Informatyka sem. VI

Projekt Podstawy Teleinformatyki:

Konferencje audio-video

Dokumentacja
Wykonali:
Mikołaj Fedec
Filip Krzemień

1. **Wstęp**

Przedmiotem naszego projektu były konferencje audio-wideo. Celem stworzenia tego systemu była możliwość prowadzenia rozmów online z innymi użytkownikami np. spotkań biznesowych bez zbędnego zakładania specjalnie konta. Przyjęliśmy następujące założenia: brak instalacji dodatkowego oprogramowania, cross platform oraz możliwość prowadzenia wielu rozmów jednocześnie. Jednak w przypadku przeglądarki Chrome i Opera należy zainstalować rozszerzenie by móc streamować swój ekran. Wybraliśmy ten temat, ponieważ wydało nam się interesujące sposób działania konferencji prowadzonych przez Internet oraz chcieliśmy poznać i nauczyć się nowych technologii, a w czasie studiów nie mieliśmy okazji poznać JavaScriptu, czy WebRTC, który po wstępnych badaniach wykorzystamy do zbudowania naszej aplikacji. Pracę udawało nam się wykonywać zgodnie z harmonogramem do czasu gdy trzeba było zaimplementować screen sharing. Był to jeden z większych problemów jakie napotkaliśmy, szczegółowo zostanie to opisane w sekcji “Problemy”.

2. **Podział prac**

Podział prac przedstawia się następująco:

Praca wspólna:

- analiza potrzebnych technologii i protokołów(praca wejścia),
- dokumentacja.

Mikołaj Fedec:

- otrzymywanie i dekodowanie danych,
- wyświetlanie danych,
- współdzielenie ekranu.

Filip Krzemień:

- implementacja signaling service,
- ustanowienie połączenia p2p,
- enkodowanie oraz wysyłanie danych.

3. **Funkcjonalności aplikacji**

Stworzona przez nas aplikacja posiada wymienione poniżej funkcjonalności:

- Logowanie - każdy nowy użytkownik, aby skorzystać z naszej aplikacji musi podać swój nickname. Nazwa ta musi być oczywiście unikalna inaczej nie można się zalogować. Ponadto nazwa ta musi mieć co najmniej trzy znaki i nie więcej niż dwadzieścia oraz zawierać wyłącznie znaki alfanumeryczne.
- Lista użytkowników - lista przedstawiająca wszystkie zalogowane osoby.
- Czat globalny - czat na którym każdy z użytkowników może pisać i prowadzić konwersację.
- Czat prywatny - czat, który prowadzi się tylko z jedną osobą, aby rozpocząć taką rozmowę należy kliknąć na nią na liście zalogowanych użytkowników.
- Konferencja audio-wideo - rozmowa audio-wideo. Jest możliwość prowadzenia wielu rozmów w tym samym momencie. Dodatkowo istnieje możliwość personalizacji rozmieszczenia połączeń z innymi użytkownikami.
- Udostępnianie ekranu - udostępnianie widoku swojego ekranu/aplikacji/karty w przeglądarce podczas konferencji.

4. **Technologia**

W tym punkcie opiszemy zastosowane przez nas technologie oraz wyjaśnimy dlaczego z nich korzystamy. Po zapoznaniu się i zbadaniu problemu, użyliśmy następujących technologii oraz frameworków:

- **Node.js**

  Opis: środowisko uruchomieniowe zaprojektowane do tworzenia wysoce skalowalnych aplikacji internetowych, szczególnie serwerów www napisanych w języku JavaScript. Umożliwia tworzenie aplikacji sterowanych zdarzeniami wykorzystujących asynchroniczny system wejścia-wyjścia.Node.js składa się z silnika V8 (stworzonego przez Google), biblioteki libUV oraz kilku innych bibliotek. Został stworzony przez Ryana Dahla na początku 2009 roku, jego rozwój sponsorowany był przez firmę Joyent, w której pracował.Pierwotnym celem Dahla było dodanie do stron internetowych możliwości technologii push, widocznej w aplikacjach takich jak Gmail. Po wypróbowaniu różnych języków zdecydował się na JavaScript, ze względu na brak istniejącego API wejścia/wyjścia. Dało mu to możliwość wykorzystania nieblokującego, sterowanego zdarzeniami

wejścia/wyjścia. Domyślnym managerem pakietów dla Node.js jest Npm.

Uzasadnienie wyboru: wykonanie serwera na potrzeby projektu.

- **HTML5**:

  Opis: język wykorzystywany do tworzenia i prezentowania stron internetowych www. Jest rozwinięciem języka HTML 4 i jego XML-owej odmiany (XHTML 1), opracowywane w ramach prac grupy roboczej WHATWG(*Web Hypertext Application Technology Working Group*) i W3C(World Wide Web Consortium).HTML5 poza dodaniem nowych elementów, usprawniających tworzenie serwisów i aplikacji internetowych, doprecyzowuje wiele niejasności w specyfikacji HTML 4, dotyczących przede wszystkim sposobu obsługi błędów. Niejasności co do sposobu, w jaki przeglądarki powinny obsługiwać błędy w kodzie HTML są jedną z podstawowych przyczyn, dla której wiele serwisów internetowych, napisanych z naruszeniem specyfikacji, w różnych przeglądarkach działa w inny sposób – w niektórych działając, w innych nie. Dzięki HTML-owi 5 obsługa błędów ma być ta sama we wszystkich przeglądarkach, czyli *zły* element będzie działać w każdej przeglądarce albo żadnej.

  Uzasadnienie wyboru: wykonanie strony internetowej do projektu.

