Software Engineering Project

Milestone 05

20 March 2025

Team 29

Ajay Thiagarajan (21f1003242)

A J R Vasu (21f3002975)

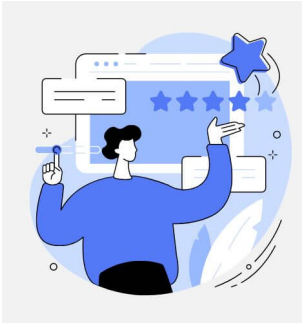
Anand K Iyer (21f1001185)

Anuj Gupta (21f3001598)

Ghanashyam R (21f1003387)

Jalaj Trivedi (21f2000730)

Niraj Kumar (21f1006589)



API Testing

1. Test Suite

This test suite is designed to verify the core functionalities of the system, including user authentication, course registration, user and admin statistics, and AI chatbot interactions. Each test module covers specific aspects of the application, ensuring reliability and consistency across various features.

Test Modules and Cases

Database Status

- **db_status**: Get DB connection status.

Login Tests (test_login.py)

- **1_student_login**: Verify login for student users.
- **2_admin_login**: Verify login for admin users.
- **3_faculty_login**: Verify login for faculty users.
- **4_email_required**: Ensure email is required for login.

Course Listing (test_courses.py)

- **list_courses**: Retrieve a list of all available courses.

Course Details (test_course.py)

- **course_details**: Fetch detailed information about a specific course using its ID.

Course Registration (test_register_courses.py)

- **1_register_course**: Allow a user to register for a course.
- **2_list_registered_courses**: Retrieve a list of registered courses for a user.
- **3_identify_user**: Identify a user during the registration process.
- **4_user_delete**: Delete a user record.
- **5_create_user**: Create a new user.
- **6_register_two_courses**: Register two courses for a user.
- **7_list_registered_courses**: Retrieve registered courses after multiple registrations.
- **8_register_invalid_course**: Attempt to register an invalid course.
- **9_list_registered_courses_without_email**: Try to list registered courses without providing an email.
- **10_list_registered_courses_invalid_user**: Attempt to list registered courses for an invalid user.
- **11_register_courses_empty_payload**: Handle registration with an empty request payload.

User Listing (test_users.py)

- **list_users**: Retrieve a list of all registered users in the system.

User Information (test_user.py)

- **1_login**: Authenticate a user.
- **2_user_details**: Fetch details of a specific user by their ID.
- **3_user_delete**: Delete a user record.
- **4_user_delete_user_not_found**: Handle cases where a non-existent user is deleted.
- **5_user_details_user_not_found**: Handle cases where details are requested for a non-existent user.

User Statistics (test_user_statistics.py)

- **1_login**: Authenticate a user before retrieving statistics.
- **2_user_statistics**: Retrieve statistics for a specific user.
- **3_user_delete**: Delete a user account.
- **4_user_statistics_user_not_found**: Handle cases where user statistics are requested for a non-existent user.

Video Transcript Retrieval (test_video_transcript.py)

- **1_transcript_video**: Retrieve a transcript for a lecture video using its URL.
- **2_transcript_invalid_video**: Handle cases where an invalid video URL is provided.
- **3_no_parameters**: Handle cases where no parameters are passed.

AI Chatbot Interaction (test_chatbot.py)

- **1_chatbot**: Verify chatbot query and response functionality.
- **2_incomplete_payload**: Handle cases where an incomplete payload is sent.
- **3_invalid_user**: Handle cases where an invalid user attempts to interact with the chatbot.

Admin Statistics (test_admin_statistics.py)

- **admin_statistics**: Retrieve administrative statistics, including user and system data management.

2. APIs and Individual Tests

Database Status API Test Documentation

Test 1: Check Database Status

API URL: `https://api-deepseek.vercel.app/db_status`
API Method: `GET`

Expected Status Code:

`200`

Actual Status Code:

`200`

Expected Keys and Data Types in Response:

- `status : bool`

Actual Keys and Data Types in Response:

- `status : bool`

Python Code:

```
def test_db_status():
    response = requests.get(API_URL)
    print(response.headers["Content-Type"])

    assertEquals(response.status_code, 200)

    assertEquals(response.headers["Content-Type"], "application/json")

    data = response.json()

    assertInstance(data, dict)

    required_keys = {"status":bool}
    verify_keys(required_keys, data)
```

Login API Test Documentation

Login

Test 1: Student Login

API URL: `https://api-deepseek.vercel.app/login`
API Method: POST

Input Data:

```
{
    "email": "student_mail",
    "name": "student_name",
    "picture": "profile_picture"
}
```

Expected Status Code:

200

Actual Status Code:

200

Expected Keys and Data Types in Response:

- userId : str
- name : str
- email : str
- role : str
- message : str
- picture : str

Actual Keys and Data Types in Response:

- userId : str
- name : str
- email : str
- role : str
- message : str
- picture : str

Python Code:

```
def test_1_student_login(student_mail,
                         student_name,
                         profile_picture,
                         login_success_msg,
```

```
                        helpers):
    input_data = {
        "email": student_mail,
        "name": student_name,
        "picture": profile_picture
    }
    payload = json.dumps(input_data)

    response = requests.post(API_LOGIN, data=payload, headers=headers)

    assertEquals(response.status_code, 200)

    assertEquals(response.headers["Content-Type"], "application/json")

    data = response.json()

    assertInstance(data, dict)

    required_keys = {"userId":str,
                     "name":str,
                     "email":str,
                     "role":str,
                     "message":str,
                     "picture":str}
    verify_keys(required_keys, data)

    assertEquals(data['email'], student_mail)

    assertEquals(data['message'], login_success_msg)

    assertEquals(data['role'], "student")
```

