YouTestMe

SDK-based Integration
with YouTestMe Proctoring

youtestme

Table of Contents

1 Introduction

Supervisor SDK software is a JavaScript library that can be connected to the page of any testing system and implement seamless integration with the proctoring system. The full SDK documentation is available at https://proctor-test1.youtestme.com/sdk/doc/.

The testing system must meet the following requirements:

1. Proctoring pages in the testing system must be opened using the HTTPS protocol, a valid SSL-certificate must be installed on the web server (you can check the validity of the certificate on your server using the SSLChecker.com service or issue Let's Encrypt certificate for free);
2. Test pages should not be completely reloaded when switching between questions, the SDK code should be launched at the beginning of the test and remain loaded until the end (the page can be reloaded by the user).

It should be noted that the following integration parameters may change:

- Server address: **proctor-test1.youtestme.com**
  use this server as an example only, in production always use the address of your proctoring server;
- Authorization provider: **jwt**
  the default name is this, but can be changed;
- JWT secret key: **eexae8phah3Pha1iereez3oo**
  used to sign the JWT token;
- Webhooks API key: **eexae8phah3Pha1iereez3oo**
  specified in the "X-Api-Key" header of the webhook to authenticate the request.

2 Proctoring scenario

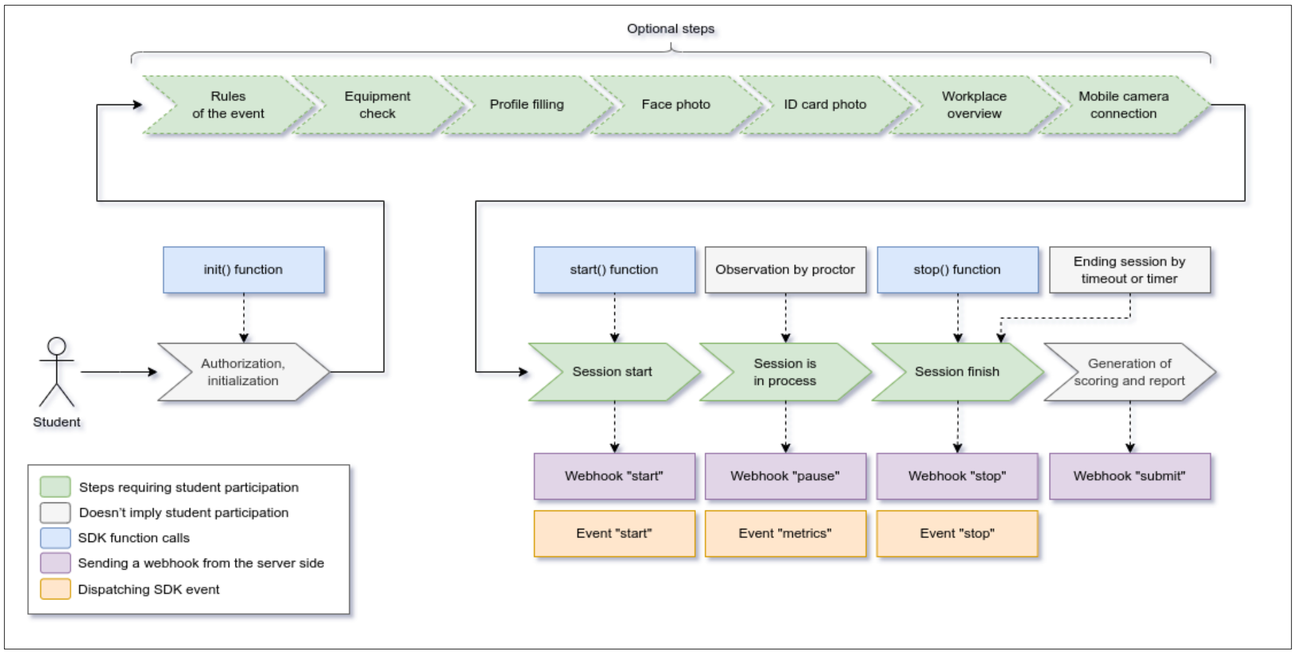
The scenario of interaction of the student and the proctor with the proctoring system and LMS in the general case is as follows:

1. the student logs in to the LMS, opens the test and initiates the start of the proctoring session;
2. before the session, the student goes through several stages (optional): agrees with the rules of the event, checks the equipment, takes a photo of the face and photo of the document, connects the mobile camera;
3. the student starts the test, the proctoring session starts in parallel;
4. during the exam, video recording of the webcam (with sound) and the computer screen is carried out, automatic tracking of violations and continuous verification/identification of the student are carried out;
5. during the session, the proctor can observe the students, the system in real time tells which student are worth paying attention to;
6. the proctor can interact with students in a chat or through video and audio communications, can prematurely end the proctoring session in case of gross violations;
7. after the exam is completed, an assessment of the level of confidence in the exam results and a video protocol with minute details of violations are formed;
8. results are passed to the LMS by webhook.