- **CSS**:

  Opis: język służący do opisu formy prezentacji stron WWW.Arkusz stylów CSS to lista dyrektyw (tzw. reguł) ustalających w jaki sposób ma zostać wyświetlana przez przeglądarkę internetową zawartość wybranego elementu (lub elementów) (X)HTML lub XML. Można w ten sposób opisać wszystkie pojęcia odpowiedzialne za prezentację elementów dokumentów internetowych, takie jak rodzina czcionek, kolor tekstu, marginesy, odstęp międzywierszowy lub nawet pozycja danego elementu względem innych elementów bądź okna przeglądarki. Wykorzystanie arkuszy stylów daje znacznie większe możliwości pozycjonowania elementów na stronie, niż oferuje sam (X)HTML. CSS został stworzony w celu odseparowania struktury dokumentu od formy jego prezentacji. Separacja ta zwiększa zakres dostępności witryny, zmniejsza zawiłość dokumentu, ułatwia wprowadzanie zmian w strukturze dokumentu. CSS ułatwia także zmiany w renderowaniu strony w zależności od obsługiwanego medium (ekran, palmtop, dokument w druku, czytnik ekranowy). Stosowanie zewnętrznych arkuszy CSS daje możliwość zmiany wyglądu

wielu stron naraz bez ingerowania w sam kod (X)HTML, ponieważ arkusze mogą być wspólne dla wielu dokumentów.

Uzasadnienie wyboru: Dokumenty pisane z wykorzystaniem arkuszy stylów są zwykle bardziej przejrzyste i krótsze. Style pozwalają w łatwy sposób zarządzać całą serią dokumentów, poprzez stosowanie zewnętrznych arkuszy stylów. Dzięki temu w łatwy i wygodny sposób, można dokonać modyfikacji rodzaju formatowania jednocześnie we wszystkich dokumentach, zmieniając dane tylko w jednym pliku.

- **JavaScript**:

  Opis: skryptowy język programowania, stworzony przez firmę Netscape, najczęściej stosowany na stronach internetowych. Najczęściej spotykanym zastosowaniem języka JavaScript są strony internetowe. Skrypty te służą najczęściej do zapewnienia interakcji poprzez reagowanie na zdarzenia, walidacji danych wprowadzanych w formularzach lub tworzenia złożonych efektów wizualnych. Skrypty JavaScriptu uruchamiane przez strony internetowe mają znacznie ograniczony dostęp do komputera użytkownika. Po stronie serwera JavaScript może działać w postaci node.js lub Ringo. W języku JavaScript można także pisać pełnoprawne aplikacje. Fundacja Mozilla udostępnia środowisko złożone z technologii takich jak XUL, XBL, XPCOM oraz JSLib. Umożliwiają one tworzenie korzystających z zasobów systemowych aplikacji o graficznym interfejsie użytkownika dopasowującym się do danej platformy. Przykładem aplikacji napisanych z użyciem JS i XUL może być klient IRC o nazwie ChatZilla, domyślnie dołączony do pakietu Mozilla. Microsoft udostępnia biblioteki umożliwiające tworzenie aplikacji JScript jako część środowiska Windows Scripting Host. Ponadto JScript.NET jest jednym z podstawowych języków środowiska .NET. Istnieje także stworzone przez IBM środowisko SashXB dla systemu Linux, które umożliwia tworzenie w języku JavaScript aplikacji korzystających np. z GNOME i OpenLDAP.

  Uzasadnienie wyboru: wykorzystujemy go zarówno podczas pisania serwera, jak i dodawania funkcjonalności do naszej aplikacji.

- **jQuery**:

  Opis: lekka biblioteka programistyczna dla języka JavaScript, ułatwiająca korzystanie z JavaScriptu (w tym manipulację drzewem DOM). Kosztem niewielkiego spadku wydajności pozwala osiągnąć interesujące efekty animacji, dodać dynamiczne zmiany strony, wykonać zapytania AJAX.

Większość wtyczek i skryptów opartych na jQuery działa na stronach nie wymagając zmian w kodzie HTML (np. zamienia klasyczne galerie złożone z miniatur linkujących do obrazków w dynamiczną galerię). Podstawowymi cechami są: niezależność od przeglądarki - eliminuje konieczność dostosowywania kodu do różnych przeglądarek WWW,małe rozmiary oraz wygoda przy tworzeniu wtyczek.

Uzasadnienie wyboru: jQuery pozwala w wygodny i zrozumiały sposób korzystać z następujących funkcjonalności: selektorów, atrybutów, manipulacji modelami DOM, definiowania własnych zdarzeń oraz rozbudowana ich obsługa.

- **Bootstrap**:

  Opis: framework CSS, rozwijany przez programistów Twittera. Zawiera zestaw przydatnych narzędzi ułatwiających tworzenie interfejsu graficznego stron oraz aplikacji internetowych. Bazuje głównie na gotowych rozwiązaniach HTML oraz CSS i może być stosowany m.in. do stylizacji takich elementów jak teksty, formularze, przyciski, wykresy, nawigacje i innych komponentów wyświetlanych na stronie. Framework korzysta także z języka JavaScript.

  Uzasadnienie wyboru: szybki, łatwy oraz przenośny między przeglądarkami sposób umożliwia tworzenie stron. Wiele gotowych

- **WebRTC**:

  Opis: standard HTML5 rozwijany przez World Wide Web Consortium, służący do komunikacji w czasie rzeczywistym poprzez przeglądarkę internetową.

  Uzasadnienie wyboru: łatwy sposób na utworzenie połączenia peer-to-peer pomiędzy użytkownikami oraz do przesyłania audio, wideo oraz informacji dotyczących połączenia.

