SOFTWARE ENGINEERING PROJECT

JAN 2025 TERM - TEAM 1

MILESTONE - 5

PRESENTED BY:

Amit Kulkarni (23f1001947)
Anjali Galav (22f3002299)
Jyotiraditya Saha (21f2000759)
Kajol Singh (21f1001886)
Saima Zainab Shroff (21f3002151)
Sandeep Kumar (21f3002365)
Siddharth Umathe (22f2001536)

CONTENTS

AI AGENT APIs

/api/generate_topic_specific_questions

METHOD: POST

- **Test Case: Generate practice questions for a particular topic - default**

```
def test_generate_topic_specific_questions_success(client):
    """Test successful generation of topic-specific questions."""
    payload = {
        "topic": "Data Visualization Libraries",
        "num_questions": 3
    }
    response = client.post('/generate_topic_specific_questions',
json=payload)
    data = response.get_json()

    assert response.status_code == 200
    assert data['success'] is True
    assert 'questions' in data
    assert len(data['questions']) == 3
    assert 'question' in data['questions'][0]
    assert 'options' in data['questions'][0]
    assert 'answer' in data['questions'][0]
    assert 'explanation' in data['questions'][0]
```

Input Data

• JSON
{
"topic": "Data Visualization Libraries",
"num_questions": 3
}

Expected Output	Actual Output
• Status Code: 200 • JSON { "success": true, "questions": [{ "question": "...,	• Status Code: 200 • JSON { "success": true, "questions": [{ "question": "...,

AI AGENT APIs

"options": ["...", "...", "..."], "answer": "...", "explanation": "..." }] }	"options": ["...", "...", "..."], "answer": "...", "explanation": "..." }] }
Result: **Success**	

- **Test Case: Generate practice questions for a particular topic with invalid question count (zero)**

```
def test_generate_topic_specific_questions_zero_questions(client):
    """Test API with zero questions requested."""
    payload = {"topic": "Artificial Intelligence", "num_questions": 0}
    response = client.post('/generate_topic_specific_questions',
json=payload)
    data = response.get_json()

    assert response.status_code == 400
    assert data['success'] is False
    assert data['message'] == 'Number of questions must be at least 1'
```

Input Data	

• JSON
{
"topic": "Artificial Intelligence",
"num_questions": 0
}	

Expected Output	**Actual Output**

• Status Code: 400
• JSON
{
"success": false,
"message": "Number of questions must be at least 1"
}
	• Status Code: 400
	• JSON
	{
	"success": false,
	"message": "Number of questions must be at least 1"
	}

Result: **Success**	

AI AGENT APIs

- **Test Case: Generate practice questions for a particular topic with invalid question count (negative number)**

```
def test_generate_topic_specific_questions_negative_questions(client):
    """Test API with negative value for `num_questions`."""
    payload = {"topic": "Cyber Security", "num_questions": -3}
    response = client.post('/generate_topic_specific_questions',
json=payload)
    data = response.get_json()

    assert response.status_code == 400
    assert data['success'] is False
    assert data['message'] == 'Number of questions must be at least 1'
```

Input Data	

• JSON
{
"topic": "Cyber Security",
"num_questions": -3
}	

Expected Output	**Actual Output**

• Status Code: 400
• JSON
{
"success": false,
"message": "Number of questions must be at least 1"
}
	• Status Code: 400
	• JSON
	{
	"success": false,
	"message": "Number of questions must be at least 1"
	}

Result: **Success**	

- **Test Case: Generate practice questions for a particular topic topic is missing in the input payload**

```
def test_generate_topic_specific_questions_missing_topic(client):
    """Test API when topic is missing in the payload."""
    payload = {"num_questions": 3}
    response = client.post('/generate_topic_specific_questions',
json=payload)
```

AI AGENT APIs

```
    data = response.get_json()

    assert response.status_code == 400
    assert data['success'] is False
    assert data['message'] == 'Topic is required'
```

Input Data	

• JSON
{
"num_questions": 3
}	

Expected Output	**Actual Output**

• Status Code: 400
• JSON
{
"success": false,
"message": "Topic is required"
}
	• Status Code: 400
	• JSON
	{
	"success": false,
	"message": "Topic is required"
	}

Result: **Success**	

- **Test Case: Generate practice questions for a particular topic with invalid data types in the input payload**

```
def test_generate_topic_specific_questions_invalid_data_type(client):
    """Test API with invalid data types in the payload."""
    payload = {"topic": 12345, "num_questions": "three"}
    response = client.post('/generate_topic_specific_questions',
json=payload)
    data = response.get_json()

    assert response.status_code == 400
    assert data['success'] is False
    assert data['message'] == 'Invalid data type for num_questions'
```

