active"
```

© **Screenshot 3:**

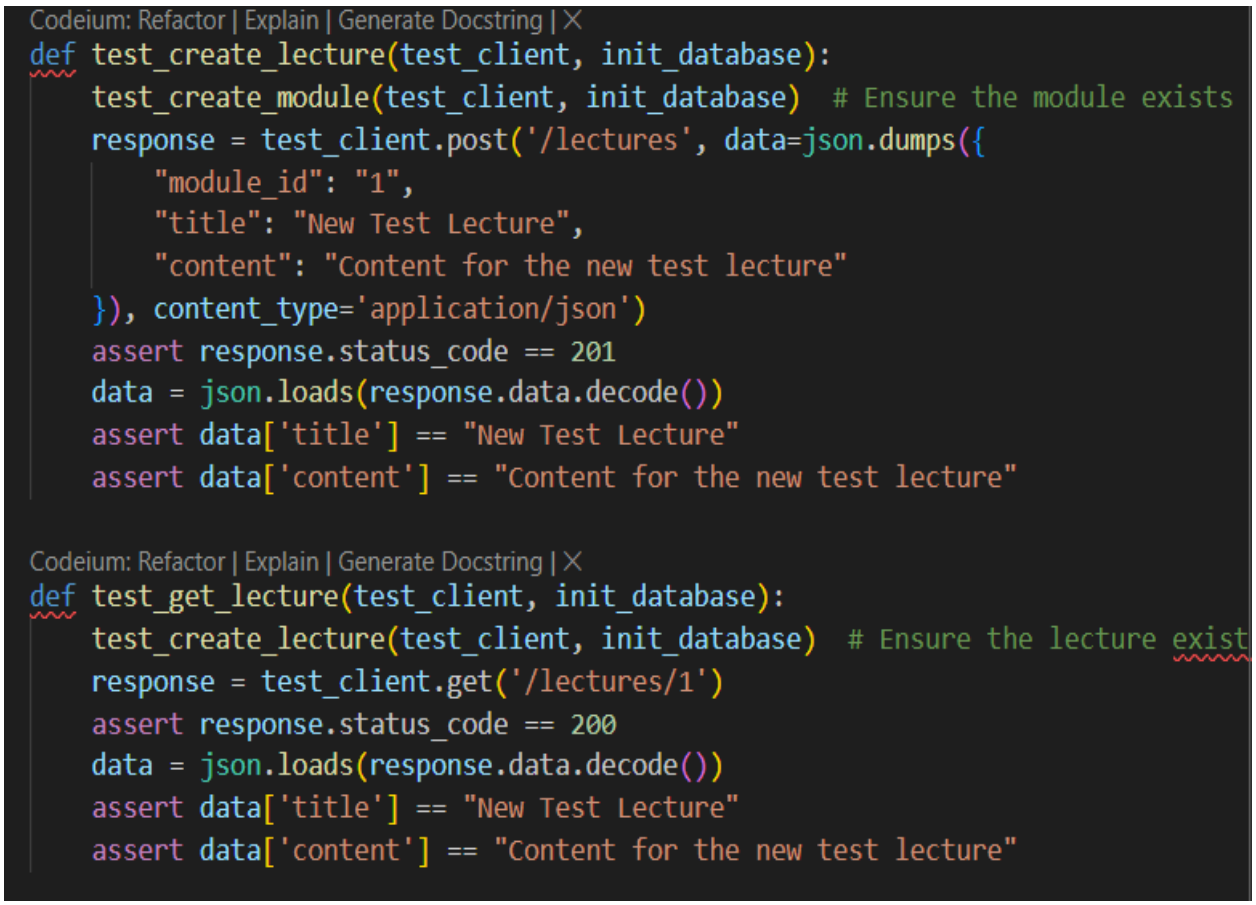
(d) Screenshot 4:

```python
def test_update_enrollment(test_client, init_database):
    test_create_enrollment(test_client, init_database)  # Ensure the enrollment
    response = test_client.put('/enrollments/1', data=json.dumps({
        "student_id": "2",
        "course_id": "1",
        "enrollment_date": "2023-01-01",
        "status": "completed"
    }), content_type='application/json')
    assert response.status_code == 200
    data = json.loads(response.data.decode())
    assert data['student_id'] == "2"
    assert data['course_id'] == "1"
    assert data['status'] == "completed"