  - **WebSocket:**

    Opis: jest technologią zapewniającą dwukierunkowy kanał komunikacji za pośrednictwem jednego gniazda TCP. Stworzono ją do komunikacji przeglądarki internetowej z serwerem internetowym, ale równie dobrze może zostać użyta w innych aplikacjach typu klient lub serwer. Została ustandaryzowana przez Internet Engineering Task Force w Request for Comments 6455 w 2011 roku a interfejs WebSocket w Web IDL jest standaryzowany przez World Wide Web Consortium.

5. **Architektura**

Projekt składa się z dwóch głównych elementów: serwera oraz aplikacji dla użytkownika. Zadaniem serwera jest dostarczanie aplikacji użytkownika oraz umożliwienie komunikacji i wymiany informacji o połączeniu WebRTC między użytkownikami.

Serwer jest utworzony za pomocą technologii node.js. Użytkownicy łącząc się z serwerem zostają dodani na listę obecnie zalogowanych. Aplikacja kliencka komunikuje się z serwerem poprzez wysyłanie odpowiednich komend enkapsulowanych w JSON. Przykład wysłania komendy:

```
send({
     type: "login",
     name: name
});
```

Funkcja *send*:

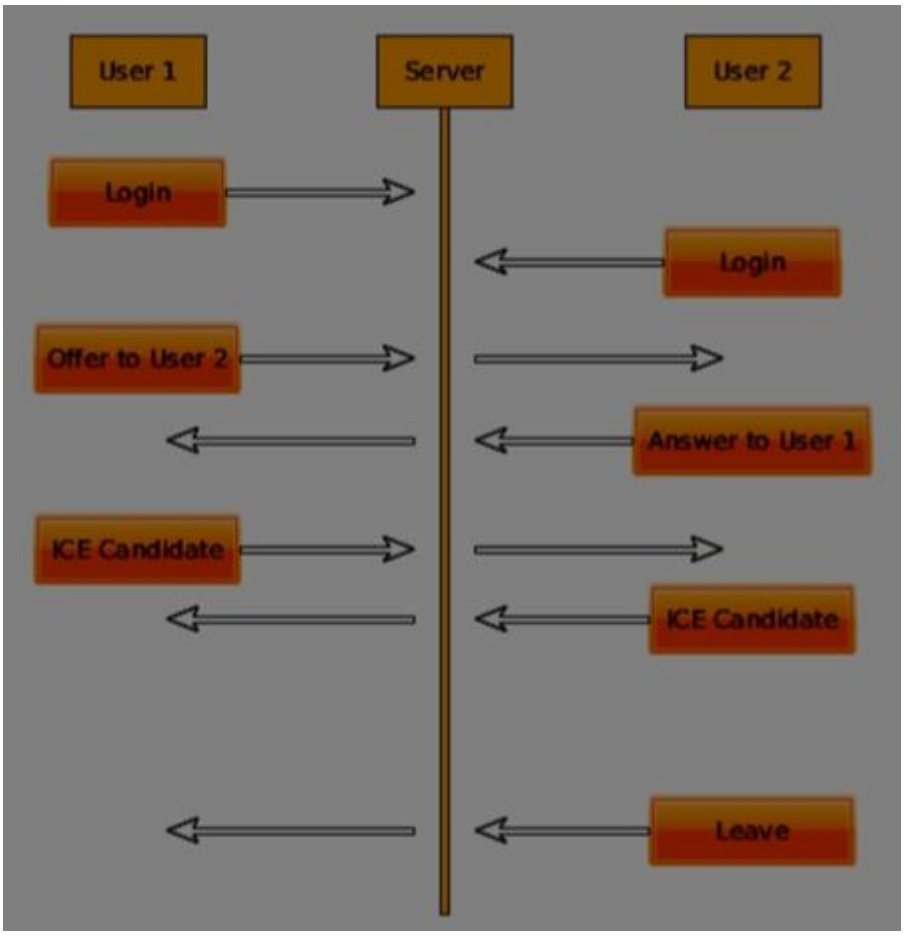
```
function send(message) {
    conn.send(JSON.stringify(message));
};
```

Dostępne komendy:

- login(*name*): prosi serwer o przydzielenie podanej nazwy użytkownika i zalogowanie się,
- users: prosi serwer o podanie aktualnej listy zalogowanych użytkowników,
- message(*message*, *name*): wysyła wiadomość tekstową o treści *message* do użytkownika o nazwie *name*,
- permission(*name*):
- accept(*name*):
- decline(*name*):
- offer(*offer*, *name*): wysyła ofertę połączenia WebRTC do użytkownika o nazwie *name*,
- answer(*answer*, *name*): wysyła odpowiedź połączenia WebRTC do użytkownika o nazwie *name*,
- candidate(*candidate*, *name*): wysyła kandydata WebRTC do użytkownika o nazwie *name*,
- leave(*name*): kończy połączenie z użytkownikiem o nazwie *name*.

Serwer do każdej wiadomości, której adresatem jest inny użytkownik dołącza także informacje, kto jest nadawcą.

W nawiązywaniu połączeń WebRTC pomaga też serwer STUN (stun2.1.google.com:19302). Jego zadaniem jest znajdowanie adresów zewnętrznych klientów, co jest potrzebne do utworzenia połączenia p2p.



Rys.1 Przykład nawiązywania połączenia

Użytkownicy 1 oraz 2 łączą się z naszym serwerem. Aplikacja użytkownika pokazuje, kto jest połączony z serwerem. Użytkownik 1 inicjuje nawiązywanie połączenia WebRTC poprzez wysłanie oferty do użytkownika 2, ten z kolei odsyła odpowiedź, po czym następuje wymiana kandydatów ICE. Celem użycia kandydatów ICE jest znalezienie bezpośredniego połączenia między użytkownikami i ustanowienie połączenia audio-video.

