Rancang Bangun Sistem Informasi Telekonferensi Menggunakan Teknologi WebRTC

Faisal Indrianto[1*], Nicky Yuda Apriyanto[2], Moh Noor Al-Azam[3], Cahyo Darujati[4]
[1,2,3,4] Sistem Informasi, Fakultas Teknik & Ilmu Komputer, Universitas Narotama
[1]faisalindrianto.18@fik.narotama.ac.id, [2]nicky.18@fik.narotama.ac.id, [3]noor.azam@narotama.ac.id,
[4]cahyo.darujati@narotama.ac.id
*Penulis Korespondensi

ABSTRAK

Telekonferensi merupakan kegiatan komunikasi atau pertemuan oleh beberapa orang yang dilakukan melalui media digital. *Web Real-Time Communication* (WebRTC) adalah teknologi *open-source* yang memungkinkan pengguna melakukan pertukaran informasi dan komunikasi secara langsung / *real-time* dan *peer-to-peer* antar pengguna melalui *web browser* untuk mendukung kegiatan telekonferensi. Sistem informasi dibangun menggunakan *framework* JavaScript Vue.js dan Node.js, serta memanfaatkan Firebase Firestore sebagai *database* dan *signaling server*. Pengembangan sistem menggunakan metode *Rapid Application Development* (RAD) dan metode pengujian fungsionalitas *black box*. Sistem memiliki fitur untuk melakukan komunikasi audio dan video, mengirimkan pesan, mengirimkan dokumen, serta membuat jadwal konferensi. Dari hasil pengujian fungsionalitas *black box* yang dilakukan, sistem dapat diimplementasikan dengan baik dan bisa digunakan karena semua fungsionalitas dapat berjalan dengan normal.

Kata kunci: WebRTC, sistem informasi, telekonferensi, JavaScript.

ABSTRACT

Teleconferencing is a communication activity or meeting by several people through digital media. Web Real-Time Communication (WebRTC) is an open-source technology that allows users to exchange real-time and peer-to-peer information and communication between users via a web browser to support teleconferencing activities. The information system is built using JavaScript frameworks such as Vue.js and Node.js and utilizes Firebase Firestore as a database and signaling server. The system development uses the Rapid Application Development (RAD) method and the black box functionality testing method. The system has features to perform audio and video communication, send messages, send documents, and schedule conferences. From the results of testing the black box functionality, the system can be implemented properly and can be used because all functionalities can run normally.

Keywords: *WebRTC, information system, teleconference, JavaScript*.

1. PENDAHULUAN

Di era industri yang begitu pesat dengan teknologi, kerja sama tim terutama dalam berkomunikasi merupakan hal yang harus dilakukan untuk meraih sebuah tujuan yang diinginkan perusahaan. Komunikasi ini harus dilakukan dalam kondisi apapun, baik secara konvensional tatap muka ataupun seperti keadaan pandemi saat ini, yang mengharuskan segala kegiatan berjalan secara daring atau *online*. Ada beberapa media dan teknologi yang dapat digunakan untuk komunikasi daring, salah satunya adalah *teleconference* [1]. *Teleconference* atau telekonferensi merupakan konferensi atau pertemuan dengan beberapa orang yang dilakukan secara jarak jauh melalui media digital [2]. Beberapa aplikasi *video conference* yang sudah ada pada saat ini memanfaatkan WebRTC untuk membangun sarana komunikasi dan pertemuan daring.

Web Real-Time Communication (WebRTC) adalah teknologi *open-source* yang memungkinkan pengguna melakukan pertukaran informasi dan komunikasi secara langsung / *real-time* melalui *web browser* tanpa mengharuskan pengguna melakukan instalasi perangkat lunak tambahan [3]. WebRTC menggunakan *Application Programming Interface* (API) berbasis JavaScript yang dapat memanfaatkan kemampuan *web browser modern* untuk mengakses berbagai kapabilitas perangkat pengguna, misalnya akses audio, video, dan *file sharing* [4]. WebRTC berkomunikasi secara *peer-to-peer* yaitu komunikasi dua arah antar perangkat pengguna. Untuk melakukan komunikasi tersebut, WebRTC membutuhkan *signaling server* untuk membangun, menyambungkan dan memutuskan koneksi WebRTC. Mekanisme penyambungan antar *peer* ini dinamakan proses *offer/answer* [5].