Codeium: Refactor | Explain | Generate Docstring | ×
def test_delete_enrollment(test_client, init_database):
    test_create_enrollment(test_client, init_database)  # Ensure the enrollment
    response = test_client.delete('/enrollments/1')
    assert response.status_code == 200
    data = json.loads(response.data.decode())
    assert data['message'] == "Enrollment deleted"

Codeium: Refactor | Explain | Generate Docstring | ×
def test_create_module(test_client, init_database):
    test_create_course(test_client, init_database)  # Ensure the course exists
    response = test_client.post('/modules', data=json.dumps({
        "course_id": "1",
        "title": "New Test Module",
        "description": "A new test module description",
        "week": 1,
        "created_at": "2023-01-01",
        "updated_at": "2023-01-01"
    }), content_type='application/json')
    assert response.status_code == 201
    data = json.loads(response.data.decode())
    assert data['title'] == "New Test Module"
```

(e) Screenshot 5:

```
98  def test_get_module(test_client, init_database):
99      test_create_module(test_client, init_database)  # Ensure the module exists
100     response = test_client.get('/modules/1')
101     assert response.status_code == 200
102     data = json.loads(response.data.decode())
103     assert data['title'] == "New Test Module"
104     assert data['description'] == "A new test module description"
105
        Codeium: Refactor | Explain | Generate Docstring | ×
106 def test_update_module(test_client, init_database):
107     test_create_module(test_client, init_database)  # Ensure the module exists
108     response = test_client.put('/modules/1', data=json.dumps({
109         "course_id": "1",
110         "title": "Updated Test Module",
111         "description": "Updated test module description",
112         "week": 2,
113         "created_at": "2023-01-01",
114         "updated_at": "2023-01-01"
115     }), content_type='application/json')
116     assert response.status_code == 200
117     data = json.loads(response.data.decode())
118     assert data['title'] == "Updated Test Module"
119     assert data['description'] == "Updated test module description"
120
        Codeium: Refactor | Explain | Generate Docstring | ×
121 def test_delete_module(test_client, init_database):
122     test_create_module(test_client, init_database)  # Ensure the module exists
123     response = test_client.delete('/modules/1')
124     assert response.status_code == 200
125     data = json.loads(response.data.decode())
126     assert data['message'] == "Module deleted"
127
```

(f) Screenshot 6:



```
Codeium: Refactor | Explain | Generate Docstring | X
def test_create_lecture(test_client, init_database):
    test_create_module(test_client, init_database)  # Ensure the module exists
    response = test_client.post('/lectures', data=json.dumps({
        "module_id": "1",
        "title": "New Test Lecture",
        "content": "Content for the new test lecture"
    }), content_type='application/json')
    assert response.status_code == 201
    data = json.loads(response.data.decode())
    assert data['title'] == "New Test Lecture"
    assert data['content'] == "Content for the new test lecture"