2.1 Technical implementation of the scenario

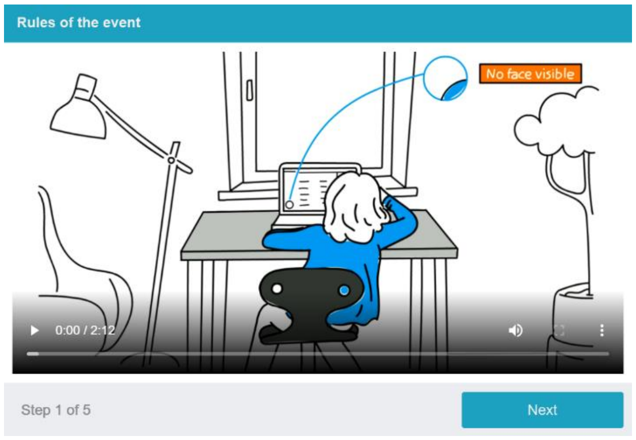
1. On the server side of your testing system (LMS), you need to implement the API for generating a JWT token and passing it to the front-end, where the proctoring session will be initialized through the Supervisor SDK in the init() function using this token.
2. The token consists of the following parts (see sections 2.1 and 2.2):
   - **Header** — remains unchanged;
   - **Payload** — user and session parameters in JSON format;
   - **Signature** — the signature is formed based on the Payload data and the secret key.
3. On the LMS front-end, you need to implement a mechanism for obtaining a token for the current user and test by API. The received JWT token will need to be substituted into the proctoring session initialization function init() in the "token" parameter of the Supervisor SDK library (see section 2.3).
4. At the moment of initialization of the proctoring session on the server side of the proctoring system, a user and a proctoring session are created (or updated) with the parameters specified in the JWT Payload. The participant sees the preliminary steps interface before starting proctoring.
5. Following the execution of the init() function, you need to execute the start() function, which will directly start the proctoring session for the participant. You can display the test itself to the participant only after the successful execution of the start() function (see section 2.4). Otherwise, the participant can access the test without proctoring.
6. After the test is completed, the stop() function must be executed so that the proctoring session is also terminated (see section 2.5).

7. After processing the session, the results are transmitted to the LMS via a webhook, the address of which is specified in the "api" parameter (in the JWT Payload), using a POST request in the "application/json" format.

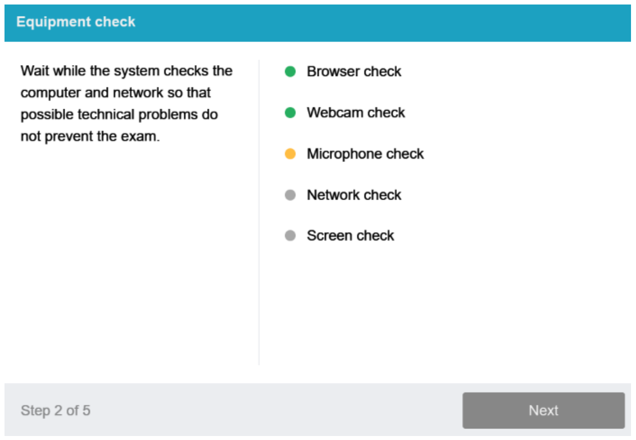


Before the start of the session, each user goes through a series of steps that are configured in advance and are necessary to fulfill all the conditions of the testing procedure (each step can be turned on or off). This includes: the rules of the event (requires user consent), equipment testing, photographing a person, photographing a document, connecting a mobile camera. Before starting the session, a computer scan starts, which includes checking the webcam, microphone, network, browser and screen capture. If there are no technical problems, the check takes place automatically, otherwise the user is given a message with a description of the problem and options for solving it. After a successful check, a proctoring session is started in which the user is monitored.
The following are the interfaces that can be displayed to the user before starting a proctoring session. Interfaces are displayed on top of the current page of the testing system and do not require the preparation of a special place for their display.

a) Rules and regulations of the exam;



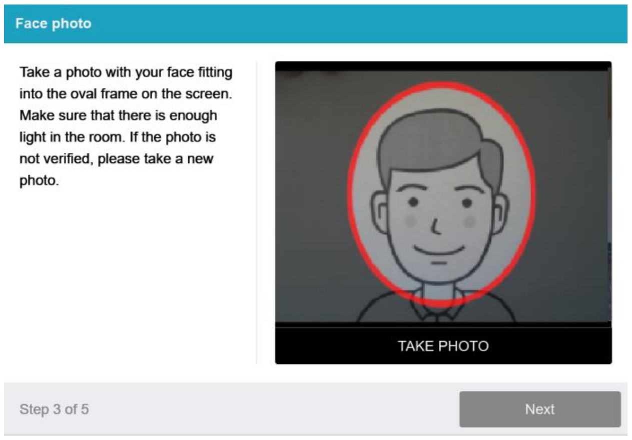
b) Computer check;


Equipment check

Wait while the system checks the computer and network so that possible technical problems do not prevent the exam.

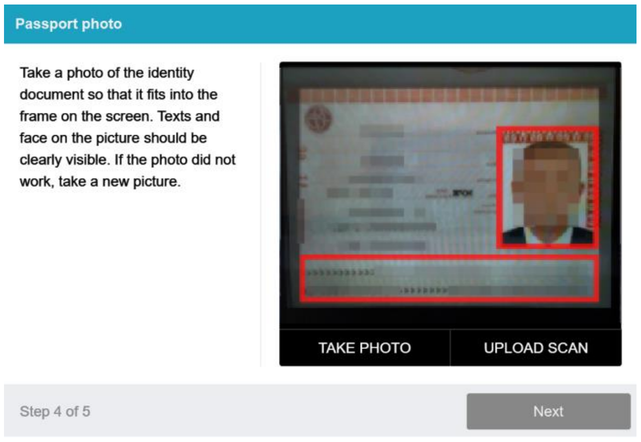
- ● Browser check
- ● Webcam check
- ● Microphone check
- ● Network check
- ● Screen check

c) Facial capture;


Face photo

Take a photo with your face fitting into the oval frame on the screen. Make sure that there is enough light in the room. If the photo is not verified, please take a new photo.

