СИСТЕМА ВИДЕОМОНИТОРИНГА И ИДЕНТИФИКАЦИИ ОБУЧАЮЩЕГОСЯ В ДИСТАНЦИОННОМ ОБУЧЕНИИ

К.Г. Виноградов
Научный руководитель: А.С. Фадеев
Томский политехнический университет
E-mail: kgv1@tpu.ru

Введение

В наше время для получения образования необходим лишь доступ к интернету

Дистанционное образование позволяет студенту самому выбирать время для обучения, ведь лекции и тестирования можно пройти в удобное для студента время. Дистанционное образование позволяет студенту легче совмещать учебу и работу, благодаря гибкому графику, а также позволяет ему при желании спокойно переехать в другой город.

Однако, дистанционное образование уступает очной форме обучения, когда речь идет о доверии к результатам оценивающих мероприятий и выданным по этим результатам дипломам, свидетельствам и сертификатам. Связано это со сложностью идентификации обучающихся. Сложно уследить самостоятельно ли студент выполняет задания или тестирования.

Таким образом, возникает задача повышения доверия к результатам оценивающих мероприятий, проводимых в дистанционном образовании. Данная работа направлена на решение выявленной задачи.

Требования к системе

К проектируемой системе видеомониторинга и идентификации обучающихся предъявляются следующие требования:

- Все действия происходя на одной странице.
- Не нужно устанавливать дополнительное ПО.
- Тестирование снимается на видео.
- Запись процесса тестирования храниться на сервере.
- У преподавателя есть возможность просмотреть запись тестирования.
- Идентификация пользователя проводиться в процессе тестирования.

При открытии теста студентом должна начаться инициализация соединения с Janus сервером. Когда студент начнет тестирование должны параллельно начаться передача видео на сервер и процесс идентификации студента. При завершении тестирования передача видео должна прекратиться и видеофайл должен сохраниться на сервере.

При открытии теста преподавателем должна начаться инициализация соединения с Janus сервером. При нажатии на кнопку «посмотреть видео» видеофайл должен передаться с сервера, после чего преподаватель мог бы его посмотреть.

Процесс тестирования на платформе дистанционного обучения после внедрения системы видеомониторинга и идентификации пользователя представлен на рисунке 1.

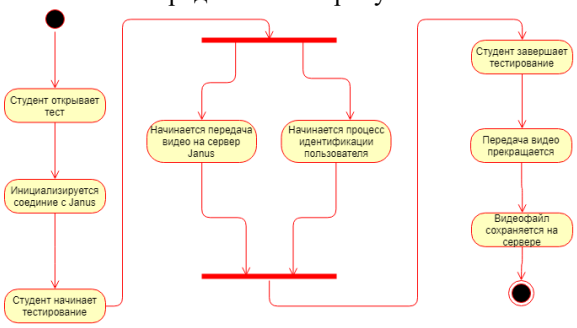

Рис. 1. Диаграмма деятельности студента

Обзор технологий

Разработка системы видеомониторинга и идентификации обучающихся была проведена для системы Moodle с помощью языков программирования PHP, JavaScript, языком разметки HTML, с помощью технологии WebRTC, проекта Janus WebRTC Server и библиотеки face-api.js.

Для передачи видео с различных браузеров и с минимальной задержкой была выбрана технология WebRTC.

WebRTC (Web Real Time Communications) – это бесплатный проект с открытым исходным кодом, который предоставляет веб-браузерам и мобильным приложениям возможности взаимодействия в режиме реального времени через простые интерфейсы прикладного программирования (API) [1].

Для того чтобы вручную не настраивать медиа-связь с браузером был использован WebRTC сервер общего назначения – Janus WebRTC Server.

Сервер Janus WebRTC был задуман как сервер общего назначения. Как таковой, он не предоставляет никаких функциональных возможностей, кроме реализации средств для настройки медиа-связи WebRTC с браузером, обмена сообщениями JSON с ним и передачи RTP/RTCP и сообщений между браузерами и логикой приложения на стороне сервера, с которой они связаны [2].

Face-api.js – это API-интерфейс JavaScript для обнаружения и распознавания лиц в браузере, реализованный поверх основного API tenorflow.js. Он реализует серию свёрточных нейронных сетей (CNN), оптимизированных для Интернета и мобильных устройств [3].

Реализация веб-приложения

В первую очередь был реализован модуль, который с помощью технологии WebRTC записывает видео на сервер.

Захват видео реализован с помощью метода getUserMedia().

Метод getUserMedia() запрашивает у пользователя разрешение использовать до одного устройства ввода видео (например, камеру или общий экран) и до одного устройства ввода звука в качестве источника для MediaStream.