Codeium: Refactor | Explain | Generate Docstring | X
def test_get_lecture(test_client, init_database):
    test_create_lecture(test_client, init_database)  # Ensure the lecture exist
    response = test_client.get('/lectures/1')
    assert response.status_code == 200
    data = json.loads(response.data.decode())
    assert data['title'] == "New Test Lecture"
    assert data['content'] == "Content for the new test lecture"
```

2.3 Unit Testing Component:

(g) Screenshot 7:

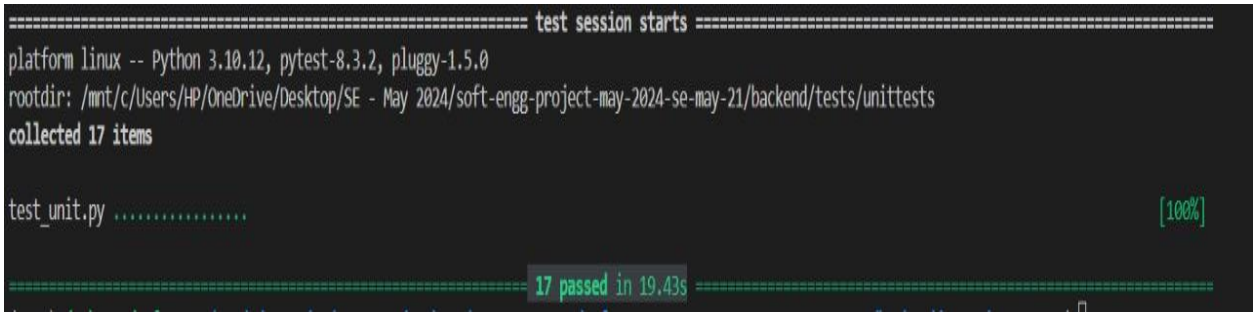
```python
import pytest
from application.models import (
    User, Course, Enrollment, Module, Lecture, Note, Assignment,
    ProgrammingAssignment, Question, Submission,
    VirtualInstructorQuery, InstructorContentPlanner, CourseStatistics,
    AdminCourseManagement, StudentDetailsManagement, AssignmentStatus, StudentCuration
)
from application.database import db

Codeium: Refactor | Explain | Generate Docstring | X
def test_user_model():
    user = User(email="test@example.com", username="testuser", password="password")
    assert user.email == "test@example.com"
    assert user.username == "testuser"
    assert user.password == "password"

Codeium: Refactor | Explain | Generate Docstring | X
def test_course_model():
    course = Course(title="Test Course", description="Test Course Description")
    assert course.title == "Test Course"
    assert course.description == "Test Course Description"

Codeium: Refactor | Explain | Generate Docstring | X
def test_enrollment_model():
    enrollment = Enrollment(student_id="1", course_id="1", enrollment_date="2023-01-01", status="active")
    assert enrollment.student_id == "1"
    assert enrollment.course_id == "1"
    assert enrollment.enrollment_date == "2023-01-01"
    assert enrollment.status == "active"
```

(h) Screenshot 8:

```
29  def test_module_model():
30      module = Module(course_id="1", title="Test Module", description="Test Module Description", week=1)
31      assert module.course_id == "1"
32      assert module.title == "Test Module"
33      assert module.description == "Test Module Description"
34      assert module.week == 1
35
    Codeium: Refactor | Explain | Generate Docstring | X
36  def test_lecture_model():
37      lecture = Lecture(module_id="1", title="Test Lecture", content="Test Lecture Content")
38      assert lecture.module_id == "1"
39      assert lecture.title == "Test Lecture"
40      assert lecture.content == "Test Lecture Content"
41
    Codeium: Refactor | Explain | Generate Docstring | X
42  def test_note_model():
43      note = Note(module_id="1", title="Test Note", content="Test Note Content")
44      assert note.module_id == "1"
45      assert note.title == "Test Note"
46      assert note.content == "Test Note Content"
47
    Codeium: Refactor | Explain | Generate Docstring | X
48  def test_assignment_model():
49      assignment = Assignment(title="Test Assignment", description="Test Assignment Description")
50      assert assignment.title == "Test Assignment"
51      assert assignment.description == "Test Assignment Description"
52
    Codeium: Refactor | Explain | Generate Docstring | X
53  def test_programming_assignment_model():
54      programming_assignment = ProgrammingAssignment(assignment_id="1", editor_content="Editor content", sandbox_environment="Sandbox environment", he
55      assert programming_assignment.assignment_id == "1"
56      assert programming_assignment.editor_content == "Editor content"
57      assert programming_assignment.sandbox_environment == "Sandbox environment"
58      assert programming_assignment.help_button_enabled is True
59
```

2.4 Result Summary:

(i) Screenshot 9:


test session starts
platform linux -- Python 3.10.12, pytest-8.3.2, pluggy-1.5.0
rootdir: /mnt/c/Users/HP/OneDrive/Desktop/SE - May 2024/soft-engg-project-may-2024-se-may-21/backend/tests/unittests
collected 17 items
test_unit.py .....
[100%]
17 passed in 19.43s