6. **Problemy**

Jednym z pierwszych problemów do rozwiązania było nawiązanie połączenia między użytkownikami. WebRTC jest połączeniem bezpośrednim, więc biorące w nim udział urządzenia muszą znać swoje adresy w sieci. Solucją na to jest użycie serwera sygnalizującego: użytkownicy nawiązują z nim połączenie, a serwer

pokazuje każdemu użytkownikowi listę, kto jest teraz on-line. Dzięki temu można wybrać do kogo chcemy zadzwonić. Oprócz tego, podczas nawiązywania połączenia WebRTC używany jest serwer STUN od Google. Użycie tego serwera pozwala urządzeniom na ustalenie swojego adresu i przekazanie go do drugiego urządzenia.

Aby umożliwić udostępnianie mediów przez przeglądarkę Chrome i Firefox przez serwer wymagany jest https serwer. HTTPS jest to szyfrowana wersja protokołu HTTP. W przeciwieństwie do komunikacji nie zaszyfrowanego tekstu w HTTP klient-serwer, HTTPS szyfrował dane przy pomocy protokołu SSL, natomiast obecnie używany jest do tego celu protokół TLS. Zapobiega to przechwytywaniu i zmienianiu przesyłanych danych. W celu przeniesienia aplikacji na HTTPS należy:

- Zmiana linków wewnętrznych z bezwzględnych (czyli takich zawierających pełny adres linkowanego zasobu) na względne (zawierające jedynie część adresu).Kiedy eliminuje się z adresu nazwę domeny lub protokołu, przeglądarka sama doda ten element na podstawie aktualnego adresu podstrony. W ten sposób, niezależnie od tego, czy strona używa protokołu HTTP czy HTTPS, zawsze będzie odwoływać się do strony z tym samym protokołem. Należy pamiętać, że mówimy o linkach wewnętrznych, jako że zewnętrzne strony internetowe mogą w ogóle nie obsługiwać HTTPS, dlatego też linki do nich zostawiamy w niezmienionej formie.
- Korekta załączników treści multimedialnych - Należy zmienić wszystkie odnośniki na względne również dla multimediów. Po przejściu na HTTPS Twoje media i treści również będą ładowały się z zabezpieczonych witryn. Warto się jednak upewnić, że content naprawdę jest dostępny za pomocą protokołu HTTPS (w przypadku materiałów z witryn zewnętrznych).
- Dla zewnętrznych skryptów należy również używać względnych adresów URL. Na przykład dla biblioteki jQuery, zamiast kodu:

  <script src="http://ajax.googleapis.com/ajax/libs/jquery/1.11.3/jquery.min.js"></script>

  trzeba użyć:

  <script src="//ajax.googleapis.com/ajax/libs/jquery/1.11.3/jquery.min.js"></script>

- Kolejnym krokiem jest instalacja certyfikatu SSL - Większość firm hostingowych oferuje możliwość szybkiego zainstalowania certyfikatu za pomocą panelu sterowania. Sam serwer musi obsługiwać protokół SSL. Certyfikat nie jest przypisany do IP lub hostingu, więc można go zainstalować na dowolnie wybranym hostingu. Jeśli bieżący nie obsługuje SSL, trzeba przejść do innego. Po udanej konfiguracji certyfikatu na serwerze, można

zająć się konfiguracją witryny. Należy skonfigurować przekierowania dla wszystkich adresów w obrębie witryny, tak aby roboty sieciowe oraz użytkownicy zostali automatycznie przeniesieni na wersję serwisu z protokołem HTTPS.

Jednym z kolejnych problemów było umożliwienie użytkownikom na dzieleniem się ekranem. Żeby rozwiązać ten problem stworzyliśmy rozszerzenie. Zadaniem rozszerzenia jest umożliwienie na przechwytywanie ekranu użytkownika, ponieważ domyślnie opcja ta jest zablokowana w takich przeglądarkach jak np. Chrome czy Opera. Na rozszerzenie składają się dwa pliki manifest.json oraz extension.js. W pliku manifest.json umieszczamy informacje jaki skrypt ma być wykonywany, ustalamy domeny w których rozszerzenie ma działać oraz do jakich zasobów daje dostęp. Natomiast w pliku extension.js umieszczamy kod który ma działać w tle w naszym przypadku jest to listener który zwraca id strumienia. Wszystko jest to możliwe dzięki API chrome.runtime. Kolejnym krokiem jest załadowanie rozszerzenia do naszej przeglądarki w tym celu należy włączyć widok chrome://extensions/ poprzez wpisanie w wyszukiwarkę. Po czym trzeba włączyć tryb programisty i wybrać opcję “załaduj rozpakowane” i wskazać miejsce na dysku gdzie znajduje się wcześniej przygotowane przez nas rozszerzenie. Gdy już nam się udało przechwycić obraz ekranu nastąpił kolejny problem jak odświeżyć obraz u osób z którymi prowadzimy konwersację. W takim przypadku należy usunąć stary strumień z naszego połączenia następnie dodać nowy strumień z widokiem naszego ekranu i wysłać nową ofertę w celu renegocjacji połączenia WebRTC.

