Technical Documentation – Student Course Assignment Service

Author: Vanitha C N

Created: 05/07/2025

Modified: 07/07/2025

Purpose: Provide an in-depth reference for Student Course Assignment Service, including base URLs, endpoints, parameters, responses, and best practices.

1. Introduction

This guide explains how to integrate with the Student Course Assignment Service REST API. The service enables client systems to assign courses to students, retrieve assigned courses and create users in a secure and scalable manner.

2. Prerequisites

- Understanding of REST APIs
- API base URL and authentication token
- JSON support in application
- HTTPS-enabled development environment

3. Authentication

All API requests must include a Bearer token.

```
Authorization: Bearer <access-token>
```

Invalid or expired tokens return HTTP 401 Unauthorized.

4. Base URLs and Authentication

API Service	**Base URL**
Get student data	https://api.clicksafety.com/data/2.5/student/
Create student data	https://api.clicksafety.com /data/2.5/student/
Get course list	https://api.clicksafety.com /data/2.5/course/

5. Endpoints and Request Details

5.1 Student data

Method: GET

Endpoint: /GetStudent/{studentid}

Description: Retrieves student data with course assignment by studentId.

Parameter	**Required**	**Description**
studentid	Yes	Student id is mandatory and must be in alphanumeric, e.g., VA257458

Request:
https://api.clicksafety.com/data/2.5/student/GetStudent?studentid=STUDENT_ID

JSON Response: {
```
"studentid": {"va257648"},
"username": {"id": 803, "firstname": "John", "lastname":
"Doe","email":
"johndoe@gmail.com", "phonenumber":"+14533344545"},
  "courses": {"courseID": "MOOD323", "Title": "Hazwoper 2025",
"startDate": 7032341135, "completionDate": null, "assignDate":
70234433212}
}
```

5.2 Create student

Method: POST

Endpoint: /createUser

Description: Creates student data with personal information and assigns courses. Course data can be null when student is created.

Parameter	**Required**	**Description**
firstname	Yes	Firstname is mandatory for user creation e.g., John
lastname	Yes	Lastname is mandatory for user creation e.g., Doe
emailid	Yes	A valid email-id is mandatory for user creation e.g., johndoe@gmail.com
phonenumber	No	Phone number is optional for user creation
address1	No	Address1 field is optional eg., 2341, street 1
address2	No	Address2 field is optional e.g., Church Street, Winston
state	No	State field is optional e.g., Alabama

country	No	Country field is optional e.g., US
zip	No	Zip field is optional e.g., 43451
courseid	No	Course id is optional e.g., MOOD435

Example Request: https://api.clicksafety.com/data/2.5/student/CreateUser

Request body:

```
{
"firstname": "John",
"lastname": "Doe",
"firstname": "johndoe@gmail.com",
"phonenumber": "+12421165567",
  "courses": [
    { "courseId": "MOOD323 },
    { "courseId": "MOOD900"}
  ]
}
```

JSON Response:

```
{
  "studentid": {"FirstName": "John", "LastName": "Doe",
"email":"johndoe@gmail.com", "phonenumber":"+12421165567",
"address1":null,"address2":null,
"state":null,"country":null,"zip":null},
  "courses": [
    {"courseid": "MOOD323", "Title": "Hazwoper 2025",
"startdate":"","completiondate":"","assigndate":"" },
    {"courseid": "MOOD900", "Title": "OSHA Course 2025",
"startdate":"","completiondate":"","assigndate":"" }
  ]
}
```

5.3 Get all courses

Method: GET

Endpoint: /GetAllCourses

Description: Returns list of courses in the library.

Request: https://api.clicksafety.com /data/2.5/course/GetAllCourses

JSON Response:

```
{[
  {"courseid": “MOOD340”,
  "title": “Hazwoper 2025”,
  "description": “”},

  {"courseid": “MOOD30”,
  "title": “OSHA 10 HOUR Construction - 2025”,
  "description": “”}]}
```

6. Error handling

Status code	**Description**
400	Indicates invalid syntax or input errors that prevent the server from processing the request.
401	Indicates the request is missing authentication credentials or the provided credentials are invalid.
404	Indicates the requested resource is unavailable or does not exist on the server.
500	A generic error response indicating something went wrong on the server side

7. Best Practices

- Store authentication tokens securely.
- Cache responses for frequent requests to minimize API calls.
- Use proper error handling for HTTP codes (e.g., 401 Unauthorized, 404 Not Found).
- Validate input before sending requests.
- Implement retry logic for transient failures.
- Log errors only (avoid sensitive data logging).