Test 2: Admin Login

API URL: https://api-deepseek.vercel.app/login
API Method: POST

Input Data:

```
{
    "email": "admin_mail",
    "name": "admin_name",
    "picture": "profile_picture"
}
```

Expected Status Code:

200

Actual Status Code:

200

Expected Keys and Data Types in Response:

- userId : str
- name : str
- email : str
- role : str
- message : str
- picture : str

Actual Keys and Data Types in Response:

- userId : str
- name : str
- email : str
- role : str
- message : str
- picture : str

Python Code:

```
def test_2_admin_login(admin_mail,
                       admin_name,
                       profile_picture,
                       login_success_msg):
    input_data = {
    "email": admin_mail,
    "name": admin_name,
    "picture": profile_picture
    }
    payload = json.dumps(input_data)

    response = requests.post(API_LOGIN, data=payload, headers=headers)

    assertEquals(response.status_code, 200)

    assertEquals(response.headers["Content-Type"], "application/json")

    data = response.json()

    assertInstance(data, dict)

    required_keys = {"userId":str,
                     "name":str,
                     "email":str,
                     "role":str,
                     "message":str,
                     "picture":str}
```

```
    verify_keys(required_keys, data)

    assertEquals(data['email'], admin_mail)

    assertEquals(data['message'], login_success_msg)

    assertEquals(data['role'], "admin")
```

Test 3: Faculty Login

API URL: https://api-deepseek.vercel.app/login
API Method: POST

Input Data:

```
{
    "email": "faculty_mail",
    "name": "faculty_name",
    "picture": "profile_picture"
}
```

Expected Status Code:

200

Actual Status Code:

200

Expected Keys and Data Types in Response:

- userId : str
- name : str
- email : str
- role : str
- message : str
- picture : str

Actual Keys and Data Types in Response:

- userId : str
- name : str
- email : str
- role : str
- message : str
- picture : str

Python Code:

```
def test_3_faculty_login(faculty_mail,
```

```
                        faculty_name,
                        profile_picture,
                        login_success_msg):
    input_data = {
    "email": faculty_mail,
    "name": faculty_name,
    "picture": profile_picture
    }
    payload = json.dumps(input_data)

    response = requests.post(API_LOGIN, data=payload, headers=headers)

    assertEquals(response.status_code, 200)

    assertEquals(response.headers["Content-Type"], "application/json")

    data = response.json()

    assertInstance(data, dict)

    required_keys = {"userId":str,
                     "name":str,
                     "email":str,
                     "role":str,
                     "message":str,
                     "picture":str}
    verify_keys(required_keys, data)

    assertEquals(data['email'], faculty_mail)

    assertEquals(data['message'], login_success_msg)

    assertEquals(data['role'], "faculty")
```

Test 4: Email Required

API URL: https://api-deepseek.vercel.app/login
API Method: POST

Input Data:

```
{
    "email": "faculty_mail",
    "name": "faculty_name",
    "picture": "profile_picture"
}
```

Expected Status Code:

400

Actual Status Code:

400

Expected Error Message:

Email is required

Actual Error Message:

Email is required

Expected Keys and Data Types in Response:

- error : str

Actual Keys and Data Types in Response:

- error : str

Python Code:

```
def test_4_email_required(student_name,
                          profile_picture,
                          email_required_msg):
    input_data = {
    "email": "",
    "name": student_name,
    "picture": profile_picture
    }
    payload = json.dumps(input_data)

    response = requests.post(API_LOGIN, data=payload, headers=headers)

    assertEquals(response.status_code, 400)

    data = response.json()

    required_keys = {"error":str}
    verify_keys(required_keys, data)

    assertEquals(data['error'], email_required_msg)
```

Courses API Test Documentation

Test 1: List All Courses

API URL: `https://api-deepseek.vercel.app/courses`
API Method: `GET`

Expected Status Code:

`200`

Actual Status Code:

`200`

Expected Keys and Data Types in Response:

- `courses` *:* `list`

Actual Keys and Data Types in Response:

- `courses` *:* `list`

Additional Data Verification

Course Structure:

Each item in the `courses` *list must contain:*

- `id` *:* `str`
- `name` *:* `str`
- `description` *:* `str`
- `startDate` *:* `str`
- `endDate` *:* `str`

Python Code:

```
def test_list_courses():
    response = requests.get(API_URL)
    print(response.headers["Content-Type"])

    assertEquals(response.status_code, 200)

    assertEquals(response.headers["Content-Type"], "application/json")

    data = response.json()

    assertInstance(data, dict)
```

```
    courses = data['courses']

    assertTrue(len(courses) > 0, "Expected response to contain at least one
course")

    assertInstance(courses, list)

    for course in courses:
        assertInstance(course, dict)

        required_keys = {"description":str,
                         "endDate":str,
                         "id":str,
                         "name":str,
                         "startDate":str
                         }
        verify_keys(required_keys, course)
```

Course Details API Test Documentation

Test 1: Fetch Course Details

API URL: `https://api-deepseek.vercel.app/course/{course_id}`
API Method: `GET`

Expected Status Code:

`200`

Actual Status Code:

`200`

Expected Keys and Data Types in Response:

- `announcements` *:* `list`
- `courseId` *:* `str`
- `name` *:* `str`
- `description` *:* `str`
- `startDate` *:* `str`
- `endDate` *:* `str`
- `weeks` *:* `list`

Actual Keys and Data Types in Response:

- `announcements` *:* `list`
- `courseId` *:* `str`
- `name` *:* `str`
- `description` *:* `str`
- `startDate` *:* `str`
- `endDate` *:* `str`
- `weeks` *:* `list`

Additional Data Verification

Announcements Structure:

Each item in the `announcements` *list must contain:*

- `announcementId` *:* `str`
- `date` *:* `str`
- `message` *:* `str`

Weeks Structure:

Each item in the `weeks` *list must contain:*

- `deadline` *:* `str`
- `modules` *:* `list`
- `title` *:* `str`
- `weekId` *:* `str`

Modules Structure:

Each item in the `modules` *list must contain:*

- `moduleId` *:* `str`
- `title` *:* `str`
- `type` *:* `str`

Optional field (only present in some modules):

- `questions` *:* `list`

Questions Structure:

If `questions` *key is present in a module, each item in the* `questions` *list must contain:*

- `correctAnswer` *:* `str`
- `options` *:* `list`
- `question` *:* `str`
- `type` *:* `str`

Python Code:

```
def test_course_details(course1_id):
    response = requests.get(API_COURSE.format(course_id=course1_id))

    assertEquals(response.status_code, 200)

    assertEquals(response.headers["Content-Type"], "application/json")

    data = response.json()

    assertInstance(data, dict)

    required_keys = {"announcements":list,
                     "courseId":str,
                     "name":str,
                     "description":str,
                     "startDate":str,
                     "endDate":str,
                     "weeks":list
                     }
    verify_keys(required_keys, data)

    announcements = data['announcements']
```

```
    for announcement in announcements:
        assertInstance(announcement, dict)

        required_keys = {"announcementId":str,
                         "date":str,
                         "message":str
                         }
        verify_keys(required_keys, announcement)

    weeks = data['weeks']
    for week in weeks:
        assertInstance(week, dict)

        required_keys = {"deadline":str,
                         "modules":list,
                         "title":str,
                         "weekId":str
                         }
        verify_keys(required_keys, week)

        modules = week['modules']
        for module in modules:
            assertInstance(module, dict)

            required_keys = {"moduleId":str,
                             "title":str,
                             "type":str
                             } #url and questions not present in all modules
            verify_keys(required_keys, module)

            if "questions" in module:
                questions = module['questions']
                assertInstance(questions, list)

                for question in questions:
                    assertInstance(question, dict)

                    required_keys = {"correctAnswer":str,
                                     #  "hint":str, # Hint can be null
sometimes.
                                      "options":list,
                                      "question":str,
                                      "type":str
                                      }
                    verify_keys(required_keys, question)
```

Register Courses API Test Documentation

Test 1: Register Course

API URL: `https://api-deepseek.vercel.app/registered-courses`
API Method: POST

Input Data:

```
{
    "email": "student_mail",
    "courses": ["course1_id"]
}
```

Expected Status Code:

200

Actual Status Code:

200

Expected Message:

User updated with new courses

Actual Message:

User updated with new courses

Expected Keys and Data Types in Response:

- message : str
- user_id : str

Actual Keys and Data Types in Response:

- message : str
- user_id : str

Python Code:

```
def test_1_register_course(student_mail,
                           student_id,
                           course1_id,
                           course_reg_success_msg):
    input_data = {
        "email": student_mail,
        "courses": [
            course1_id
        ]
```

```
    }

    headers = {
    'Content-Type': 'application/json'
    }

    payload = json.dumps(input_data)

    response = requests.post(API_REGISTER_COURSE, data=payload,
headers=headers)

    assertEquals(response.status_code, 200)

    assertEquals(response.headers["Content-Type"], "application/json")

    data = response.json()

    assertInstance(data, dict)

    required_keys = {"message":str,
                     "user_id":str
                     }
    verify_keys(required_keys, data)

    assertEquals(data['message'], course_reg_success_msg)

    assertEquals(data['user_id'], student_id)
```

Test 2: List Registered Courses

API URL: `https://api-deepseek.vercel.app/registered-courses?email={email}`
API Method: `GET`

Expected Status Code:

`200`

Actual Status Code:

`200`

Expected Keys and Data Types in Response:

- `registeredCourses : list`

Actual Keys and Data Types in Response:

- `registeredCourses : list`

Python Code:

```
def test_2_list_registered_courses(student_mail,
                                   course1_id):
    response =
requests.get(API_REGISTERED_COURSES.format(email=student_mail))

    assertEquals(response.status_code, 200)

    assertEquals(response.headers["Content-Type"], "application/json")

    data = response.json()

    assertInstance(data, dict)

    required_keys = {"registeredCourses":list}
    verify_keys(required_keys, data)

    courses = data['registeredCourses']
    assertInstance(courses, list)

    set_courses = set()
    for course in courses:
        assertInstance(course, dict)

        required_keys = {"id":str,
                         "name":str,
                         "description":str
                         }
        verify_keys(required_keys, course)

        set_courses.add(course['id'])
    # Cannot be assertEquals, due to non-standard message.
    assertTrue(set_courses == {course1_id}, f"Expected response to have
following courses: {course1_id}, but found the following keys:
{list(set_courses)}")
```