Listing 1 Kod pliku extension.js

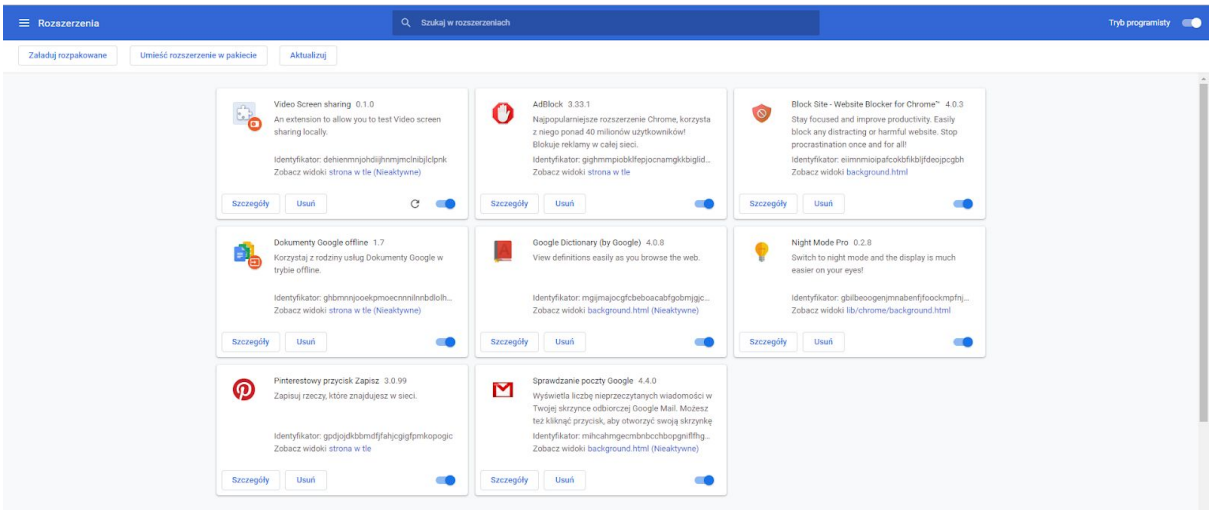
```
chrome.runtime.onMessageExternal.addListener(
  (message, sender, sendResponse) => {
    if (message == 'version') {
      sendResponse({
        type: 'success',
        version: '0.1.0'
      });
      return true;
    }
    const sources = message.sources;
    const tab = sender.tab;
    chrome.desktopCapture.chooseDesktopMedia(sources,
tab, streamId => {
      if (!streamId) {
        sendResponse({
          type: 'error',
```

```
            message: 'Failed to get stream ID'
          });
        } else {
          sendResponse({
            type: 'success',
            streamId: streamId
          });
        }
      });
      return true;
    }
);
```

7. Instrukcja użycia

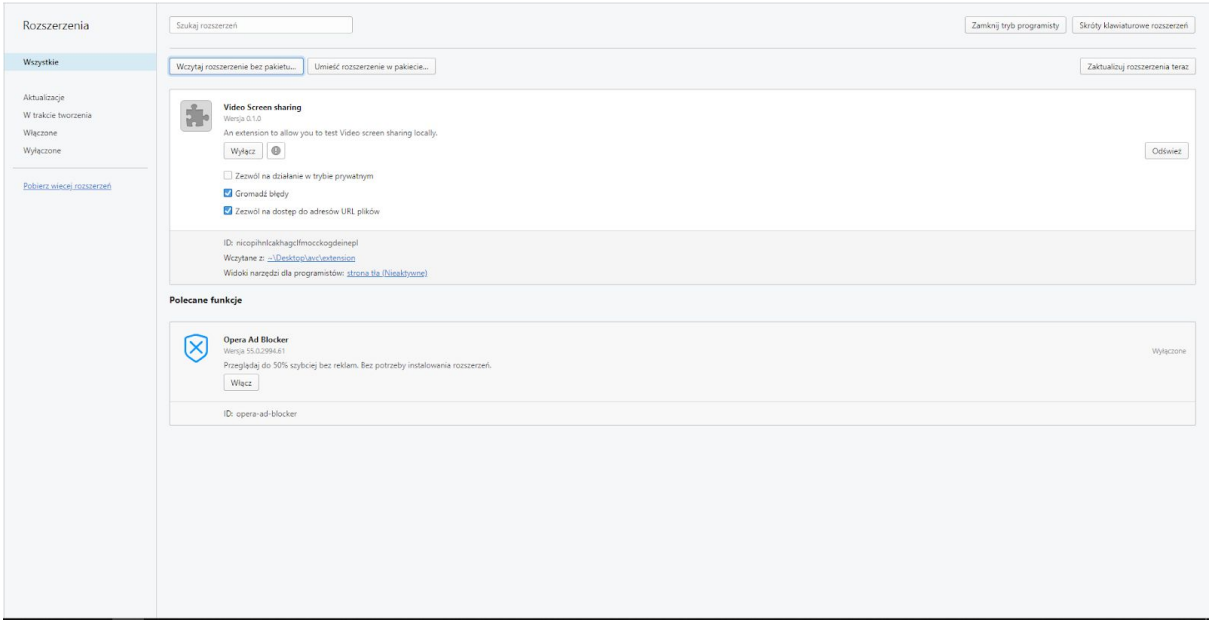
1. Instalacja rozszerzenia

Jeśli korzystasz z przeglądarki internetowej Mozilla Firefox możesz pominąć ten podpunkt. Jeśli korzystasz z Chrome lub Opera proszę podążać za instrukcjami.
Jak wcześniej było zaznaczone rozszerzenie jest konieczne by móc udostępnić swój ekran w naszej aplikacji. W tym celu należy w polu wyszukiwarki wpisać chrome://extensions.



Rys.2 Widok rozszerzeń Chrome

Kolejnym krokiem jest włączenie trybu programisty. Opcja ta jest dostępna w prawym górnym rogu, a następnie należy kliknąć “Załaduj rozpakowane” i wybrać miejsce na dysku, gdzie znajduję się rozszerzenie. Na koniec należy włączyć rozszerzenie i wpisać identyfikator rozszerzenia w kodzie w celu umożliwienia dostępu do udostępniania ekranu. Innym sposobem jest wydanie rozszerzenia w Google Store, wtedy użytkownik jedynie instaluje aplikację ze sklepu i może korzystać z naszej aplikacji, ponieważ gdy rozszerzenie jest wydane publicznie to posiada stały identyfikator. My jednak skorzystaliśmy z pierwszego sposobu i nie opublikowaliśmy rozszerzenia.
W przypadku przeglądarki Opera jest podobnie należy przejść do opera://extensions. Następnie włączyć tryb programisty, kliknąć przycisk “Wczytaj rozszerzenie bez pakietu” i wybrać lokalizację z dysku gdzie znajduje się nasze rozszerzenie.