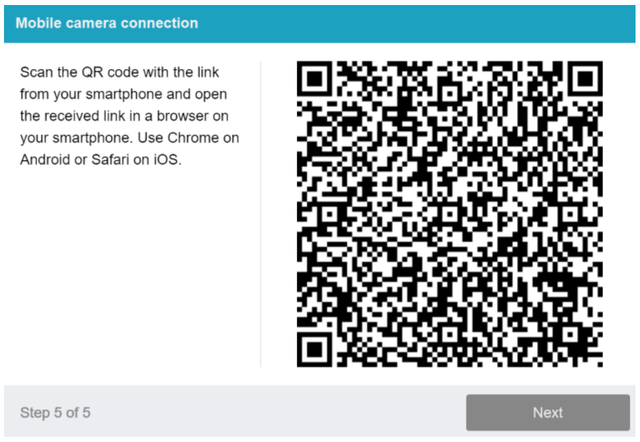
d) ID capture;



Passport photo

Take a photo of the identity document so that it fits into the frame on the screen. Texts and face on the picture should be clearly visible. If the photo did not work, take a new picture.

e) Smartphone camera connection.

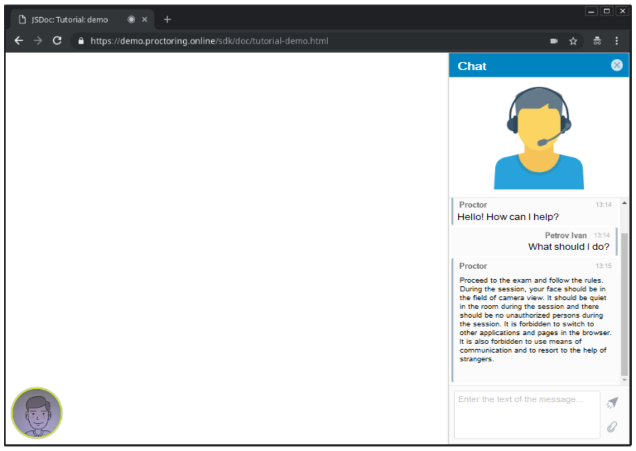


Mobile camera connection

Scan the QR code with the link from your smartphone and open the received link in a browser on your smartphone. Use Chrome on Android or Safari on iOS.

After successfully completing these steps, the observation mode starts. In this mode, the user can display video from his camera (preview), notification of problems (violations detected by the system automatically), and you can also provide access to the chat to contact the proctor. A proctor can contact the user at any time through chat or video and audio communications. The proctor can terminate the session ahead of schedule, indicating the conclusion (positive or negative) and comment, in which case the user is shown a corresponding message. The session page should be opened in a single copy, the system monitors this independently. If several pages with proctoring are opened in the browser, then the previous pages are automatically blocked and proctoring in them stops without finishing the session itself. You can continue the session only in the last open tab.

a) Video from camera (in circle) and chat with proctor (text, audio, video);

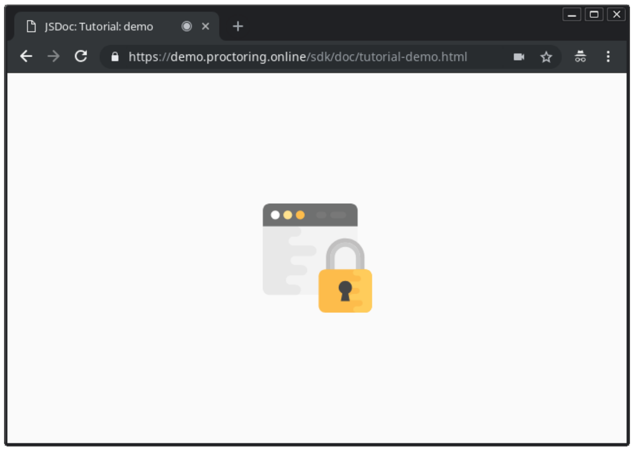

JSDoc: Tutorial: demo
https://demo.proctoring.online/sdk/doc/tutorial-demo.html
Chat
Enter the text of the message...

Proctor 13:14
Hello! How can I help?

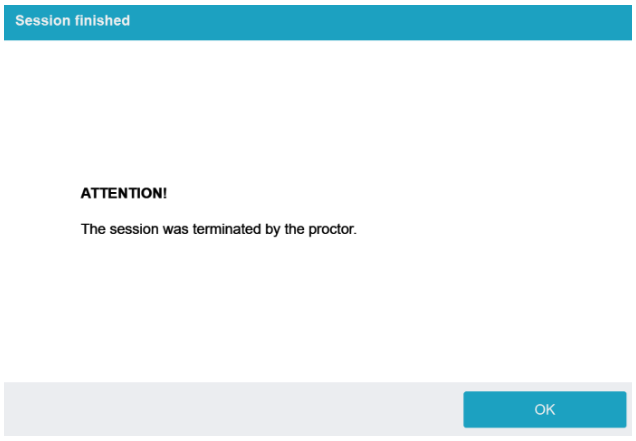
Petrov Ivan 13:14
What should I do?