AI AGENT APIs

Input Data	

• JSON
{
"topic": 12345,
"num_questions": "three"
}	

Expected Output	**Actual Output**

• Status Code: 400
• JSON
{
"success": false,
"message": "Invalid data type for num_questions"
}
	• Status Code: 400
	• JSON
	{
	"success": false,
	"message": "Invalid data type for num_questions"
	}

Result: **Success**	

/api/video_summarizer

METHOD: POST

- **Test Case: Generate video summary with a valid lecture ID - default**

```python
def test_video_summarizer_success(client):
    """Test successful video summarization with valid lecture_id."""

    payload = {"lecture_id": "2"}
    response = client.post('/video_summarizer', json=payload)
    data = response.get_json()

    assert response.status_code == 200
    assert data['success'] is True
```

AI AGENT APIs

Input Data	

• JSON
{
"lecture_id": "2"
}	

Expected Output	**Actual Output**

• Status Code: 200
• JSON
{
"success": true,
"message": "Summary generated successfully",
"summary": "..."
}
	• Status Code: 200
	• JSON
	{
	"success": true,
	"message": "Summary generated successfully",
	"summary": "..."
	}

Result: **Success**	

- **Test Case: Generate video summary with missing lecture ID**

```python
def test_video_summarizer_missing_lecture_id(client):
    """Test API when `lecture_id` is missing in the payload."""
    payload = {}
    response = client.post('/video_summarizer', json=payload)
    data = response.get_json()

    assert response.status_code == 400
    assert data['success'] is False
    assert data['message'] == 'lecture_id is required'
```

Input Data	
• JSON {}	
Expected Output	**Actual Output**

• Status Code: 400
• JSON
{
"success": false,
"message": "lecture_id is required"
}
	• Status Code: 400
	• JSON
	{
	"success": false,
	"message": "lecture_id is required",
	}

Result: **Success**	

AI AGENT APIs

- **Test Case: Generate video summary with non-existing lecture ID**

```python
def test_video_summarizer_lecture_not_found(client):
    """Test API when the requested lecture_id is not found."""

    payload = {"lecture_id": "99_99"}
    response = client.post('/video_summarizer', json=payload)
    data = response.get_json()

    assert response.status_code == 404
    assert data['success'] is False
    assert data['message'] == 'Lecture not found'
```

Input Data	

• JSON
{
"lecture_id": "99_99"
}	

Expected Output	**Actual Output**

• Status Code: 400
• JSON
{
"success": false,
"message": "Lecture not found"
}
	• Status Code: 400
	• JSON
	{
	"success": false,
	"message": "Lecture not found",
	}

Result: Success	

AI AGENT APIs

/api/chat_history

METHOD: POST

- **Test Case: Chat history stored successfully**

```
def test_chatbot_response(client):
    """Test if chatbot returns a valid response"""
    input_data = {
        "user_id": 1,
        "query": "What is machine learning?",
        "response": "Machine learning is a subfield of artificial
intelligence."
    }
    response = client.post('/chat_history', json=input_data)

    assert response.status_code == 201

    data = response.get_json()

    assert data["success"] is True
    assert data["message"] == "Chat history saved successfully"
    assert data["user_id"] == 1
```

Input Data	

• JSON
{
"user_id": 1,
"query": "What is machine learning?",
"response": "Machine learning is a subfield of artificial intelligence."
}	

Expected Output	**Actual Output**

• Status Code: 201
• JSON
{
"success": true,
"message": "Chat history saved successfully",
"file_path": "..."
}
	• Status Code: 201
	• JSON
	{
	"success": true,
	"message": "Chat history saved successfully",
	"file_path": "..."
	}

Result: Success	

AI AGENT APIs

- **Test Case: Query not provided**

```python
def test_chatbot_missing_query(client):
    """Test chatbot response when the query is missing"""
    input_data = {
        "user_id": 2  # No query provided
    }
    response = client.post('/chat_history', json=input_data)

    assert response.status_code == 400

    data = response.get_json()

    assert data["success"] is False
    assert data["message"] == "Missing required fields"
```

Input Data	

• JSON
{
"user_id": 1,
"query": "What is machine learning?",
"response": "Machine learning is a subfield of artificial intelligence."
}	

Expected Output	**Actual Output**

• Status Code: 400
• JSON
{
"success": false,
"message": "Missing required fields"
}
	• Status Code: 400
	• JSON
	{
	"success": false,
	"message": "Missing required fields"
	}