Penelitian ini bertujuan untuk mengetahui apa saja kapabilitas teknologi WebRTC dan pengimplementasian teknologi tersebut untuk membangun sebuah sistem informasi telekonferensi yang dapat dimanfaatkan sebagai sarana komunikasi dan kolaborasi bagi suatu tim atau perusahaan. Sistem informasi telekonferensi ini dibangun menggunakan *framework front* end JavaScript Vue.js dan *framework back-end* JavaScript Node.js yang berbasis

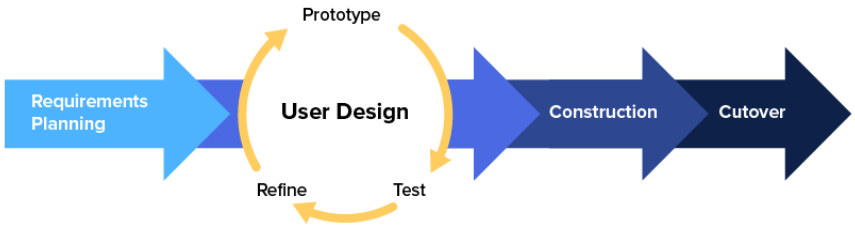
web. Vue.js adalah *framework* JavaScript yang digunakan untuk membangun tampilan antarmuka sebuah website secara progresif dan berbasis komponen dari sisi *client* [6]. Node.js adalah sebuah *runtime environement* yang memungkinkan kita membuat sebuah aplikasi web dan menjalankan kode JavaScript dari sisi *server* [7]. Sistem menggunakan *database* berbasis NoSQL yaitu Firebase Firestore. NoSQL *database* adalah sebuah model basis data yang tidak memiliki relasi dan memiliki skema yang fleksibel [8]. Firebase Firestore adalah salah satu komponen basis data dari Firebase, yaitu produk *Backend-as-a-Service* (BaaS) dari Google. Firebase Firestore memiliki struktur yang berbasis *document stores*, dimana kumpulan *key* dan *value* disimpan di dalam sebuah dokumen, lalu dokumen tersebut tersimpan di dalam kumpulan dokumen lainnya yang disebut *collection* [9].

Penelitian ini menggunakan metode pengembangan *Rapid Application Development* (RAD) yang merupakan salah satu model dari *System Development Life Cycle* (SDLC), yaitu metode yang menekankan pada proses pembuatan aplikasi berdasarkan pembuatan *prototype*, iterasi, dan *feedback* yang berulang-ulang sehingga tercipta sebuah pengembangan sistem yang cepat dan menghasilkan sebuah sistem yang stabil [10]. Kemudian metode pengujian yang digunakan untuk sistem yang dibuat adalah metode *Black Box Testing*, yaitu metode pengujian sistem yang menekankan pada detail sistem informasi dimulai dari tampilan, fungsionalitas sistem, dan kesesuaian alur fungsi dan bisnis proses dari sisi pengguna [11].

2. METODOLOGI PENELITIAN

Metode mengumpulkan data yang dibutuhkan dalam penelitian ini adalah sebagai berikut:

1. Studi Pustaka
   Studi kepusatakaan digunakan sebagai pemahaman teori-teori literatur yang berhubungan dengan penelitian yang dilakukan. Studi pustaka diperoleh dari berbagai jurnal yang relevan, penelitian terdahulu, buku, dokumentasi, dan artikel yang ada di internet.
2. Studi Lapangan
   Studi lapangan dilakukan untuk mengobservasi serta mengevaluasi permasalahan yang harus diselesaikan sehingga mendapatkan data yang dibutuhkan berdasarkan kebutuhan di lapangan yang berhubungan dengan penelitian yang akan dilakukan.