Proctor 13:15
Proceed to the exam and follow the rules. During the session, your face should be in the field of camera view. It should be quiet in the room during the session and there should be no unauthorized persons during the session. It is forbidden to switch to other applications and pages in the browser. It is also forbidden to use means of communication and to resort to the help of strangers.

b) Content blocking in incident mode;



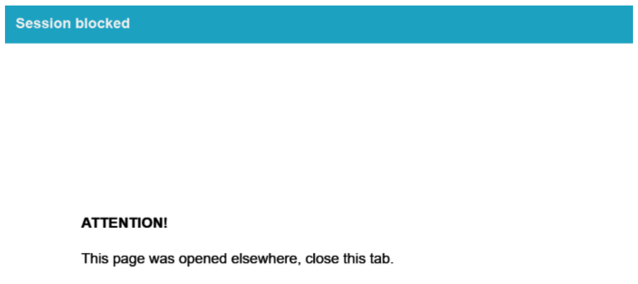
c) The session is completed by proctor;



ATTENTION!

The session was terminated by the proctor.

d) The page has been re-opened in another tab.

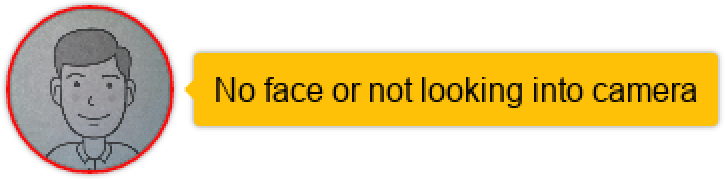


Session blocked

ATTENTION!

This page was opened elsewhere, close this tab.

Notifications of problems (irregularities) are automatically displayed to the user, thereby giving him/her an opportunity to rectify the situation if the problem arose unintentionally. If the user does not respond to messages within a minute, then access to the content is automatically blocked until the problem is fixed (this behaviour is controlled by the “addons[].lock” option).

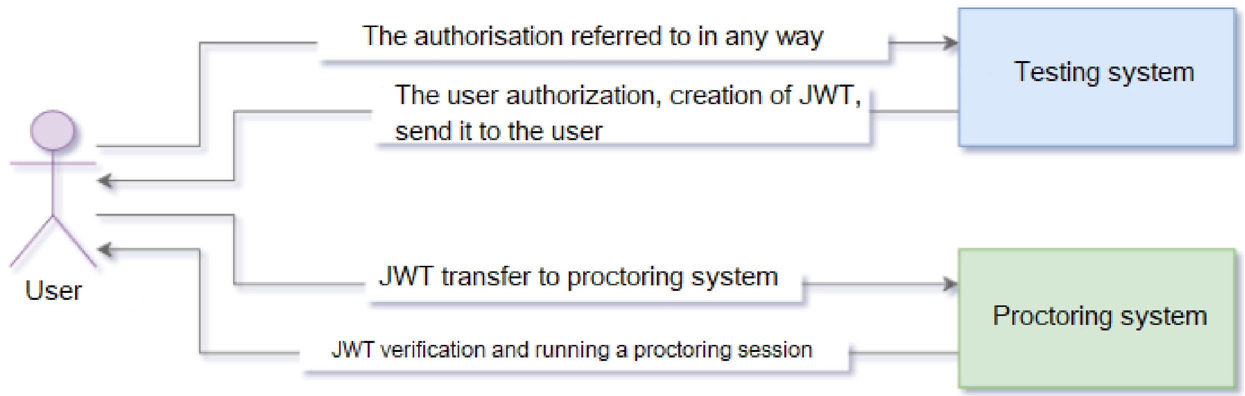


Content copy protection mode prohibits copying text and pictures from the page through the clipboard and context menu, also it prohibits saving the page through the print dialog.
After the test is completed, the proctoring session is stopped and all interfaces are hidden. Proctoring results are transferred to the testing system by webhook from the proctoring server.

3 SDK integration

3.1 JSON Web Token Standard (RFC 7519)

For secure transmission of parameters for a proctoring session with protection from changes on the part of the user, tokens are used according to the JSON Web Token (RFC 7519). The integration consists in implementing the JWT token generation mechanism on the side of the testing system and using the SDK library functions to manage the proctoring session.
JSON Web Token (JWT) is a JSON object that is defined in the open standard RFC 7519. It is considered one of the safest ways to transfer information between two participants. To create it, you need to define a header with general information on the token, payload data, such as the user name, his role, etc. and signature.
The diagram below shows the principle of user interaction with testing and proctoring systems using JWT.


The authorisation referred to in any way
Testing system
The user authorization, creation of JWT, send it to the user
User
JWT transfer to proctoring system
Proctoring system
JWT verification and running a proctoring session

The general interaction scenario is as follows:

1. The user of the testing system is authorized in it using the authorization mechanisms provided by the testing system.
2. Before the start of each individual proctoring session, the testing system generates a JWT using a predefined secret key that should not be accessible to users. The key storage method is selected by the testing system itself.
3. The JWT is transmitted to the user, and the user initiates a session in the proctoring system via the SDK using the JWT token received from the testing system. The JWT stores user identifier (username field), proctoring session identifier (identifier field), and other session parameters.
4. The proctoring system server receives the JWT, checks its validity, and then starts the proctoring session for this user.