Result: **Success**	

AI AGENT APIs

/api/explain_error

METHOD: POST

- **Test Case: Error explanation generated successfully**

```
def test_explain_error_success(client):
    """Test API with a valid code snippet that contains an error."""
    input_data = {
        "code_snippet": "print(1/0)"  # Causes ZeroDivisionError
    }
    response = client.post('/explain_error', json=input_data)

    assert response.status_code == 200
    assert response.json["success"] is True
    assert response.json["message"] == "Error explanation generated
successfully"
    assert "explanation" in response.json
    assert "divide by zero" in response.json["explanation"]
```

Input Data	

• JSON
{
"code_snippet": "print(1/0)"
}	

Expected Output	**Actual Output**

• Status Code: 200
• JSON
{
"success": true,
"message": "Error explanation generated successfully",
"explanation": "..."
}
	• Status Code: 200
	• JSON
	{
	"success": true,
	"message": "Error explanation generated successfully",
	"explanation": "..."
	}

Result: Success	

AI AGENT APIs

- Test Case: Missing code snippet

```python
def test_explain_error_missing_code_snippet(client):
    """Test API when the request body is missing the 'code_snippet'
key."""
    input_data = {}
    response = client.post('/explain_error', json=input_data)

    assert response.status_code == 400
    assert response.json["success"] is False
    assert response.json["message"] == "Code snippet is required"
```

Input Data	
• JSON {}	
Expected Output	**Actual Output**

• Status Code: 400
• JSON
{
"success": false,
"message": "Code snippet is required"
}
	• Status Code: 400
	• JSON
	{
	"success": false,
	"message": "Code snippet is required"
	}

Result: **Success**	

/api/generate_week_summary

METHOD: POST

- Test Case: Week summary generated successfully

```python
#Test case for successful summary generation when the week exists.
def test_generate_week_summary_success(client):
    response = client.post('/generate_week_summary', json={"week_id": 2})
    assert response.status_code == 200
    assert response.json['success'] is True
    assert 'summary' in response.json
```

AI AGENT APIs

Input Data	

• JSON
{
"week_id": 2
}	

Expected Output	**Actual Output**

• Status Code: 200
• JSON
{
"success": true,
"message": "Week summary generated successfully",
"week_id": 2,
"summary": "..."
}
	• Status Code: 200
	• JSON
	{
	"success": true,
	"message": "Week summary generated successfully",
	"week_id": 2,
	"summary": "..."
	}

Result: **Success**	

- **Test Case: Missing week ID**

```python
#Test case when `week_id` is missing in the request body.
def test_generate_week_summary_missing_week_id(client):
    response = client.post('/generate_week_summary', json={})

    assert response.status_code == 400
    assert response.json['success'] is False
    assert response.json['message'] == 'week_id is required’
```

Input Data	
• JSON {}	
Expected Output	**Actual Output**

• Status Code: 400
• JSON
{
"success": false,
"message": "week_id is required"
}
	• Status Code: 400
	• JSON
	{
	"success": false,
	"message": "week_id is required"
	}

Result: **Success**	

AI AGENT APIs

- Test Case: Missing week ID

```
#Test case when `week_id` does not exist in the database.
def test_generate_week_summary_non_existent_week(client):
    response = client.post('/generate_week_summary', json={"week_id":
999})

    assert response.status_code == 404
    assert response.json['success'] is False
    assert response.json['message'] == 'Week not found'
```

Input Data	

• JSON
{
"week_id": 999
}	

Expected Output	**Actual Output**

• Status Code: 404
• JSON
{
"success": false,
"message": "Week not found"
}
	• Status Code: 404
	• JSON
	{
	"success": false,
	"message": "Week not found"
	}

Result: **Success**	

/api/generate_mock

METHOD: POST

- Test Case: Mock test generated successfully

```
#Test case for successful generation of a mock test
def test_generate_mock_success(client):
    response = client.post('/generate_mock', json={'quiz_type': 'quiz1',
'num_questions': 10})

    assert response.status_code == 200
    data = response.get_json()
    assert data['success'] is True
    assert 'questions' in data
```

AI AGENT APIs

Input Data	

• JSON
{
"quiz_type": "quiz1",
"num_questions": 10
}	

Expected Output	**Actual Output**

• Status Code: 200
• JSON
{
"message": "Mock test generated successfully for quiz1",
"success": true,
"quiz_type": "quiz1",
"num_questions": 10,
"questions": [
{
"question": "...",
"options": [".."],
"correct_answer": "..."
}
]
}
	• Status Code: 200
	• JSON
	{
	"message": "Mock test generated successfully for quiz1",
	"success": true,
	"quiz_type": "quiz1",
	"num_questions": 10,
	"questions": [
	{
	"question": "...",
	"options": [".."],
	"correct_answer": "..."
	}
	]
	}