Gambar 1. ***Model Rapid Application Development (RAD)***

Metode pengembangan perangkat lunak yang digunakan adalah *Rapid Application Development* (RAD), yaitu model proses pengembangan perangkat lunak sekuensial linier yang menekankan pada siklus pengembangan yang relatif singkat. RAD sangat cocok digunakan untuk mengembangkan sistem informasi secara cepat karena siklus pengembangan lebih pendek, lebih fleksibel, dan menekankan keterlibatan pengguna [12]. Tahapan dari metode RAD adalah sebagai berikut:

1. *Requirements Planning*
   Peneliti mengidentifikasi kebutuhan seperti daftar fitur dan kapabilitas sistem yang diperlukan untuk memecahkan masalah yang dapat diselesaikan oleh aplikasi yang dibuat.
2. *User Design*
   Peneliti membuat rancangan dan desain sistem informasi yang akan dibuat sesuai dengan kebutuhan dari *requirements planning* dan data yang diperoleh menjadi sebuah desain sistem. Desain sistem yang digunakan dalam penelitian adalah *use case diagram, sequence diagram,* dan *class diagram.*
3. *Construction*
   Tahapan *contruction* merupakan tahapan membuat sistem informasi sesuai dengan kebutuhan dan desain yang sudah dibuat. Peneliti menyusun kode program atau yang biasa disebut *coding*, untuk mengubah desain yang sudah dibuat menjadi sebuah sistem informasi yang dapat digunakan.
4. *Cutover*
   Sistem informasi yang telah selesai dibuat akan dilakukan pengujian menyeluruh untuk mengurangi risiko cacat sistem. Metode pengujian yang akan digunakan adalah metode *Black Box Testing.*

Black Box Testing adalah metode pengujian sistem yang menekankan pada detail sistem informasi dimulai dari tampilan, fungsionalitas sistem, dan kesesuaian alur fungsi dan bisnis proses dari sisi pengguna. Berbeda dengan *White Box Testing*, yang menekankan pada pengujian internal dari sisi kode aplikasi yang menyeluruh,

Black Box Testing lebih condong pada perspektif pengguna, tanpa harus mengetahui isi *source code* dari sistem informasi, sehingga pengujian dapat dilakukan dengan cepat, sederhana, dan fleksibel. Pengujian dilakukan dengan memberikan *input* atau masukan kemudian melihat apakah *output* atau hasil keluaran sesuai dengan hasil yang diharapkan [11].

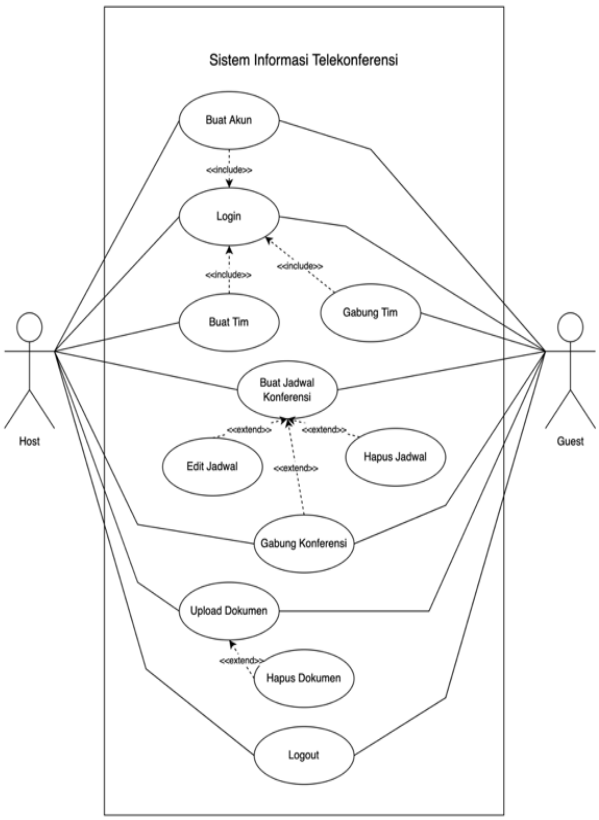
3. HASIL DAN PEMBAHASAN

3.1 Analisis Kebutuhan

Dalam tahap ini, peneliti melakukan analisa apa saja kebutuhan dan kapabilitas dari sistem informasi yang akan dibuat berdasarkan data yang dikumpulkan. Ada beberapa kebutuhan akan fitur-fitur yang harus dibuat untuk memecahkan masalah yang ditemukan. Berdasarkan hasil observasi peneliti, sistem nantinya akan terbagi menjadi dua jenis, yaitu sistem untuk telekonferensi personal dan telekonferensi untuk tim. Adapun fitur-fitur yang nantinya tersedia pada sistem infomasi telekonferensi adalah sebagai berikut:

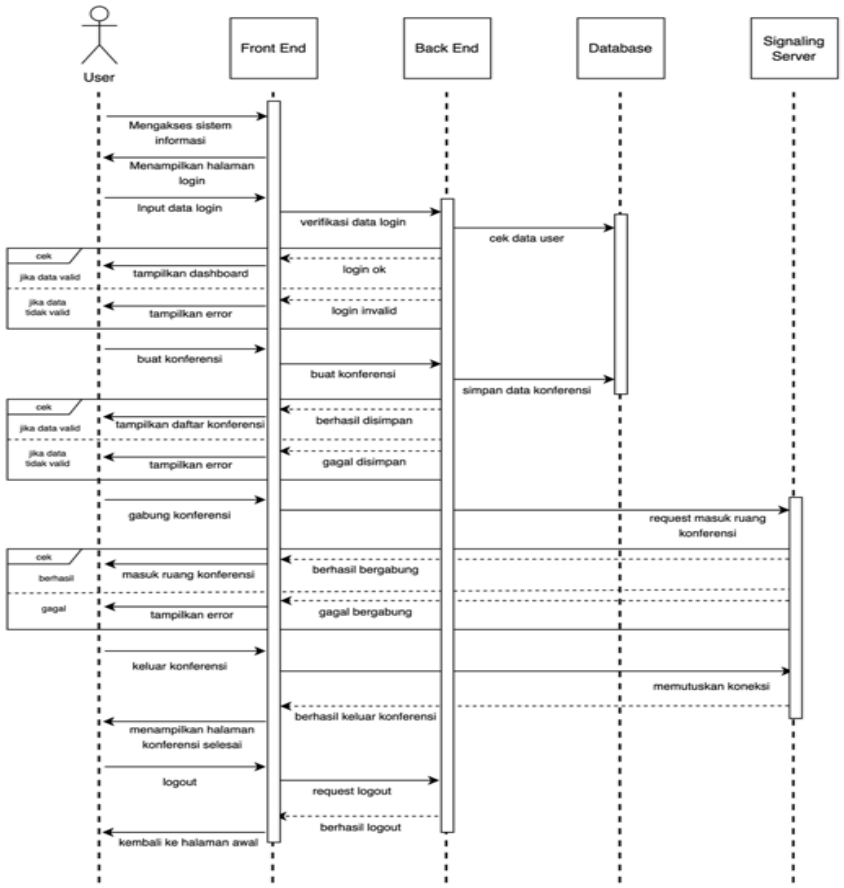
1. Manajemen Data Pribadi
2. Manajemen Data Tim
3. Modul Telekonferensi

Unified Modeling Language (UML) digunakan untuk memetakan kebutuhan sistem lebih lanjut. UML adalah sebuah permodelan visual dari perancangan sebuah sistem yang bertujuan untuk membangun, memvisualisasikan, dan mendokumentasikan sebuah perangkat lunak yang berbasis objek [13]. Model UML yang dibuat terdiri dari *use case diagram*, *sequence diagram,* dan *class diagram.* Adapun masing masing diagram adalah sebagai berikut:



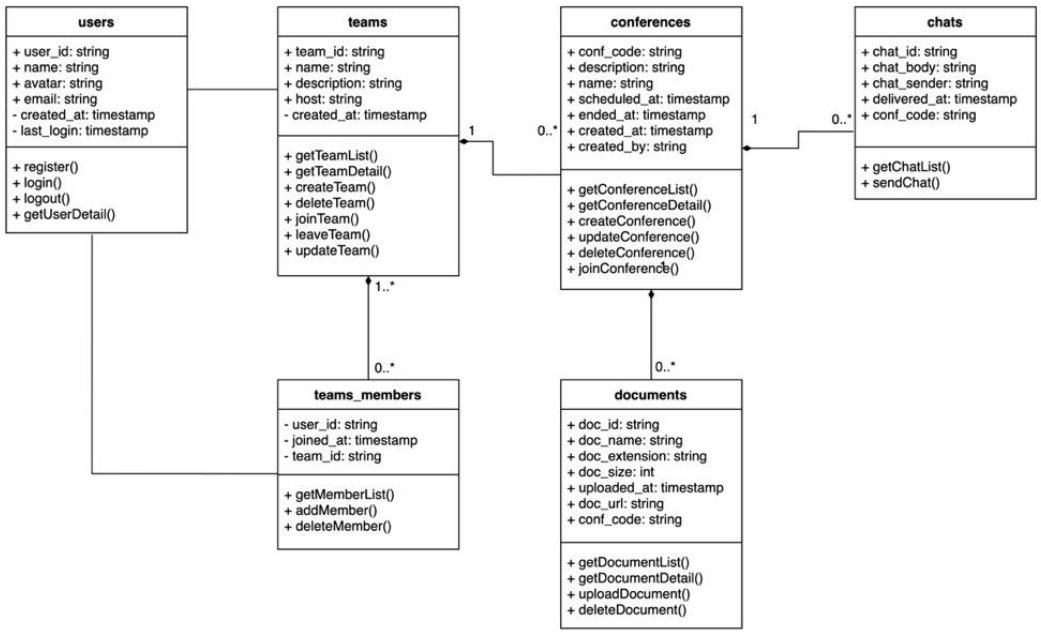
Gambar 2. *use case diagram* sistem

Use case diagram menggambarkan interaksi antara pengguna dengan sistem. Gambar 2 merupakan *use case diagram* sistem yang terdiri dari dua aktor, yaitu aktor *Host* dan *Guest* beserta fitur apa saja yang dapat digunakan pengguna di dalam sistem informasi telekonferensi. Ada tiga modul utama dari sistem, yakni manajemen akun dan tim, manajemen konferensi, dan manajemen dokumen. Pengguna dapat membuat akun dan *login* berdasarkan akun yang telah dibuat, lalu memilih untuk bergabung atau membuat tim. Setelah itu pengguna dapat membuat jadwal konferensi, lalu mengubah atau menghapus jadwal tersebut. Terakhir pengguna dapat mengunggah dokumen dan menghapus dokumen.



Gambar 3. *sequence diagram* sistem

Sequence diagram adalah bentuk diagram yang dihasilkan dari perkembangan *use case diagram*. *Sequence diagram* menggambarkan interaksi antar objek sistem berdasarkan urutan waktu. Gambar 2 menjelaskan alur penggunaan sistem informasi terkait interaksi dengan *front end*, *back end*, *database* dan *signaling server*, serta bagaimana respon etika etika terjadi *error* pada salah satu alur.



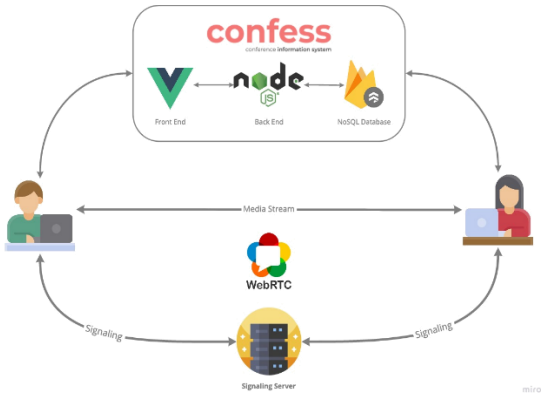
Gambar 4. *class diagram* sistem

Gambar 4 diatas *merupakan class diagram* sistem dimana *class diagram* merupakan bentuk diagram yang menggambarkan struktur sistem berdasarkan *class*, atribut data, metode / fungsi, dan hubungan antar objek. *Class*

terdiri dari pengguna, tim, konferensi, *chat*, member tim, dan dokumen. Di setiap *class* juga dijelaskan ada atribut data apa saja dan metode atau fungsi yang tersedia.