Test 3: Identify User

API URL: https://api-deepseek.vercel.app/login
API Method: POST

Input Data:

```
{
    "email": "student2_mail",
    "name": "student2_name",
```

```
    "picture": "profile_picture"
}
```

Expected Status Code:

200

Actual Status Code:

200

Expected Keys and Data Types in Response:

- userId : str

Actual Keys and Data Types in Response:

- userId : str

Python Code:

```
def test_3_identify_user(student2_mail,
                 student2_name,
                 profile_picture):
    global user_id
    input_data = {
        "email": student2_mail,
        "name": student2_name,
        "picture": profile_picture
    }
    payload = json.dumps(input_data)

    response = requests.post(API_LOGIN, data=payload, headers=headers)

    assertEquals(response.status_code, 200)

    data = response.json()

    user_id = data['userId']
```

Test 4: Delete User

API URL: https://api-deepseek.vercel.app/user/{user_id}
API Method: DELETE

Expected Status Code:

200

Actual Status Code:

200

Expected Message:

User deleted successfully

Actual Message:

User deleted successfully

Expected Keys and Data Types in Response:

- message : str

Actual Keys and Data Types in Response:

- message : str

Python Code:

```
def test_4_user_delete(student2_mail,
                       user_del_success_msg):
    global user_id
    response = requests.delete(API_USER.format(user_id=user_id))

    assertEquals(response.status_code, 200)

    data = response.json()

    required_keys = {"message":str}
    verify_keys(required_keys, data)

    assertEquals(data['message'], user_del_success_msg)
```

Test 5: Create User

API URL: https://api-deepseek.vercel.app/login
API Method: POST

Expected Status Code:

200

Actual Status Code:

200

Expected Keys and Data Types in Response:

- userId : str

Actual Keys and Data Types in Response:

- `userId: str`

Python Code:

```
def test_5_create_user(student2_mail,
                student2_name,
                profile_picture):
    global user_id
    input_data = {
        "email": student2_mail,
        "name": student2_name,
        "picture": profile_picture
    }
    payload = json.dumps(input_data)

    response = requests.post(API_LOGIN, data=payload, headers=headers)

    assertEquals(response.status_code, 200)

    data = response.json()

    user_id = data['userId']
```

Test 6: Register Two Courses

API URL: `https://api-deepseek.vercel.app/registered-courses`
API Method: `POST`

Input Data:

```
{
    "email": "student2_mail",
    "courses": ["course1_id", "course2_id"]
}
```

Expected Status Code:

`200`

Actual Status Code:

`200`

Expected Message:

`User updated with new courses`

Actual Message:

```
User updated with new courses
```

Expected Keys and Data Types in Response:

- `message : str`
- `user_id : str`

Actual Keys and Data Types in Response:

- `message : str`
- `user_id : str`

Python Code:

```
def test_6_register_two_courses(student2_mail,
                           course1_id,
                           course2_id,
                           course_reg_success_msg):
    input_data = {
        "email": student2_mail,
        "courses": [
            course1_id, course2_id
        ]
    }

    headers = {
    'Content-Type': 'application/json'
    }

    payload = json.dumps(input_data)

    response = requests.post(API_REGISTER_COURSE, data=payload,
headers=headers)

    assertEquals(response.status_code, 200)

    assertEquals(response.headers["Content-Type"], "application/json")

    data = response.json()

    assertInstance(data, dict)

    required_keys = {"message":str,
                     "user_id":str}
    verify_keys(required_keys, data)

    assertEquals(data['message'], course_reg_success_msg)
```

```
    assertEquals(data['user_id'], user_id)
```

Test 7: List Registered Courses for Two Courses

API URL: `https://api-deepseek.vercel.app/registered-courses?email={email}`
API Method: `GET`

Expected Status Code:

`200`

Actual Status Code:

`200`

Expected Keys and Data Types in Response:

- `registeredCourses : list`

Actual Keys and Data Types in Response:

- `registeredCourses : list`

Python Code:

```
def test_7_list_registered_courses(student2_mail,
                                   course1_id,
                                   course2_id):
    response =
requests.get(API_REGISTERED_COURSES.format(email=student2_mail))

    assertEquals(response.status_code, 200)

    assertEquals(response.headers["Content-Type"], "application/json")

    data = response.json()

    assertInstance(data, dict)

    required_keys = {"registeredCourses":list}
    verify_keys(required_keys, data)

    courses = data['registeredCourses']

    set_courses = set()
    for course in courses:
        assertInstance(course, dict)

        required_keys = {"id":str,
```

```
                        "name":str,
                        "description":str}
        verify_keys(required_keys, course)

        set_courses.add(course['id'])

    # Cannot be assertEquals, due to non-standard message.
    assertTrue(set_courses == {course1_id, course2_id}, f"Expected response
to have following courses: {course1_id, course2_id}, but found the following
keys: {list(set_courses)}")
```

Test 8: Register Invalid Course

API URL: https://api-deepseek.vercel.app/registered-courses
API Method: POST

Expected Status Code:

400

Actual Status Code:

400

Expected Error Message:

Invalid course ID

Actual Error Message:

Invalid course ID

Expected Keys and Data Types in Response:

- details : str
- error : str

Actual Keys and Data Types in Response:

- details : str
- error : str

Python Code:

```
def test_8_register_invalid_course(student_mail,
                        student_id,
                        invalid_course_id,
                        invalid_course_msg):
    input_data = {
        "email": student_mail,
```

```
        "courses": [
            invalid_course_id
        ]
    }

    headers = {
    'Content-Type': 'application/json'
    }

    payload = json.dumps(input_data)

    response = requests.post(API_REGISTER_COURSE, data=payload,
headers=headers)

    assertEquals(response.status_code, 400)

    assertEquals(response.headers["Content-Type"], "application/json")

    data = response.json()

    assertInstance(data, dict)

    required_keys = {"details":str,
                     "error":str
                    }
    verify_keys(required_keys, data)

    assertEquals(data['error'], invalid_course_msg)
```

Test 9: List Registered Courses Without Email

API URL: `https://api-deepseek.vercel.app/registered-courses?email=`
API Method: `GET`

Expected Status Code:

```
400
```

Actual Status Code:

```
400
```

Expected Error Message:

```
Email is required
```

Actual Error Message:

```
Email is required
```

Expected Keys and Data Types in Response:

- `error : str`

Actual Keys and Data Types in Response:

- `error : str`

Python Code:

```
def test_9_list_registered_courses_without_email(
                          email_required_msg):
    response = requests.get(API_REGISTERED_COURSES.format(email=''))

    assertEquals(response.status_code, 400)

    assertEquals(response.headers["Content-Type"], "application/json")

    data = response.json()

    assertInstance(data, dict)

    required_keys = {"error":str}
    verify_keys(required_keys, data)

    assertEquals(data['error'], email_required_msg)
```

Test 10: List Registered Courses for Invalid User

API URL: `https://api-deepseek.vercel.app/registered-courses?email={invalid_student_mail}`
API Method: `GET`

Expected Status Code:

`404`

Actual Status Code:

`404`

Expected Error Message:

`User not found`

Actual Error Message:

`User not found`

Expected Keys and Data Types in Response:

- `error : str`

Actual Keys and Data Types in Response:

- `error : str`

Python Code:

```
def test_10_list_registered_courses_invalid_user(invalid_student_mail,
                          user_not_found_msg):
    response =
requests.get(API_REGISTERED_COURSES.format(email=invalid_student_mail))

    assertEquals(response.status_code, 404)

    assertEquals(response.headers["Content-Type"], "application/json")

    data = response.json()

    assertInstance(data, dict)

    required_keys = {"error":str}
    verify_keys(required_keys, data)

    assertEquals(data['error'], user_not_found_msg)
```

Test 11: Register Courses with Empty Payload

API URL: `https://api-deepseek.vercel.app/registered-courses`
API Method: `POST`

Expected Status Code:

`400`

Actual Status Code:

`400`

Expected Error Message:

`Email and courses are required`

Actual Error Message:

`Email and courses are required`

Expected Keys and Data Types in Response:

- error : str

Actual Keys and Data Types in Response:

- error : str

Python Code:

```
def test_11_register_courses_empty_payload(reg_bad_request_msg):
    input_data = {
    }

    headers = {
    'Content-Type': 'application/json'
    }

    payload = json.dumps(input_data)

    response = requests.post(API_REGISTER_COURSE, data=payload,
headers=headers)

    assertEquals(response.status_code, 400)

    assertEquals(response.headers["Content-Type"], "application/json")

    data = response.json()

    assertInstance(data, dict)

    required_keys = {"error":str}
    verify_keys(required_keys, data)

    assertEquals(data['error'], reg_bad_request_msg)
```

Users API Test Documentation

Test 1: List All Users

API URL: `https://api-deepseek.vercel.app/users`
API Method: `GET`

Expected Status Code:

`200`

Actual Status Code:

`200`

Expected Keys and Data Types in Response:

- `id : str`
- `name : str`
- `email : str`
- `role : str`
- `profilePictureUrl : str`
- `registeredCourses : list`

Actual Keys and Data Types in Response:

- `id : str`
- `name : str`
- `email : str`
- `role : str`
- `profilePictureUrl : str`
- `registeredCourses : list`

Python Code:

```
def test_list_users():
    response = requests.get(API_URL)
    print(response.headers["Content-Type"])

    assertEquals(response.status_code, 200)

    assertEquals(response.headers["Content-Type"], "application/json")

    users = response.json()

    assertInstance(users, list)
```

```
    for user in users:
        assertInstance(user, dict)

        required_keys = {"id":str,
                         "name":str,
                         "profilePictureUrl":str,
                         "registeredCourses":list,
                         "role":str,
                         "email":str
                         }
        verify_keys(required_keys, user)
```

User Details API Test Documentation

Test 1: User Login

API URL: https://api-deepseek.vercel.app/login
API Method: POST

Input Data:

```
{
    "email": "student2_mail",
    "name": "student2_name",
    "picture": "profile_picture"
}
```

Expected Status Code:

200

Actual Status Code:

200

Expected Keys and Data Types in Response:

- userId : str

Actual Keys and Data Types in Response:

- userId : str

Python Code:

```
def test_1_login(student2_mail,
                 student2_name,
                 profile_picture):
    global user_id
    input_data = {
        "email": student2_mail,
        "name": student2_name,
        "picture": profile_picture
    }
    payload = json.dumps(input_data)

    response = requests.post(API_LOGIN, data=payload, headers=headers)

    assertEquals(response.status_code, 200)

    data = response.json()

    user_id = data['userId']
```