Result: **Success**	

- **Test Case: Missing quiz type field**

```python
# Test case: Missing quiz_type field
def test_generate_mock_missing_quiz_type(client):
    response = client.post('/generate_mock', json={
        "num_questions": 5
    })

    assert response.status_code == 400
    data = response.get_json()
    assert data['success'] is False
    assert data['message'] == 'quiz_type is required'
```

AI AGENT APIs

Input Data	

• JSON
{
"num_questions": 5
}	

Expected Output	**Actual Output**

• Status Code: 400
• JSON
{
"success": false,
"message": "quiz_type is required"
}
	• Status Code: 400
	• JSON
	{
	"success": false,
	"message": "quiz_type is required"
	}

Result: **Success**	

- **Test Case: Invalid quiz type**

```python
# Test case: Non-existent quiz_type
def test_generate_mock_non_existent_quiz_type(client):
    response = client.post('/generate_mock', json={
        'quiz_type': 'unknown_quiz',
        'num_questions': 5
    })

    assert response.status_code == 404
    data = response.get_json()
    assert data['success'] is False
    assert 'message' in data
```

Input Data	

• JSON
{
"quiz_type": "unknown_quiz",
"num_questions": 5
}	

Expected Output	**Actual Output**
• Status Code: 404	• Status Code: 404

AI AGENT APIs

• JSON
{
"success": false,
"message": "..."
}
	• JSON
	{
	"success": false,
	"message": "..."
	}

Result: **Success**	

/api/generate_notes

METHOD: POST

- **Test Case: Notes generated successfully**

```
#Test Case: Generate Notes Successfully
def test_generate_notes_success(client):
    response = client.post('/generate_notes', json={"topic":
"Reinforcement Learning"})

    assert response.status_code == 200
    assert response.json['success'] is True
    assert response.json['message'] == 'Notes generated successfully for
topic "Reinforcement Learning"'
    assert 'notes' in response.json
```

Input Data	

• JSON
{
"topic": "Reinforcement Learning"
}	

Expected Output	**Actual Output**

• Status Code: 200
• JSON
{
"success": true,
"message": "Notes generated successfully for topic "Reinforcement Learning"",
"notes": "..."
}
	• Status Code: 200
	• JSON
	{
	"success": true,
	"message": "Notes generated successfully for topic "Reinforcement Learning"",
	"notes": "..."
	}

Result: **Success**	

AI AGENT APIs

- Test Case: Missing topic

```python
#Test Case: missing topic
def test_generate_notes_missing_topic(client):
    response = client.post('/generate_notes', json={})

    assert response.status_code == 400
    assert response.json['success'] is False
    assert response.json['message'] == 'topic is required'
```

Input Data	
• JSON {}	
Expected Output	**Actual Output**

• Status Code: 400
• JSON
{
"success": false,
"message": "topic is required"
}
	• Status Code: 400
	• JSON
	{
	"success": false,
	"message": "topic is required"
	}

Result: **Success**	

/api/topic_recommendation

METHOD: POST

- Test Case: Topic recommendation based on incorrect quiz responses

```python
# Test case for successful recommendation generation based on incorrect
answers
def test_topic_recommendation_success(client):
    # Mock data for incorrect answers
    submitted_answers = [
        {"question_id": 1, "selected_option_id": 2},
        {"question_id": 2, "selected_option_id": 5}
    ]

    # Make the request
```

AI AGENT APIs

```
    response = client.post('/topic_recommendation',
                           json={"answers": submitted_answers})

    # Assertions
    assert response.status_code == 200
    assert response.json['success'] is True

    # Check structure exists
    assert 'message' in response.json
    assert 'suggestions' in response.json

    # Check suggestions has expected structure
    assert 'overall_assessment' in response.json['suggestions']
    assert 'topic_suggestions' in response.json['suggestions']
    assert 'general_tips' in response.json['suggestions']

    # Check content presence rather than exact equality
    assert len(response.json['suggestions']['topic_suggestions']) > 0
    assert len(response.json['suggestions']['general_tips']) > 0

    # For text fields, check that key phrases are present
    assert 'programming' in
response.json['suggestions']['overall_assessment'].lower()
```

Input Data	

• JSON
{
"answers": [
{"question_id": 1, "selected_option_id": 2},
{"question_id": 2, "selected_option_id": 5}
]
}	