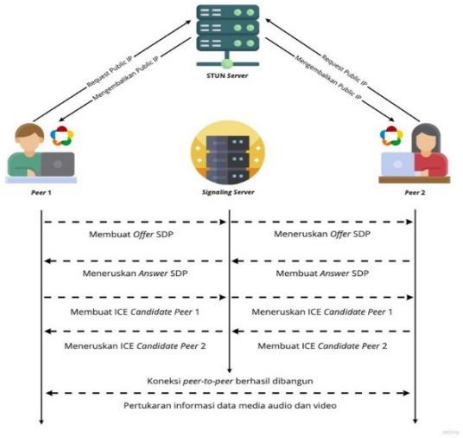
3.2 Pengembangan

Pada tahap pengembangan, peneliti melakukan *coding* sistem informasi sesuai dengan analisa kebutuhan yang sudah disusun menjadi sebuah sistem informasi yang dapat digunakan. Ada empat komponen penting dari sistem informasi yang harus dibuat dan dikembangkan, yaitu *signaling server*, *database*, *back end* dan tampilan *front end*. Semua komponen ini harus dapat berjalan dengan normal secara terpadu dan selaras agar dapat memberikan pengalaman pengguna yang baik. Sistem bekerja dengan memanfaatkan berbagai teknologi dan *framework* untuk dapat berjalan dengan maksimal. Dalam pemanfaatan nya, sistem terbagi menjadi dua bagian, yang pertama bagian sistem informasi itu sendiri, dan sistem untuk mengatur koneksi WebRTC. Di sisi sistem informasi, pengguna dapat melakukan berbagai manajemen data seperti pada umumnya dalam sebuah sistem informasi, misalnya manajemen akun, manajemen jadwal, manajemen tim, dan manajemen sumber daya. Sistem terbagi menjadi dua jenis, yaitu sistem informasi versi personal dan versi tim. Pada versi personal, pengguna dapat melakukan konferensi secara langsung tanpa harus *log in* dan membuat jadwal terlebih dahulu. Lalu pada versi tim, fitur tambahan yang tersedia adalah manajemen tim dan jadwal konferensi.



Gambar 5. ilustrasi cara kerja sistem

Gambar 5 menunjukkan bahwa sistem akan membangun koneksi WebRTC antar pengguna melalui *signaling server* agar pengguna dapat melakukan komunikasi dua arah secara *peer-to-peer*. *Signaling server* juga digunakan untuk memberi tahu antar *peer* jika ada pengguna baru yang ingin bergabung atau keluar dari sesi konferensi. Proses agar pengguna dapat berkomunikasi secara dua arah harus melewati berbagai tahap, dimulai dari pengguna membuat ruang konferensi, lalu mendapatkan akses media audio dan video dari perangkat pengguna, hingga membangun dan memutuskan koneksi antar pengguna. Proses penyambungan antar *peer* ini disebut mekanisme *offer/answer*. Perangkat pengguna pada umumnya menggunakan sebuah *Network Address Translation* (NAT) yang tidak memiliki IP publik, sehingga dibutuhkan sebuah *server Session Traversal Utilities for* NAT (STUN). *Server* STUN membantu pengguna untuk melangkahi NAT sehingga bisa saling terhubung secara *peer-to-peer*, teknik ini disebut NAT *Traversal*. Peneliti memanfaatkan kapabilitas *real-time* dari Firebase Firestore sebagai *signaling server* untuk menyimpan data *offer* dan *answer Session Description Protocol* (SDP) yang sudah dibuat lalu *peer* yang terhubung akan menangkap data *offer* dan *answer* yang baru saja ditambahkan.