Если разрешение предоставлено, видео и звуковые дорожки, поступающие с этих устройств, доставляются на указанный обратный вызов. Если запрос на использование устройств отклонен, не существует совместимых устройств ввода или есть какого-либо другое условие вызывающее ошибку, обратный функции выполняется с MediaStreamError объектом, описывающим, что пошло не так [4].

Для облегчения работы с WebRTC было принято решение работать с проектом Janus. Для этого был внедрен JavaScript код для взаимодействия с Janus непосредственно в проект Moodle. Оказалось, что Moodle не использует асинхронного взаимодействия с сервером и во время прохождения теста страница постоянно обновляется.

Это говорит о невозможности взаимодействия Moodle с Janus напрямую, потому что WebRTC получает видеопоток с веб–камеры с помощью JavaScript, а во время обновления страницы все объекты инициализируются заново, что вызвало бы постоянное переподключение к серверу Janus.

Было принято решение о помещении страницы Moodle в тег <iframe>.

Тег <iframe> создает плавающий фрейм, который находится внутри обычного документа, он позволяет загружать в область заданных размеров любые другие независимые документы.

Тег <iframe> является контейнером, содержание которого игнорируется браузерами, не поддерживающими данный тег. Для таких браузеров можно указать альтернативный текст, который увидят пользователи. Он должен располагаться между элементами <iframe> и </iframe> [5].

Теперь страница стала статичной и ничего не мешает получению информации с веб-камеры и передаче ее на сервер. Но и это привело к проблеме. В браузерах определена политика безопасности, которая не позволяет главному окну манипулировать DOM–элементами iframe, если он ссылается на ресурс с другого домена.

Для решения этой проблемы был реализован обмен сообщениями между главным окном и iframe. Реализовать это удалось с помощью функции window.postMessage().

Window.postMessage() – этот метод позволяет безопасно отправлять кросдоменные запросы. Обычно сценариям на разных страницах разрешен доступ друг к другу только если страницы, которые их выполняли, передаются по одному протоколу (обычно это https), номер порта (443 — по умолчанию для https) и хост (modulo Document.domain установленный страницами на одно и тоже значение). window.postMessage() предоставляет контролируемый механизм, чтобы обойти это ограничение способом, который безопасен при правильном использовании[6].

При вызове метода window.postMessage() он вызывает MessageEvent для отправки в целевом окне, когда завершается любой ожидающий сценарий, который должен быть выполнен (например, оставшиеся обработчики событий, если window.postMessage() вызывается из обработчика событий ранее заданных ожидающих таймаутов). MessageEvent имеет тип message, data-свойство которого устанавливает значение первого аргумента в методе window.postMessage(), свойство origin соответствует адресу основного документа в вызове window.postMessage во время вызова window.postMessage(), свойство source указывает на окно, из которого window.postMessage() вызвали.

Реализация идентификации обучающегося

Идентификация обучающихся реализована с помощью библиотеки face–api.js.

Идентификация происходит путем определения лица на изображении, определении ориентиров лица и сравнении их с исходными данными, в результате чего формируется процент доверия, говорящий о том, насколько человек на изображение лица схоже с исходным изображением.

Заключение

Данная работа посвящена решению задачи повышения доверия к результатам оценивающих мероприятий, проводимых в дистанционной форме путем внедрения системы видеомониторинга и идентификации обучающихся в образовательную платформу Moodle.

Была реализована система видеомониторинга и идентификации обучающихся, так же она была внедрена в систему управления обучением Moodle.

Реализованная система соответствует всем предъявленным ему требованиям.

Список использованных источников

1. WebRTC. [Электронный ресурс] / WebRTC. – URL: https://webrtc.org (дата обращения 16.11.2019).
2. Janus [Электронный ресурс] Meetecho. URL: https://janus.conf.meetecho.com (дата обращения 16.11.2019).
3. face-api.js—JavaScript API for Face Recognition in the Browser with tensorflow.js [Электронный ресурс] itnext. URL: https://itnext.io/face-api-js-

javascript-api-for-face-recognition-in-the-browser-with-tensorflow-js-bcc2a6c4cf07 (дата обращения 16.11.2019).
4. Navigator.getUserMedia() [Электронный ресурс] MDN Web Docs. URL: https://developer.mozilla.org/en-US/docs/Web/API/Navigator/getUserMedia (дата обращения 16.11.2019).
5. Тег <iframe> [Электронный ресурс] htmlbook.ru. URL: http://htmlbook.ru/html/iframe (дата обращения 16.11.2019).
6. Window.postMessage() [Электронный ресурс] MDN web docs URL: https://developer.mozilla.org/ru/docs/Web/API/Window/postMessage Дата обращения: 06.06.2019. (дата обращения 16.11.2019).