Expected Output	**Actual Output**
• Status Code: 200 • JSON { "success": true, "message": "All answers	• Status Code: 200 • JSON { "success": true, "message": "All answers are

AI AGENT APIs

are correct! Great job!",
"suggestions": {
"overall_assessment":
"All questions were answered
correctly. Excellent
performance!",
"topic_suggestions":
[],
"general_tips": []
}
}
	correct! Great job!",
	"suggestions": {
	"overall_assessment":
	"All questions were answered
	correctly. Excellent
	performance!",
	"topic_suggestions":
	[],
	"general_tips": []
	}
	}

Result: **Success**	

- **Test Case: Topic recommendation when all answers are correct**

```
# Test case for when all answers are correct
def test_topic_recommendation_all_correct(client):
    # Mock data for correct answers
    submitted_answers = [
        {"question_id": 1, "selected_option_id": 3},
        {"question_id": 2, "selected_option_id": 6}
    ]

    # Make the request
    response = client.post('/topic_recommendation',
                           json={"answers": submitted_answers})

    # Assertions
    assert response.status_code == 200
    assert response.json['success'] is True
    assert "All answers are correct! Great job!" in
response.json['message']
    assert response.json['suggestions']['overall_assessment'] == "All
questions were answered correctly. Excellent performance!"
    assert len(response.json['suggestions']['topic_suggestions']) == 0
```

AI AGENT APIs

Input Data	

• JSON
{
"answers": [
{"question_id": 1, "selected_option_id": 3},
{"question_id": 2, "selected_option_id": 6}
]
}	

Expected Output	**Actual Output**

• Status Code: 200
• JSON
{
"success": true,
"message": "Topic suggestions generated successfully",
"suggestions": {
"overall_assessment": "You need to improve in programming basics",
"topic_suggestions": ["Variables", "Data Types"],
"general_tips": ["Practice daily"]
}
}
	• Status Code: 200
	• JSON
	{
	"success": true,
	"message": "Topic suggestions generated successfully",
	"suggestions": {
	"overall_assessment": "You need to improve in programming basics",
	"topic_suggestions": ["Variables", "Data Types"],
	"general_tips": ["Practice daily"]
	}
	}

Result: **Success**	

- **Test Case: Topic recommendation when answers are missing**

```python
# Test case for missing answers
def test_topic_recommendation_missing_answers(client):
    # Request with empty answers array
    response = client.post('/topic_recommendation', json={"answers": []})

    # Assertions
    assert response.status_code == 400
    assert response.json['success'] is False
    assert response.json['message'] == 'Answers are required'

    # Request with missing answers field
```

AI AGENT APIs

```
    response = client.post('/topic_recommendation', json={})

    # Assertions
    assert response.status_code == 400
    assert response.json['success'] is False
    assert response.json['message'] == 'Answers are required'
```

Input Data	

• JSON
{
"answers": []
}	

Expected Output	**Actual Output**

• Status Code: 400
• JSON
{
"success": false,
"message": "Answers are required"
}
	• Status Code: 400
	• JSON
	{
	"success": false,
	"message": "Answers are required"
	}

Result: **Success**	

/api/download_report

METHOD: POST

- **Test Case: Notes generated successfully**

```
# Test case for successful report generation and download
def test_download_report_success(client):
    # Test data with all required fields
    test_data = {
        "username": "testuser",
        "score": 85,
        "total": 100,
        "suggestions": ["Study more", "Practice regularly"],
        "questions": [{"question": "What is 2+2?", "answer": "4"}]
    }

    response = client.post('/download_report', json=test_data)
```

AI AGENT APIs

```
    assert response.status_code == 200
    # send_file was successfully called and returned our mock response
```

Input Data

• JSON
{
"username": "testuser",
"score": 85,
"total": 100,
"suggestions": [
"..."
],
"questions": [
{
"question_text": "...",
"options": [
"..."
],
"correct_answer": "..."
}
]
}

Expected Output	Actual Output

• Status Code: 200
• JSON
{
"success": true,
"message": "PDF generated and downloaded successfully",
"file_path": "..."
}
	• Status Code: 200
	• JSON
	{
	"success": true,
	"message": "PDF generated and downloaded successfully",
	"file_path": "..."
	}

Result: **Success**	

- **Test Case: Missing required fields**

```
# Test case for missing required fields
def test_download_report_missing_fields(client):
    # Test with missing username
    response = client.post('/download_report', json={"score": 85, "total":
100})
```

AI AGENT APIs

```
    assert response.status_code == 400
    assert response.json['success'] is False
    assert response.json['message'] == "Invalid input: 'username',
'score', and 'total' are required fields."
```