For more information on how JWT works, see the article: “Understanding JSON Web Tokens (JWT) in 5 Easy Steps”

3.2 Token Generation

Token generation should be performed on the server, and only the token string should be transmitted to the client. To create a token:

1. Generate payload data in JSON format that describes the user and a specific session of this user in the following format:

JWT payload
{ "username": "a34c1a1a-53ef-4728-8dc5-9c4779a8586e", "nickname": "John Doe", "identifier": "565b30b8-5cfb-42e2-a292-478d20630d1b", "template": "default", "subject": "Tutorial: proctoring", "tags": ["male"] }

```
{
  "username": "a34c1a1a-53ef-4728-8dc5-9c4779a8586e",
  "nickname": "John Doe",
  "identifier": "565b30b8-5cfb-42e2-a292-478d20630d1b",
  "template": "default",
  "subject": "Tutorial: proctoring",
  "tags": [ "male" ]
}
```

2. Generate a token on the server of the testing system based on payload data using the library (see the information on the site jwt.io). The encryption algorithm is HS256, the type of token is JWT. For the proctor-test1.youtestme.com server, the secret key for generating tokens is "eexae8phah3Pha1iereez3oo". Using the data above, the following token should turn out:

Token
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VybmFtZSI6ImEzNGMxYTFhLTUzZWYtNDcyOC04ZGM1LTljNDc3OWE4NTg2ZSIsIm5pY2tuYW1lIjoiSm9obiBEb2UiLCJpZGVudGlmaWVyIjoiNTY1YjMwYjgtNWNmYi00MmUyLWEyOTItNDc4ZDIwNjMwZDFiIiwidGVtcGxhdGUiOiJkZWZhdWx0Iiwic3ViamVjdCI6IlR1dG9yaWFsOiBwcm9jdG9yaW5nIiwidGFncyI6WyJtYWxlIl19.Sr3fp-kzjVPZX4pvKgF7zzhIblJMA1eT7gKhJgSYIXU

A description of the parameters that can be used in JWT payload is given in Table 2.1. Please try to use the required minimum of parameters, usually these are: exp, username, nickname, template, identifier, subject, tags, api. It is advisable to use the rest of the parameters only if their change is foreseen on the side of the testing system. If you need to specify different session parameters for different cases it would be better to use the templates.

Table 2.1. Description of parameters

Parameter	**Type**	**Description**
JWT Validation		
exp*	Number	UNIX time (in seconds), after which the token becomes invalid

User Profile		
username*	String	unique user id, use of characters is allowed: A-Za-z0-9_-, there should be only one ID for each user
role	String	user's role ("student" or "proctor"), default is " student"
nickname	String	user's visible name
group	String	user's group
labels	String[]	additional user data
lang	String	interface language ("en"), the default language is the browser language
referrer	String	redirection page after logging out of the proctoring system
Proctoring Session		
identifier*	String	unique external proctoring session identifier, use of characters is allowed: A-Za-z0-9_-, each individual session must have its own ID
template*	String	template ID for the proctoring session, use of characters is allowed: A-Za-z0-9_-
subject	String	title of session (exam)
timeout	Number	the session timeout in minutes, after which it should be stopped if the session is inactive (integer)
deadline	Date	deadline of session availability, it cannot be started after this date (in ISO-8601 format **)
invites	String[]	a list of users to add to the members in the session
tags	String[]	tags for search
url	String	URL for a test page for IFRAME integration
api	String	API address to send results for this session to another server

* — required parameters
** — ISO-8601 format (in UTC): YYYY-MM-DDTHH:mm:ss.sssZ

3.3 Token Usage

In general, working with SDK for a user with the "student" role means inserting a small JavaScript code on each page where proctoring is needed. It is essential that the "supervisor.js" script is loaded from the same proctoring server where the connection is embedded in the code. Otherwise, after the proctoring server update, an SDK version and a server version may be different, which will cause problems in functioning of the proctoring.

It is necessary to pass token to the client to the browser and use it in the init() function:

HTML

```html
<script src="//proctor-test1.youtestme.com/sdk/supervisor.js"></script>
<script>
 // create an instance of the Supervisor class
 var supervisor = new Supervisor({
  url: 'https://proctor-test1.youtestme.com'
 });
 // initializing a proctoring session
 // the token field you can specify a string,
 // a function or a promise
 supervisor.init({
  // to indicate that data is transmitted in the format of a JWT
  provider: 'jwt',
  // get string with JWT token from your server
  // your server side must have the appropriate API implemented
  token: fetch('/api/token').then(function(response) {
   if (response.ok) return response.text();
   else throw Error('Failed to get JWT');
  })
 }).then(function() {
  // start proctoring session immediately after initialization
  return supervisor.start();
 }).then(function() {
  // start testing in the e-learning system here
 }).catch(function(err) {
  // in case of an error, display the appropriate message
  alert(err.toString());
  // redirect to home page,
  // to prevent the test from starting without proctoring
  location.href = '/';
 });
```

```
</script>
```

In most cases, for example, when the page is refreshed in the browser, the session can be resumed. In this case, use the same identifier and username in token payload and re-call the init() function.