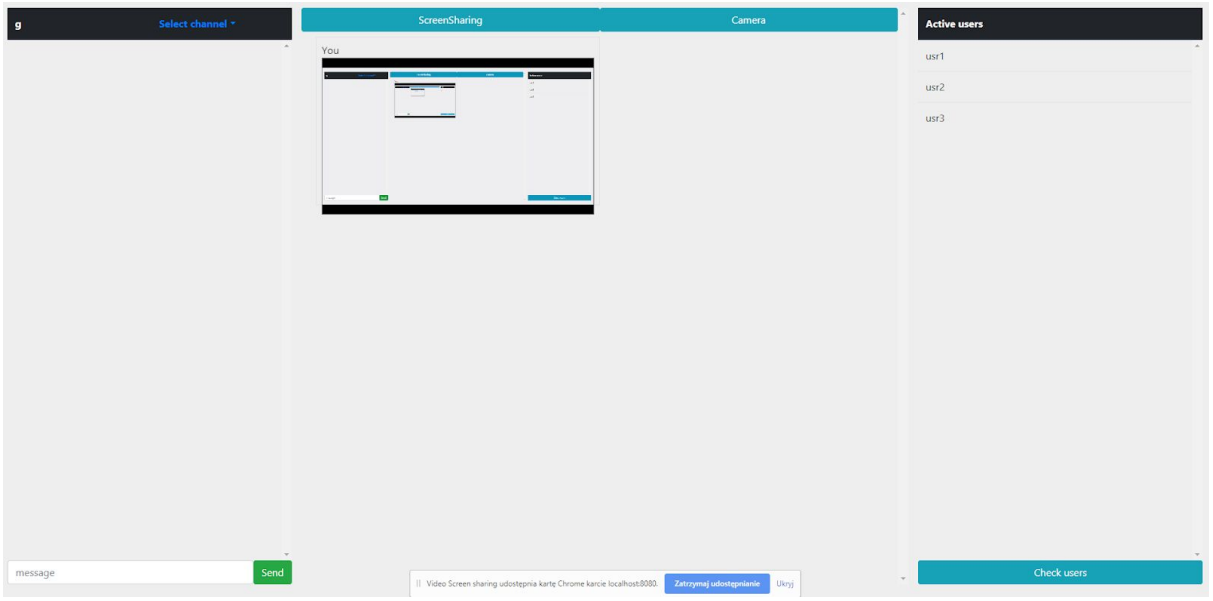


Rys. 3 Widok rozszerzeń Opera

2. Instrukcja użycia aplikacji

Rys.4 Ekran Logowania

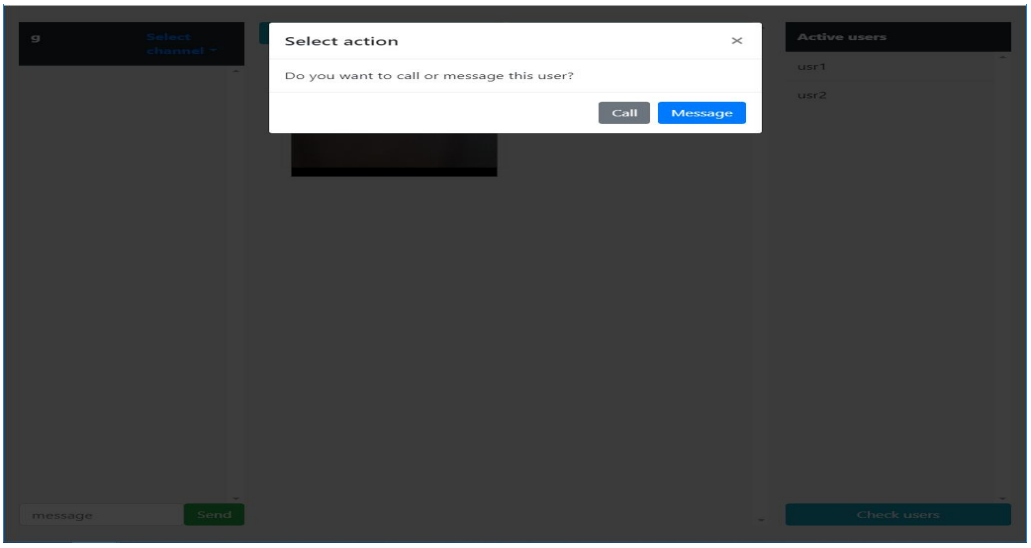
Aby zalogować się, potrzeba jedynie wpisać nazwę użytkownika składającą się z 3-20 znaków alfanumerycznych i wcisnąć klawisz “Sign In”. Dodatkowo dwóch użytkowników nie może posiadać takiej samej nazwy.