Input Data	

• JSON
{
"score": 85,
"total": 100
}	

Expected Output	**Actual Output**

• Status Code: 400
• JSON
{
"success":false,
"message": "PDF generated and downloaded successfully""Invalid input: 'username', 'score', and 'total' are required fields."
}
	• Status Code: 400
	• JSON
	{
	"success":false,
	"message": "PDF generated and downloaded successfully""Invalid input: 'username', 'score', and 'total' are required fields."
	}

Result: **Success**	

AUTHENTICATION APIs

/api/google_signup

METHOD: POST

- **Test Case: Signup with Google account authentication (New User) - default**

```
# Test case for successful signup (new user)
def test_google_signup_success(client):
    response = client.post('/google_signup', json={"access_token":
"valid_token"})

    assert response.status_code == 201
    assert response.json['Success'] is True
    assert response.json['access_token'] == 'mock_jwt_token'
    assert response.json['message'] == 'User registered successfully'
```

Input Data	

• JSON
{
"access_token": "valid_token"
}	

Expected Output	**Actual Output**

• Status Code: 201
• JSON
{
"Success": true,
"access_token": "mock_jwt_token",
"message": "User registered successfully"
}
	• Status Code: 201
	• JSON
	{
	"Success": true,
	"access_token": "mock_jwt_token",
	"message": "User registered successfully"
	}

Result: Success	

AUTHENTICATION APIs

- **Test Case: Signup with Google account authentication with missing access token**

```
# Test case for missing access token
def test_google_signup_missing_token(client):
    response = client.post('/google_signup', json={})

    assert response.status_code == 400
    assert response.json['Success'] is False
    assert response.json['message'] == 'Google access token is required'
```

Input Data	
• JSON {}	
Expected Output	**Actual Output**

• Status Code: 400
• JSON
{
"Success": false,
"message": "Google access token is required"
}
	• Status Code: 400
	• JSON
	{
	"Success": false,
	"message": "Google access token is required"
	}

Result: **Success**	

/api/google_login

METHOD: POST

- **Test Case: Login with Google account authentication - default**

```
# Test case for successful login
def test_google_login_success(client):
    response = client.post('/google_login', json={"access_token":
"valid_token"})

    assert response.status_code == 200
    assert response.json['Success'] is True
    assert response.json['access_token'] == 'mock_jwt_token'
    assert response.json['message'] == 'Login successful'
```

AUTHENTICATION APIs

Input Data	

• JSON
{
"access_token": "valid_token"
}	

Expected Output	**Actual Output**

• Status Code: 200
• JSON
{
"Success": true,
"access_token":
"mock_jwt_token",
"message": "Login
successful"
}
	• Status Code: 200
	• JSON
	{
	"Success": true,
	"access_token":
	"mock_jwt_token",
	"message": "Login
	successful"
	}

Result: **Success**	

- **Test Case: Login with Google account authentication with missing access token**

```python
# Test case for missing access token
def test_google_login_missing_token(client):
    response = client.post('/google_login', json={})

    assert response.status_code == 400
    assert response.json['Success'] is False
    assert response.json['message'] == 'Google access token is required'
```

Input Data	
• JSON {}	
Expected Output	**Actual Output**

• Status Code: 400
• JSON
{
"Success": false,
"message": "Google access
token is required"
}
	• Status Code: 400
	• JSON
	{
	"Success": false,
	"message": "Google access token
	is required"
	}

Result: **Success**	

WEEK APIs

/api/weeks

METHOD: GET

- **Test Case: Retrieve details of all weeks**

```python
def test_get_weeks_with_data(client)
    """Test GET /weeks when there are weeks in the database."""
    response = client.get('/weeks')
    assert response.status_code == 200
    data = response.get_json()

    assert data["success"] is True
    assert data["message"] == "Weeks retrieved successfully"
    assert isinstance(data["weeks"], list)
```

Expected Output	Actual Output

• Status Code: 201
• JSON
{
"success": true,
"message": "Weeks retrieved successfully",
"weeks": [
{
"id": 1,
"week_number": 1,
"title": "..."
},
// Additional Weeks...
]
}
	• Status Code: 201
	• JSON
	{
	"success": true,
	"message": "Weeks retrieved successfully",
	"weeks": [
	{
	"id": 1,
	"week_number": 1,
	"title": "..."
	},
	// Additional Weeks...
	]
	}

Result: Success	

WEEK APIs

/api/weeks/{week_id)