3.4 Session Start

After the init() function is successfully executed, you should start a proctoring session with the start() function. Functions run asynchronously and return a promise, so the start() function should be called only after the init() function has been successfully completed.
When the session is successfully started, the "start" event is called, which can be subscribed to using the following script:

HTML

```
<script>
  // Subscribe to session start event
  supervisor.on('start', function() {
    console.log('started');
  });
</script>
```

3.5 Session Completion

When completing a test, it is also necessary to call the stop() function to end the proctoring session. Otherwise, the last minute may not be saved, and the session will be terminated automatically only after some time specified in the timeout.

HTML

```
<script>
  // stop the session
  supervisor.stop()
    .then(function() {
      // log out the session
      return supervisor.logout();
  });
</script>
```

Once the session is ended (the session can be finished by a participant, a proctor or automatically by timeout and deadline values), the "stop" event is called, which can be subscribed to as follows:

HTML

```
<script>
  // Subscribe to the session completion event
  supervisor.on('stop', function() {
    console.log('stopped');
  });
</script>
```

3.6 Metrics event

When the session is started, you can subscribe to the “metrics” event which triggers every 10 seconds and provides the violation score for each metric.

The code sample below shows how you can subscribe to the “metrics” event when the proctoring session is started:

HTML

```
<script>
        supervisor.on(['metrics'], function(data) {
                console.log('Metrics event data:' + JSON.stringify(data));
                metricsData = data;
                if (data.metrics.violated ) {
                        console.log('Violation is detected.');
                        // If the exam should be terminated on violation of any metric, you can call exam termination from here. Exam termination should call supervisor.stop() and supervisor.logout().
                }
                if (data.metrics.c2 * data.weights.w2 > 100 - data.threshold ) {
                        console.log('Violation detected: Face invisible or not looking into the camera.');
                        // If the exam should be terminated on violation of specific metric, you can call exam termination from here. Exam termination should call supervisor.stop() and supervisor.logout().
                }

        });
</script>
```

The sample JSON of data of metrics event:

JSON

```
{
  "metrics":{
    "c1":0,
    "c2":10,
    "c3":5,
    "m1":0,
    "m2":4,
    "n1":0,
    "s1":0,
    "n2":0
  },
  "weights":{
    "c1":4,
    "c2":4,
    "c3":4,
    "m1":4,
    "m2":1,
    "n1":4,
    "s1":4,
    "n2":4
  },
  "peak":"c2",
  "score":36,
  "threshold":50,
  "violated":true
}
```

The explanation of the JSON fields of data of metrics event:

Parameter	Type	Description
metrics	Json	Json object containing the percentage violation of each metric in the during the last 10 seconds.
weights	Json	Json object containing the weight of each metric for calculating the overall score.
peak	String	The metric with highest violation percentage.
score*	Number	Overall score in the last 10 seconds.

threshold	Number	When the overall score is below this value it is considered that violations are made.
violated	Boolean	True if the overall score is below the threshold.

*The overall score is calculated using the following formula:

$$E = 100 - \sum_{k \in M} \left(w_k x_k \right)$$

where E ∈ [0, 100] is the overall score (if E < 0, then E = 0), xk is the violation percentage of the metric k, wk is the weighting coefficient of the metric k, M ∈ {b1,b2,c1,c2,...} are the metrics.

The list of all metrics:

- b1 - Browser is not supported.
- b2 - Focus changed to a different window.
- b3 - Full-screen mode is disabled.
- c1 - The webcam is disabled.
- c2 - Face invisible or not looking into the camera.
- c3 - Several faces in front of the camera.
- c4 - Face does not match the profile.
- c5 - Found a similar profile.
- k1 - Atypical keyboard keywriting.
- m1 - The microphone is muted or its volume is low.
- m2 - Conversation or noise in the background.
- n1 - No network connection.
- n2 - No connection to a mobile camera.
- s1 - Screen activities are not shared.
- s2 - A second display is used.

3.7 Token Based Links

Token can also be used to initialize and start a proctoring session by a link without using SDK if for some reason SDK on the test page fails. To do this, just create a link in the following format (token is substituted for "jwt"; if several integrations are connected, instead of "jwt" you need to specify the name of the authorization provider):

URL
https://proctor-test1.youtestme.com/api/auth/jwt?token=<JWT>
Explanations: • proctor-test1.youtestme.com — the proctoring system server address, • jwt — name of the authorization provider,

- <JWT> — the generated token.

Following the link a student will get into the proctoring system, and after passing the test and exiting the proctoring system, he/she will be redirected back to the testing system. In this case the test page opens in IFRAME of the proctoring system, and its address is taken from the "url" field. In order for the page to open correctly in IFRAME, you must configure the Content-Security-Policy and X-Frame-Options correctly on the test page (either do not pass these headers at all) and make sure HTTPS with valid SSL certificate is supported. You should also take into account that there were changes to cookie setup policy Set-Cookie in browsers Chrome 80+. It requires adding two configurations "SameSite=None; Secure" for trans-domain cookies. Here is an example of response header options which allows you to open a third-party page in IFRAME on a proctoring system page and use cookies in IFRAME:

HTTP Header
Set-Cookie: <cookie-name>=<cookie-value>; SameSite=None; Secure
Content-Security-Policy: frame-src https://*.youtestme.com

3.8 Data-attributes for automated initialisation

For simple SDK use cases, there is an automatic initialisation and session startup, as well as a payload transfer using the data attributes of the script (the "script" tag).
Table 2.2 lists the supported data attributes.
Table 2.2 - Data attributes for automatic initialisation

Parameter	Description
data-supervisor	If this parameter is set up it indicates that automatic SDK initialisation is required. Two values are supported: start - the proctor session is initialised and started; init - only the session is initialised without starting.
data-provider	This parameter indicates which provider is used on the initialisation, e.g., signup, login or plain.
data-referrer	This parameter defines a URL which will be used as a reference once the test is finished and closed.
data-*	Other possible fields for a payload which can be used for the specified provider.