Rys. 5 Interfejs aplikacji

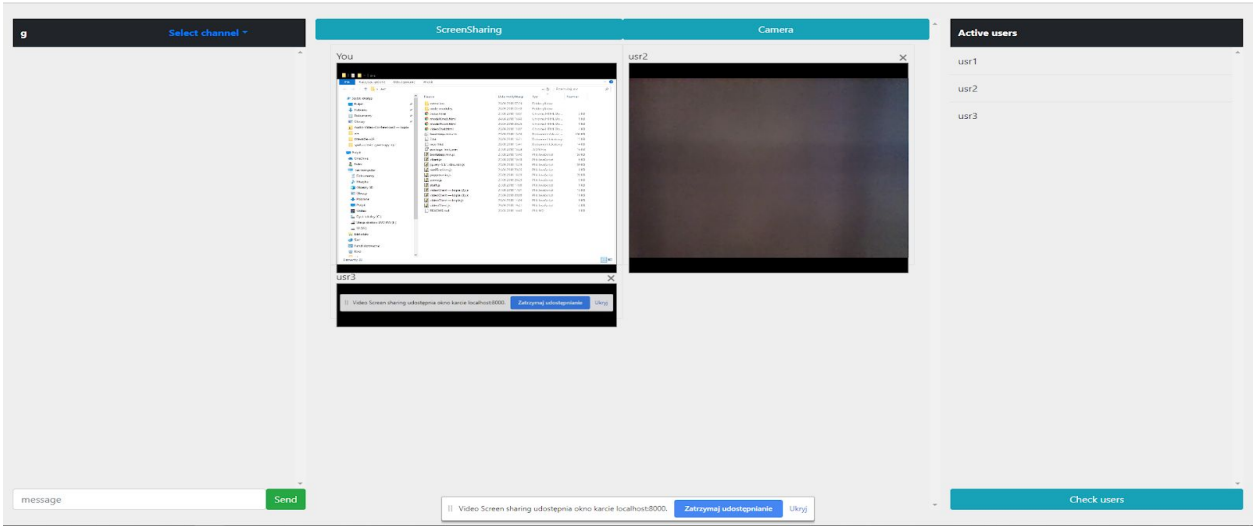
Po lewej stronie widoczny jest czat. Można dzięki niemu wysłać wiadomość do wszystkich użytkowników, lub wiadomość prywatną do jednego. Kanał na jakim chcemy pisać wybieramy klikając przycisk “Select channel”. Domyślnie wybrany jest kanał globalny “g”.

Po prawej stronie znajduje się lista aktualnie dostępnych użytkowników. Poprzez kliknięcie na jednego z nich, pojawia się pytanie czy chcemy rozpocząć połączenie czy też wysłać do niego wiadomość(Rys. 6).



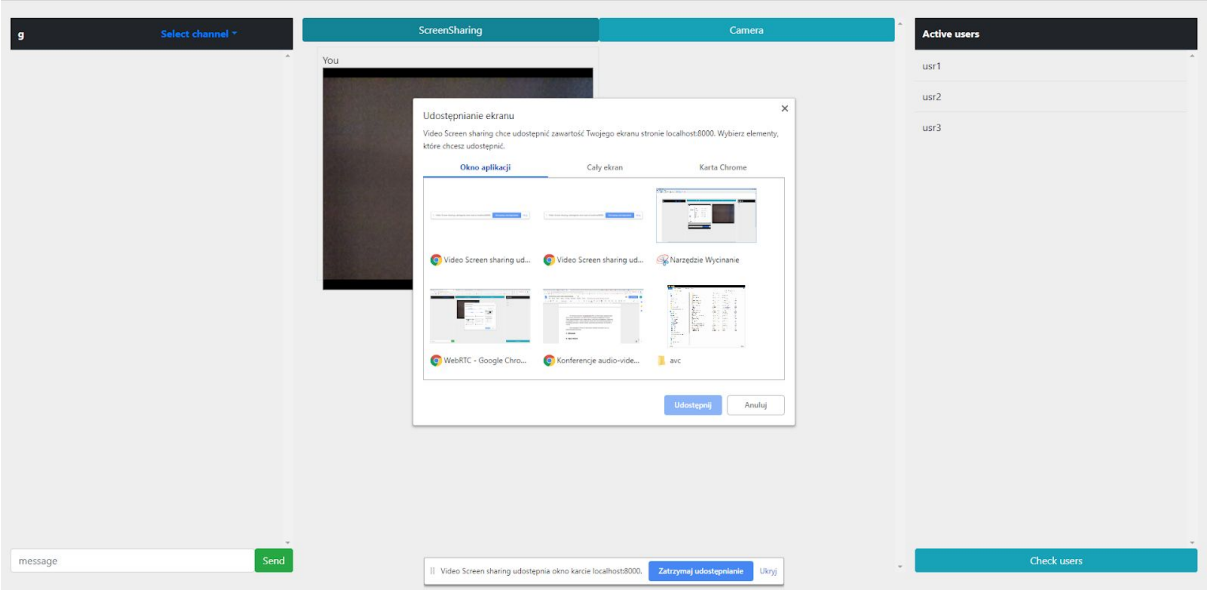
Rys. 6 Wybór akcji po kliknięciu na użytkownika

W centrum wyświetlają się strumienie wideo od użytkowników, z którymi uda nam się połączyć. Można wybrać które okno ustawić na pełen ekran i ponadto każde połączenie można wyciszyć.