Test 2: Fetch User Details

API URL: `https://api-deepseek.vercel.app/user/{user_id}`
API Method: `GET`

Expected Status Code:

`200`

Actual Status Code:

`200`

Expected Keys and Data Types in Response:

- `id : str`
- `name : str`
- `email : str`
- `profilePictureUrl : str`
- `registeredCourses : list`
- `role : str`

Actual Keys and Data Types in Response:

- `id : str`
- `name : str`
- `email : str`
- `profilePictureUrl : str`
- `registeredCourses : list`
- `role : str`

Python Code:

```
def test_2_user_details():
    global user_id
    response = requests.get(API_USER.format(user_id=user_id))

    assertEquals(response.status_code, 200)

    assertEquals(response.headers["Content-Type"], "application/json")

    data = response.json()

    assertInstance(data, dict)

    required_keys = {"id":str,
                     "name":str,
```

```
                    "email":str,
                    "profilePictureUrl":str,
                    "registeredCourses":list,
                    "role":str
                    }
    verify_keys(required_keys, data)

    assertEquals(data['id'], user_id)

    assertEquals(data['role'], "student")
```

Test 3: Delete User

API URL: `https://api-deepseek.vercel.app/user/{user_id}`
API Method: `DELETE`

Expected Status Code:

```
200
```

Actual Status Code:

```
200
```

Expected Message:

```
User deleted successfully
```

Actual Message:

```
User deleted successfully
```

Expected Keys and Data Types in Response:

- `message : str`

Actual Keys and Data Types in Response:

- `message : str`

Python Code:

```
def test_3_user_delete(user_del_success_msg):
    global user_id
    response = requests.delete(API_USER.format(user_id=user_id))

    data = response.json()

    required_keys = {"message":str}
    verify_keys(required_keys, data)
```

```
    assertEquals(data['message'], user_del_success_msg)
```

Test 4: Delete Non-Existent User

API URL: https://api-deepseek.vercel.app/user/{user_id}
API Method: DELETE

Expected Status Code:

404

Actual Status Code:

404

Expected Error Message:

User not found

Actual Error Message:

User not found

Expected Keys and Data Types in Response:

- error : str

Actual Keys and Data Types in Response:

- error : str

Python Code:

```
def test_4_user_delete_user_not_found(user_not_found_msg):
    global user_id
    response = requests.delete(API_USER.format(user_id=user_id))

    data = response.json()

    required_keys = {"error":str}
    verify_keys(required_keys, data)

    assertEquals(data['error'], user_not_found_msg)
```

Test 5: Fetch Non-Existent User Details

API URL: https://api-deepseek.vercel.app/user/{user_id}
API Method: GET

Expected Status Code:

404

Actual Status Code:

404

Expected Error Message:

User not found

Actual Error Message:

User not found

Expected Keys and Data Types in Response:

- error : str

Actual Keys and Data Types in Response:

- error : str

Python Code:

```
def test_5_user_details_user_not_found(user_not_found_msg):
    global user_id
    response = requests.get(API_USER.format(user_id=user_id))

    assertEquals(response.status_code, 404)

    assertEquals(response.headers["Content-Type"], "application/json")

    data = response.json()

    assertInstance(data, dict)

    required_keys = {"error":str}
    verify_keys(required_keys, data)

    assertEquals(data['error'], user_not_found_msg)
```

User Statistics API Test Documentation

Test 1: User Login

API URL: https://api-deepseek.vercel.app/login
API Method: POST

Input Data:

```
{
    "email": "student2_mail",
    "name": "student2_name",
    "picture": "profile_picture"
}
```

Expected Status Code:

200

Actual Status Code:

200

Expected Keys and Data Types in Response:

- userId: str

Actual Keys and Data Types in Response:

- userId: str

Python Code:

```
def test_1_login(student2_name,
                 student2_mail,
                 profile_picture):
    global user_id
    input_data = {
        "email": student2_mail,
        "name": student2_name,
        "picture": profile_picture
    }
    payload = json.dumps(input_data)

    response = requests.post(API_LOGIN, data=payload, headers=headers)

    assertEquals(response.status_code, 200)

    data = response.json()

    user_id = data['userId']
```

Test 2: Fetch User Statistics

API URL: `https://api-deepseek.vercel.app/user-statistics/{user_id}`
API Method: `GET`

Expected Status Code:

`200`

Actual Status Code:

`200`

Expected Keys and Data Types in Response:

- `id` *:* `str`
- `name` *:* `str`
- `email` *:* `str`
- `statistics` *:* `dict`
- `registeredCourses` *:* `list`
- `role` *:* `str`

Actual Keys and Data Types in Response:

- `id` *:* `str`
- `name` *:* `str`
- `email` *:* `str`
- `statistics` *:* `dict`
- `registeredCourses` *:* `list`
- `role` *:* `str`

Additional Data Verification

Statistics Structure:

Each `statistics` *object must contain:*

- `averageScore` *:* `int`
- `modulesCompleted` *:* `int`
- `questionsAttempted` *:* `int`