METHOD: GET

- **Test Case: Retrieve details of a specific week by its ID**

```python
def test_get_week_details_valid_id(client):
    #Test GET /weeks/<week_id> with a valid existing week ID.
    response = client.get('/weeks/1')

    assert response.status_code == 200
    data = response.get_json()

    assert data["success"] is True
    assert data["message"] == "Week details retrieved successfully"
    assert "week" in data
    assert data["week"]["id"] == 1
    assert "lectures" in data["week"]
    assert "assignments" in data["week"]
```

Endpoint	
GET /api/weeks/1	
Expected Output	**Actual Output**

● Status Code: 200
● JSON
{
"success": true,
"message": "Week details retrieved successfully",
"week": {
"id": 1,
"week_number": 1,
"title": "...",
"lectures": [
{
"id": 1,
"title": "...",
"video_id": "..."
},
// Additional Lectures
],
"assignments": [
{
"id": 1,
"title": "...",
"assignment_type":
	● Status Code: 200
	● JSON
	{
	"success": true,
	"message": "Week details retrieved successfully",
	"week": {
	"id": 1,
	"week_number": 1,
	"title": "...",
	"lectures": [
	{
	"id": 1,
	"title": "...",
	"video_id": "..."
	},
	// Additional Lectures
	],
	"assignments": [
	{
	"id": 1,
	"title": "...",
	"assignment_type":

WEEK APIs

"..."
},
// Additional Assignments
]
}
}
	"..."
	},
	// Additional Assignments
	]
	}
	}

Result: **Success**	

- **Test Case: Retrieve details of a non-existent week by its ID**

```python
def test_get_week_details_invalid_id(client):
    #Test GET /weeks/<week_id> with an ID that does not exist.
    response = client.get('/weeks/9999')

    assert response.status_code == 404
    data = response.get_json()

    assert data["success"] is False
    assert data["message"] == "Week not found"
```

Endpoint	
GET /api/weeks/9999	
Expected Output	**Actual Output**

• Status Code: 404
• JSON
{
"success": false,
"message": "Week not found"
}
	• Status Code: 404
	• JSON
	{
	"success": false,
	"message": "Week not found"
	}

Result: **Success**	

ASSIGNMENT APIs

/api/assignments

METHOD: GET

- **Test Case: Retrieve details of all the existing assignments**

```
# Test when assignments exist
def test_get_assignments_with_data(client):
    """
    Test case for GET /assignments when assignments are present in the
database.
    Expects a 200 status code with a list containing assignment data.
    """
    response = client.get('/assignments')  # Make the GET request
    assert response.status_code == 200  # Check for success status code

    json_data = response.get_json()  # Parse the response as JSON
    assert json_data['success'] is True # Ensure the success flag is True
    assert json_data['message'] == 'Assignments retrieved successfully'
# Validate the response message
    assert len(json_data['assignments']) > 0  # Confirm at least one
assignment is returned
```

Expected Output	Actual Output

• Status Code: 200
• JSON
{
"success": true,
"message": "Assignments retrieved successfully",
"assignments": [
{
"id": 1,
"title": "...",
"due_date": "...",
"description": "...",
}
// Additional assignments...
]
}
	• Status Code: 200
	• JSON
	{
	"success": true,
	"message": "Assignments retrieved successfully",
	"assignments": [
	{
	"id": 1,
	"title": "...",
	"due_date": "...",
	"description": "...",
	}
	// Additional assignments...
	]
	}

Result: **Success**	

ASSIGNMENT APIs

/api/assignments/{assignment_id}

METHOD: GET

- **Test Case: Retrieve details of a specific assignment by its ID**

```
# Test when assignments exist
def test_get_assignments_with_data(client):
    """
    Test case for GET /assignments when assignments are present in the
database.
    Expects a 200 status code with a list containing assignment data.
    """
    response = client.get('/assignments')  # Make the GET request
    assert response.status_code == 200  # Check for success status code

    json_data = response.get_json()  # Parse the response as JSON
    assert json_data['success'] is True # Ensure the success flag is True
    assert json_data['message'] == 'Assignments retrieved successfully'
# Validate the response message
    assert len(json_data['assignments']) > 0  # Confirm at least one
assignment is returned
```

Endpoint	
GET /api/assignments/1	
Expected Output	**Actual Output**

• Status Code: 200
• JSON
{
"success": true,
"message": "Assignment retrieved successfully",
"assignment": {
"id": 1,
"title": "...",
"due_date":"...",
"description":"..."
}
}
	• Status Code: 200
	• JSON
	{
	"success": true,
	"message": "Assignment retrieved successfully",
	"assignment": {
	"id": 1,
	"title": "...",
	"due_date":"...",
	"description":"..."
	}
	}