This initialisation method can be used in conjunction with other integrations, such as LTI or JWT links. In this case, an additional parameter "redirect" is specified in the authorisation link, which must contain the address of the page (/api/auth/<provider>?redirect=/path/to/sdk) that hosts the code with automatic SDK initialisation with one single data attribute "data-supervisor". A code sample:

HTML
<script src="https://proctor-test1.youtestme.com/sdk/supervisor.js" data-supervisor="start"></script>

The "data-supervisor" attribute with the value "init" can be used to start preliminary proctoring steps (such as hardware checks) without starting a proctoring session. A code sample:

HTML
<script src="https://proctor-test1.youtestme.com/sdk/supervisor.js" data-supervisor="init"></script>

You can use the "signup" provider to start a proctoring session on the test system page with manual registration for the event. A code sample:

HTML
<script src="https://proctor-test1.youtestme.com/sdk/supervisor.js" data-supervisor="start" data-provider="signup"></script>

4 Monitoring and access to proctor protocols

A user with the “Proctor” role can log in to real-time sessions and view protocols by logging in to the proctoring system using JWT token, login and password or by opening protocol using a unique link. In the monitoring interface, a proctor sees only those sessions in which he/she is a member.
A proctor can log into the proctoring system by login and password. For this aim, proctors’ accounts must be created in advance through the admin panel, and their logins must be added to the correspondent sessions (the "invites" field).
You can create special links to go to a specific session protocol. Using these links, you can open the session protocol without being a system user. To allow opening sessions by a link without authorization, you need to enable the "addons[].shared" option for a particular session (you can also enable this option in the template). In this case, it is recommended to use complex session identifiers that are protected from attacks. A link to protocol is formed as follows:

URL
https://proctor-test1.youtestme.com/api/report/<identifier>
Explanations: • proctor-test1.youtestme.com — the proctoring system server address, • <identifier> — the session identifier.

You can authorize a specific proctor with automatic account creation in the proctoring system using a link with a JWT token. Here is an example of filling in the fields of JWT token for authorizing a proctor:

JWT payload

{
  "username": "proctor1",
  "role": "proctor"
}

Here you can see an example of a link to protocol with authorization by JWT token:

URL
https://proctor-test1.youtestme.com/api/auth/jwt?token=<JWT>
Explanations:

- proctor-test1.youtestme.com — the proctoring system server address,
- jwt — name of the authorization provider,
- <JWT> — the generated token.

In addition, you can specify the session identifier in the JWT ("identifier" field), then the protocol of the specified session will be opened for the proctor immediately after authorization.

5 Getting the results of proctoring

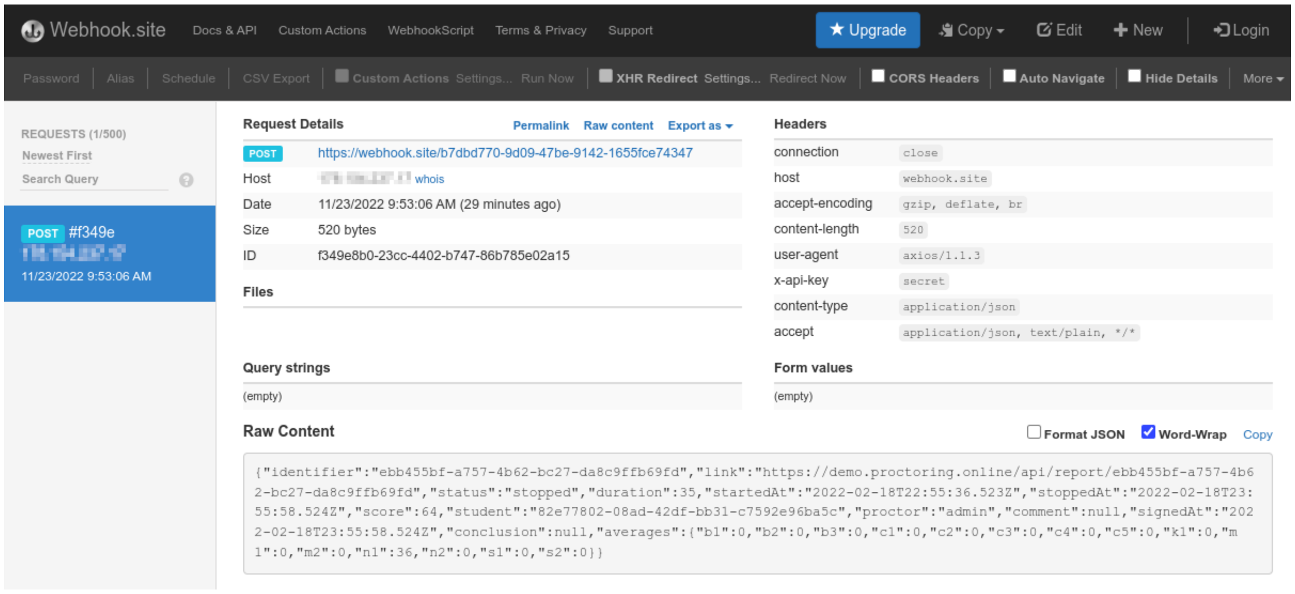
The proctoring system can transfer the results of proctoring sessions to the testing system using webhooks in JSON format. This is useful when you want to combine test results and proctoring results directly in the testing system. The results are transmitted immediately after the end of the session by the student or proctor, when the session is automatically terminated by timeout or when the proctor's conclusion is issued (or changed) after checking the protocol for some time after the end of the session.
The results are transmitted by an HTTP request by the POST method to the testing system server from the side of the proctoring system server using the URL specified in the "api" field of each session. The content type ("Content-Type" header) is "application/json". Access to the API should be limited by the key passed in the header of the HTTP request (by default, this is the "X-Api-Key" header). A request with one "identifier" can be executed several times with different data, therefore, it is necessary to provide for their updating in the testing system when data is received again from the proctoring system. If all is well, then the response code should be 200, the content of the response body is not taken into account and is usually left blank. In case of an error, the API should return a response code other than 200.