Python Code:

```
def test_2_user_statistics():
    global user_id
    response = requests.get(API_USER_STATS.format(user_id=user_id))
```

```
    assertEquals(response.status_code, 200)

    assertEquals(response.headers["Content-Type"], "application/json")

    data = response.json()

    assertInstance(data, dict)

    required_keys = {"id":str,
                     "name":str,
                     "email":str,
                     "statistics":dict,
                     "registeredCourses":list,
                     "role":str
                     }
    verify_keys(required_keys, data)

    statistics = data['statistics']

    required_keys = {"averageScore":int,
                     "modulesCompleted":int,
                     "questionsAttempted":int
                     }
    verify_keys(required_keys, statistics)

    assertEquals(data['id'], user_id)

    assertEquals(data['role'], "student")
```

Test 3: Delete User

API URL: `https://api-deepseek.vercel.app/user/{user_id}`
API Method: `DELETE`

Expected Status Code:

```
200
```

Actual Status Code:

```
200
```

Expected Message:

```
User deleted successfully
```

Actual Message:

```
User deleted successfully
```

Expected Keys and Data Types in Response:

- `message : str`

Actual Keys and Data Types in Response:

- `message : str`

Python Code:

```
def test_3_user_delete(user_del_success_msg):
    global user_id
    response = requests.delete(API_USER.format(user_id=user_id))

    data = response.json()

    required_keys = ["message"]
    assertTrue(set(required_keys) == set(data.keys()), f"Expected response to
have following keys: {required_keys}, but found the following keys:
{list(data.keys())}")

    assertEquals(data['message'], user_del_success_msg)
```

Test 4: Fetch Statistics for Non-Existent User

API URL: `https://api-deepseek.vercel.app/user-statistics/{user_id}`
API Method: `GET`

Expected Status Code:

`404`

Actual Status Code:

`404`

Expected Error Message:

`Internal Server Error`

Actual Error Message:

`Internal Server Error`

Expected Keys and Data Types in Response:

- `message : str`

Actual Keys and Data Types in Response:

- `message : str`

Python Code:

```
def test_4_user_statistics_user_not_found(server_error_msg):
    global user_id
    response = requests.get(API_USER_STATS.format(user_id=user_id))

    assertEquals(response.status_code, 404) # Bug

    assertEquals(response.headers["Content-Type"], "application/json")

    data = response.json()

    assertInstance(data, dict)

    required_keys = {"message":str}
    verify_keys(required_keys, data)

    assertEquals(data['message'], server_error_msg)
```

Video Transcript API Test Documentation

Test 1: Fetch Video Transcript

API URL: `https://api-deepseek.vercel.app/video-transcript?videoURL={video_url}`
API Method: `GET`

Expected Status Code:

`200`

Actual Status Code:

`200`

Expected Keys and Data Types in Response:

- `videoID : str`
- `videoURL : str`
- `transcript : str`

Actual Keys and Data Types in Response:

- `videoID : str`
- `videoURL : str`
- `transcript : str`

Python Code:

```
def test_1_transcript_video(video_url):
    response = requests.get(API_URL.format(video_url=video_url))

    assertEquals(response.status_code, 200)

    assertEquals(response.headers["Content-Type"], "application/json")

    data = response.json()

    assertInstance(data, dict)

    required_keys = {"videoID":str,
                     "videoURL":str,
                     "transcript":str,
                     }
    verify_keys(required_keys, data)
```

Test 2: Fetch Transcript for Invalid Video

API URL: `https://api-deepseek.vercel.app/video-transcript?videoURL={invalid_video_url}`
API Method: `GET`

Expected Status Code:

`404`

Actual Status Code:

`404`

Expected Error Message:

`Transcript not found for the given video URL`

Actual Error Message:

`Transcript not found for the given video URL`

Expected Keys and Data Types in Response:

- `error : str`

Actual Keys and Data Types in Response:

- `error : str`

Python Code:

```
def test_2_transcript_invalid_video(invalid_video_url,
                                    transcript_not_found_msg):
    response = requests.get(API_URL.format(video_url=invalid_video_url))

    assertEquals(response.status_code, 404)

    assertEquals(response.headers["Content-Type"], "application/json")

    data = response.json()

    assertInstance(data, dict)

    required_keys = {"error":str}
    verify_keys(required_keys, data)

    assertEquals(data['error'], transcript_not_found_msg)
```

Test 3: Fetch Transcript Without Video URL

API URL: `https://api-deepseek.vercel.app/video-transcript?videoURL=`
API Method: `GET`

Expected Status Code:

400

Actual Status Code:

400

Expected Error Message:

videoURL is required

Actual Error Message:

videoURL is required

Expected Keys and Data Types in Response:

- `error : str`

Actual Keys and Data Types in Response:

- `error : str`

Python Code:

```
def test_3_no_parameters(video_url_required_msg):
    response = requests.get(API_URL.format(video_url=''))

    assertEquals(response.status_code, 400)

    assertEquals(response.headers["Content-Type"], "application/json")

    data = response.json()

    assertInstance(data, dict)

    required_keys = {"error":str}
    verify_keys(required_keys, data)

    assertEquals(data['error'], video_url_required_msg)
```

AI Chatbot API Test Documentation

Test 1: Chatbot Interaction

API URL: `https://api-deepseek.vercel.app/chatbot`
API Method: `POST`

Input Data:

```
{
    "query": "What is ML?",
    "option": "option",
    "sessionId": "S1234",
    "userEmail": "student_mail"
}
```

Expected Status Code:

`200`

Actual Status Code:

`200`

Expected Keys and Data Types in Response:

- `sessionId` *:* `str`
- `question` *:* `str`
- `answer` *:* `str`
- `chatHistory` *:* `list`