Result: Success	

ASSIGNMENT APIs

- **Test Case: Retrieve details of a non-existent assignment by its ID**

```
# Test when the assignment does not exist
def test_get_assignment_not_found(client):
    """
    Test case for GET /assignments/<assignment_id> when the assignment ID
does not exist.
    Expects a 404 status code with an 'Assignment not found' message.
    """
    response = client.get('/assignments/999')  # Using a non-existent
assignment ID
    assert response.status_code == 404  # Check for not found status code

    json_data = response.get_json()  # Parse the response as JSON
    assert json_data['success'] is False  # Ensure the success flag is
False
    assert json_data['message'] == 'Assignment not found'  # Validate the
not found message



    assert len(json_data['assignments']) > 0  # Confirm at least one
assignment is returned
```

Endpoint	
GET /api/assignments/999	
Expected Output	**Actual Output**

• Status Code: 404
• JSON
{
"success": false,
"message": "Assignment not found"
}
	• Status Code: 404
	• JSON
	{
	"success": false,
	"message": "Assignment not found"
	}

Result: Success	

PROGRAMMING ASSIGNMENT APIs

/api/programming_assignments/{question.id}

METHOD: GET

- **Test Case: Retrieve details of a specific programming assignment by its ID**

```python
# Test case for retrieving a programming assignment
def test_get_programming_assignment(client):

    question = ProgrammingAssignment.query.filter_by(id = 3).first()
    response = client.get(f'/programming_assignments/{question.id}')
    assert response.status_code == 200
    assert response.json['success'] == True
    assert response.json['data']['assignment_id'] == 101
```

Endpoint	
GET /api/programming_assignments/3	
Expected Output	**Actual Output**

• Status Code: 200
• JSON
{
"success": True,
"data": {
"assignment_id": 101,
"problem_statement": "...",
"input_format": "...",
"output_format": "...",
"constraints": "...",
"sample_input": "...",
"sample_output": "...",
"test_cases": [...]
}
}
	• Status Code: 200
	• JSON
	{
	"success": True,
	"data": {
	"assignment_id": 101,
	"problem_statement": "...",
	"input_format": "...",
	"output_format": "...",
	"constraints": "...",
	"sample_input": "...",
	"sample_output": "...",
	"test_cases": [...]
	}
	}

Result: Success	

PROGRAMMING ASSIGNMENT APIs

- **Test Case: Retrieve details of a non-existent programming assignment by its ID**

```python
# Test case for non-existent assignment retrieval
def test_get_non_existent_assignment(client):
    response = client.get('/programming_assignments/999')
    assert response.status_code == 404
    assert response.json['success'] == False
    assert "Programming assignment not found" in response.json['message']
```

Endpoint	
GET /api/programming_assignments/999	
Expected Output	**Actual Output**

• Status Code: 404
• JSON
{
"success": false,
"message": "Programming assignment not found"
}
	• Status Code: 404
	• JSON
	{
	"success": false,
	"message": "Programming assignment not found"
	}

Result: **Success**	

PROGRAMMING ASSIGNMENT APIs

/api/programming_assignments/{assignment_id}/execute

METHOD: POST

- **Test Case: Execute a programming assignment successfully**

```
# Test case for successful code execution with all test cases passing
def test_execute_solution_success(client):
    response = client.post('/programming_assignments/1/execute',
                           json={
                               "code": '''
                               def is_prime(N):
                                   if N <= 1:
                                   return 'NO'
                                   for i in range(2, int(N**0.5) + 1):
                                       if N % i == 0:
                                           return 'NO'
                                   return 'YES'
                                N = int(input())
                                print(is_prime(N))
                           '''})

    assert response.status_code == 200
    assert response.json['success'] is True
    assert response.json['score'] == 100
    assert response.json['passed_count'] == 4
    assert response.json['total_cases'] == 4
```

Input Data

- JSON

```
{
    "code": '''
            def is_prime(N):
              if N <= 1:
                 return 'NO'
              for i in range(2, int(N**0.5) + 1):
                 if N % i == 0:
                   return 'NO'
                 return 'YES'
            N = int(input())
            print(is_prime(N))'''
}
```

PROGRAMMING ASSIGNMENT APIs

Expected Output	Actual Output

• Status Code: 200
• JSON
{
"success": true,
"score": 100,
"passed_count": 4,
"total_cases": 4
}
	• Status Code: 200
	• JSON
	{
	"success": true,
	"score": 100,
	"passed_count": 4,
	"total_cases": 4
	}

Result: **Success**	