Table 4.1. Field description

Field	**Type**	**Description**
identifier*	String	Session ID
status	String	Session status: **started** - started, in the process of work **skipped** - skipped, was not started before the deadline **stopped** - completed but not rated by the proctor **accepted** - positively rated by the proctor **rejected** - negatively rated by the proctor
duration	Number	The actual duration of the session (minutes)
startedAt	Date	Date and time of the actual start of the session
stoppedAt	Date	Date and time of the actual end of the session
score	Number	Automatic evaluation for proctoring sessions (0-100)
averages	Object	The list of average indicators per session for each metric separately in the format "metric: value"
student	String	Student's username
proctor	String	Proctor's username who rated the session
comment	String	Proctor's comment on the session

link	String	Link to the session protocol

* — required fields, other fields may not be transmitted or take "null".



[redacted]

11/23/2022 9:53:06 AM

Request Details

POST	https://webhook.site/b7dbd770-9d09-47be-9142-1655fce74347
Host	[redacted] whois
Date	11/23/2022 9:53:06 AM (29 minutes ago)
Size	520 bytes
ID	f349e8b0-23cc-4402-b747-86b785e02a15

Headers

connection	close
host	webhook.site
accept-encoding	gzip, deflate, br
content-length	520
user-agent	axios/1.1.3
x-api-key	secret
content-type	application/json
accept	application/json, text/plain, */*

Files

Query strings

(empty)

Form values

(empty)

Raw Content

```
{"identifier":"ebb455bf-a757-4b62-bc27-da8c9ffb69fd","link":"https://demo.proctoring.online/api/report/ebb455bf-a757-4b6
2-bc27-da8c9ffb69fd","status":"stopped","duration":35,"startedAt":"2022-02-18T22:55:36.523Z","stoppedAt":"2022-02-18T23:
55:58.524Z","score":64,"student":"82e77802-08ad-42df-bb31-c7592e96ba5c","proctor":"admin","comment":null,"signedAt":"202
2-02-18T23:55:58.524Z","conclusion":null,"averages":{"b1":0,"b2":0,"b3":0,"c1":0,"c2":0,"c3":0,"c4":0,"c5":0,"k1":0,"m
1":0,"m2":0,"n1":36,"n2":0,"s1":0,"s2":0}}
```

The webhook.site service can be used to test the webhook results API. To do this, in the "api" field of the session created via JWT, specify the URL that is listed on the page of the service. Then start and end the session and a few minutes after the session is over a request will come for the specified URL. You will be able to see the request parameters in the interface of this service.

6 Integration checklist

To verify that the SDK integration is correct, use the following checklist:

1. The supervisor.js SDK script must be downloaded from the same server to which you are connecting, otherwise the server and SDK versions will be different, which will lead to errors in the SDK. The proctor-test1.youtestme.com server can only be used to develop and verify your integration.
2. Initialization of proctoring with init() is usually performed just before testing with proctoring. During the initialization process, the user can see interfaces with preliminary steps (event rules, equipment check, photos, mobile camera connection). The init() function may return an error, in which case the test cannot be started.
3. Proctoring is started by the start() function when testing starts. In the process of starting proctoring, access to the camera, microphone and screen may be requested. After successful proctoring, a circle with the participant's camera may be displayed on the screen. Training or some preliminary actions should not be included in the proctoring session, only the test itself (with a time limit), otherwise it may negatively affect the assessment of trust. The start() function may return an error, in which case the test cannot be started.
4. When refreshing the page (F5 or closing and reopening), the test and proctoring session should be restored in the same way as the first run through the init() and start() functions, in this case the session identifiers (identifier field) and user (username field) should not change.
5. After the test is completed, the proctoring session should be terminated with the stop() function and then it is recommended to exit through logout().
6. Once a proctoring session is finished, access to the test should also be closed. A proctoring session can end at the initiative of the participant, the proctor or automatically by timeout and deadline. The stop event subscription can be used to handle this scenario.
7. You can use the service webhook.site to test the transfer of results via webhooks.