Actual Keys and Data Types in Response:

- `sessionId` *:* `str`
- `question` *:* `str`
- `answer` *:* `str`
- `chatHistory` *:* `list`

Additional Data Verification

Chat History Structure:

Each item in the `chatHistory` *list must contain:*

- `query` *:* `str`
- `answer` *:* `str`
- `timestamp` *:* `str`
- `user` *:* `dict`

User Structure:

Each `user` *object in the* `chatHistory` *must contain:*

- `id` *:* `str`
- `name` *:* `str`
- `email` *:* `str`
- `role` *:* `str`
- `profilePictureUrl` *:* `str`

Python Code:

```
def test_1_chatbot(student_mail,
                   student_id,
                   student_name,
                   student_pp):
    session_id = "S1234"
    query = "What is ML?"
    input_data = {
        "query": query,
        "option": "option",
        "sessionId": session_id,
        "userEmail": student_mail
    }
    payload = json.dumps(input_data)

    response = requests.post(API_CHATBOT, data=payload, headers=headers)

    assertEquals(response.status_code, 200)

    assertEquals(response.headers["Content-Type"], "application/json")

    data = response.json()

    assertInstance(data, dict)

    required_keys = {"sessionId":str,
                         "question":str,
                         "answer":str,
                         "chatHistory": list
                     }
    verify_keys(required_keys, data)

    assertEquals(data['sessionId'], session_id)

    assertEquals(data['question'], query)

    chathistories = data["chatHistory"]
    for chathistory in chathistories:
        assertInstance(chathistory, dict)
```

```
        required_keys = {"query":str,
                             "answer":str,
                             "timestamp":str,
                             "user": dict
                         }
        verify_keys(required_keys, chathistory)

        user = chathistory['user']
        required_keys = {"id":str,
                             "name":str,
                             "email":str,
                             "role": str,
                             "profilePictureUrl": str
                         }
        verify_keys(required_keys, user)

        assertEquals(user['name'], student_name)

        assertEquals(user['email'], student_mail)

        assertEquals(user['role'], "student")

        assertEquals(user['profilePictureUrl'], student_pp)
```

Test 2: Incomplete Payload

API URL: https://api-deepseek.vercel.app/chatbot
API Method: POST

Input Data:

```
{
    "option": "option",
    "sessionId": "S1234",
    "userEmail": "student_mail"
}
```

Expected Status Code:

400

Actual Status Code:

400

Expected Error Message:

Query and option are required

Actual Error Message:

```
Query and option are required
```

Expected Keys and Data Types in Response:

- `error : str`

Actual Keys and Data Types in Response:

- `error : str`

Python Code:

```
def test_2_incomplete_payload(student_mail,
                                    query_option_required_msg):
    input_data = {
        "option": "option",
        "sessionId": "S1234",
        "userEmail": student_mail
    }
    payload = json.dumps(input_data)

    response = requests.post(API_CHATBOT, data=payload, headers=headers)

    assertEquals(response.status_code, 400)

    assertEquals(response.headers["Content-Type"], "application/json")

    data = response.json()

    assertInstance(data, dict)

    required_keys = {"error":str}
    verify_keys(required_keys, data)

    assertEquals(data['error'], query_option_required_msg)
```

Test 3: Invalid User

API URL: `https://api-deepseek.vercel.app/chatbot`
API Method: `POST`

Input Data:

```
{
    "query": "What is ML?",
    "option": "option",
    "sessionId": "S1234",
    "userEmail": "invalid_student_mail"
}
```

Expected Status Code:

404

Actual Status Code:

404

Expected Error Message:

User not found

Actual Error Message:

User not found

Expected Keys and Data Types in Response:

- error : str

Actual Keys and Data Types in Response:

- error : str

Python Code:

```
def test_3_invalid_user(invalid_student_mail,
                            user_not_found_msg):
    input_data = {
        "query": "What is ML?",
        "option": "option",
        "sessionId": "S1234",
        "userEmail": invalid_student_mail
    }
    payload = json.dumps(input_data)

    response = requests.post(API_CHATBOT, data=payload, headers=headers)

    assertEquals(response.status_code, 404)

    assertEquals(response.headers["Content-Type"], "application/json")

    data = response.json()

    assertInstance(data, dict)

    required_keys = {"error":str}
    verify_keys(required_keys, data)

    assertEquals(data['error'], user_not_found_msg)
```

Admin Statistics API Test Documentation

API URL: `https://api-deepseek.vercel.app/admin-statistics`
API Method: `GET`

Expected Status Code:

`200`

Actual Status Code:

`200`

Expected Keys and Data Types in Response:

- `totalUsers : int`
- `totalModules : int`
- `activeUsers : int`
- `questionsAttempted : int`

Actual Keys and Data Types in Response:

- `totalUsers : int`
- `totalModules : int`
- `activeUsers : int`
- `questionsAttempted : int`

Python Code:

```
def test_admin_statistics():
    response = requests.get(API_URL)
    print(response.headers["Content-Type"])

    assertEquals(response.status_code, 200)

    assertEquals(response.headers["Content-Type"], "application/json")

    data = response.json()

    assertInstance(data, dict)

    required_keys = {"totalUsers":int,
                        "totalModules":int,
                        "activeUsers":int,
                        "questionsAttempted":int
                    }
    verify_keys(required_keys, data)
```