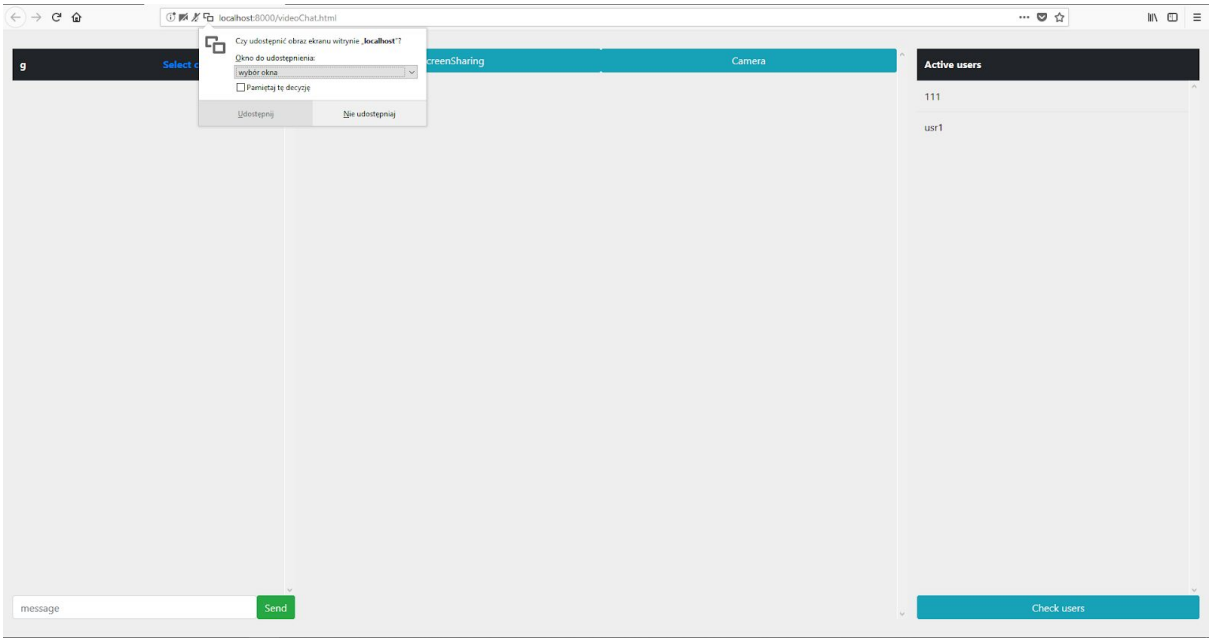


Rys. 7 Widok z aktywnymi rozmowami

Po kliknięciu przycisku “ScreenSharing”(Rys. 8) otrzymujemy zapytanie jakie okno chcemy przechwycić. Do wyboru jest w przypadku przeglądarek Chrome i Opera: przechwytywanie okna, całego ekranu, oraz karty w przeglądarce. Natomiast w przeglądarce Firefox można wyłącznie wybrać które okno przechwytywać(Rys. 9). Po kliknięciu przycisku “Camera” stream z powrotem jest zmieniany na strumień z kamery.

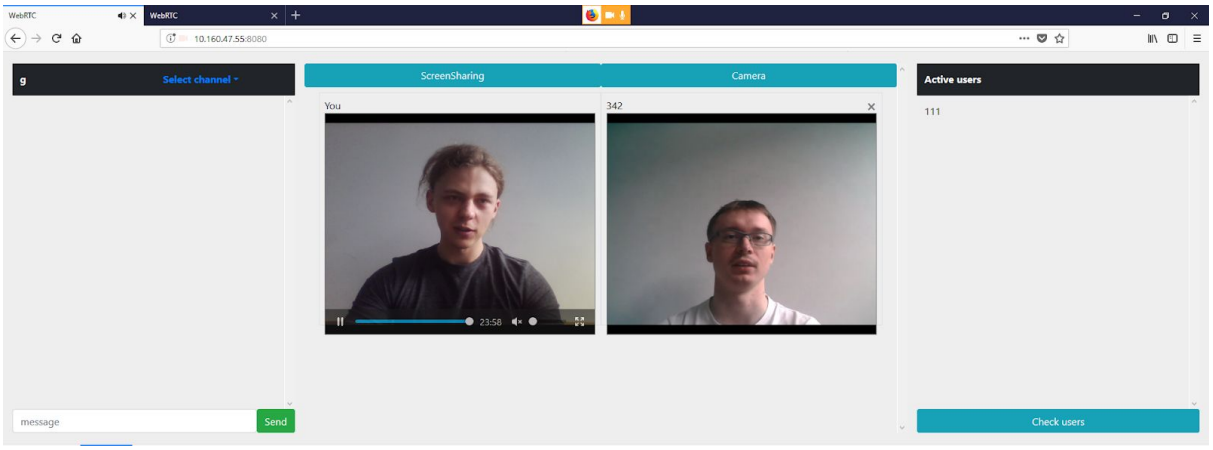


Rys. 8 Wybór elementu do udostępniania dla Chrome



Rys. 9 Wybór okna do udostępniania dla Firefox

Dodatkowo dla przeglądarki Chrome na dole ekranu widnieje komunikat o tym, że przechwytywany jest obraz.



Rys. 10 Prezentacja poprawnego działania aplikacji

Rysunek 10 przedstawia audio-wideo konferencję prowadzoną przez nas. Serwer znajduję się na jednym z naszych komputerów. Aplikacja działa wyłącznie w sieci LAN.

8. **Wnioski**

Dzięki wykonaniu projektu w ramach przedmiotu Podstawy Teleinformatyki poznaliśmy takie technologie i frameworki jak: HTML5, Node.js, JavaScript, Bootstrap, jQuery czy WebRTC. Ponadto dowiedzieliśmy się w jaki sposób działają konferencje audio-video i sami zaimplementowaliśmy przykładową aplikację. Dodatkowo stworzyliśmy rozszerzenie do Chrome'a by móc udostępniać swój ekran.

Możliwymi kierunkami rozwoju naszego projektu jest dodanie rozszerzenia do Google Store, by ułatwić korzystanie z naszego produktu. Poza tym można dodać kilka nowych funkcjonalności np. możliwość ustawiania pozycji video przez użytkownika, czy usprawnić interfejs, żeby był bardziej intuicyjny albo dodać powiadomienia gdy otrzymamy wiadomość od innego użytkownika. Dodatkowo można umożliwić korzystanie z aplikacji na urządzeniach mobilnych, które posiadają system Android i iOS.

9. Bibliografia

https://developer.mozilla.org/en-US/docs/Web/API/WebRTC_API
http://devdocs.io/javascript/
http://getbootstrap.com/
https://api.jquery.com/
https://www.tutorialspoint.com/webrtc/

10. Spis treści