Gambar 6. Proses offer/answer

Gambar 6 dapat dipahami bahwa untuk memulai sebuah sesi, *peer* 1 terlebih dahulu membuat sebuah *offer* SDP yang diteruskan oleh *signaling server* ke *peer* 2. *Peer* 1 juga melakukan pemantauan ke *signaling server* untuk menunggu *answer* dari *peer* 2. *Peer* 2 lalu menangkap *offer* tersebut ketika sebuah dokumen SDP ditambahkan ke Firebase Firestore dan membuat *answer* SDP yang diteruskan ke *peer* 1 melalui *signaling server*. Proses *offer/answer* ini juga harus dilakukan kembali jika ada *peer* baru yang bergabung ke sesi konferensi agar dapat saling terhubung secara *peer-to-peer*.

```
const peerConnection = new RTCPeerConnection()
// kode ruang konfereosi dari inputan user
const kodeRuang = 'lemper-2121'
// referensi koleksi dokumen Firebase Firestore
const meetRef = firebase.firestore().collection('meets').doc(kodeRuang)

const createOffer = async (peerTwo, peerOne) => {
  const offer = await peerConnection.createOffer()

  // menambahkan offer SDP ke peer
  await peerConnection.setLocalDescription(offer)

  // menambahkan offer SDP dari peer 1 ke dokumen peer 2 agar bisa ditangkap peer 2
  await meetRef.collection(peerTwo)
    .doc('SDP').collection('offer')
    .doc(peerOne).set({ offer })
}

const receiveAnswer = async (peerOne, peerTwo) => {
  // memantau jika ada answer SDP dari peer 2
  meetRef.collection(peerOne).doc('SDP').collection('answer').doc(peerTwo)
    .onSnapshot(async snapshot => {
      if (snapshot.exists) {
        const data = snapshot.data()
        const rtcSessionDescription = new RTCSessionDescription(data.answer)

        // menambahkan answer SDP ke remote
        await peerConnection.setRemoteDescription(rtcSessionDescription)
      }
    })
}
```

```
const peerConnection = new RTCPeerConnection()
// kode ruang konferensi dari inputan user
const kodeRuang = 'lemper-123'
// koleksi dokumen Firebase Firestore
const meetRef = db.collection('meets').doc(kodeRuang)

const createAnswer = async (peerTwo, peerOne) => {
  const answer = await peerConnection.createOffer()

  // menambahkan answer SDP ke peer
  await peerConnection.setLocalDescription(answer)

  // menambahkan answer SDP dari peer 2 ke dokumen peer 1 agar bisa diterima peer 1
  await meetRef.collection(peerOne)
    .doc('SDP').collection('answer')
    .doc(peerTwo).set({ answer })
}

const receiveOffer = async (peerOne, peerTwo) => {
  // memantau jika ada offer SDP dari peer 1
  meetRef.collection(peerTwo).doc('SDP').collection('offer').doc(peerOne)
    .onSnapshot(async snapshot => {
      if (snapshot.exists) {
        const data = snapshot.data()
        const rtcSessionDescription = new RTCSessionDescription(data.offer)

        // menambahkan offer SDP ke remote
        await peerConnection.setRemoteDescription(rtcSessionDescription)
        createAnswer()
      }
    })
}
```

Gambar 7 script untuk membuat dan menerima *offer/answer*

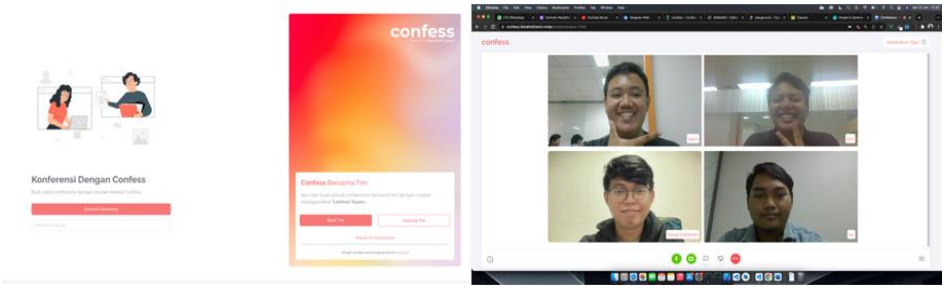
Gambar 7 merupakan *script* untuk membuat dan menerima *offer/answer*. Setelah proses *offer/answer* dilakukan, antar *peer* kemudian melakukan pengumpulan *Interactive Connectivity Estabilishment* (ICE) *Candidates* dari lawan *peer* yang merupakan kembalian dari STUN *server* yang berisi IP dan *Port* yang digunakan agar antar *peer* dapat saling terkoneksi dan berkomunikasi. Setelah proses *offer/answer* dilakukan, maka koneksi berhasil dibangun lalu *output* audio dan video dari *peer* lain dapat ditampilkan dan dua pengguna sudah dapat saling berkomunikasi pada sistem informasi melalui *web browser* yang ditampilkan melalui *front end*.

```
const peerConnection = new RTCPeerConnection()
// kode ruang konferensi dari inputan user
const kodeRuang = 'lemper-2121'
// referensi koleksi dokumen Firebase Firestore
const meetRef = firebase.firestore().collection('meets').doc(kodeRuang)

// mengirimkan ice candidates ke signaling server
const signalICECandidates = (lawanPeer, peerSaya) => {
  const callerCandidatesCollection = meetRef.collection(peerSaya)

  // memantau jika ice candidate berhasil dibuat
  peerConnection.addEventListener('icecandidate', event => {
    if (event.candidate) {
      // menyimpan ice candidate ke database
      callerCandidatesCollection.add(event.candidate.toJSON()).then(doc => {
        doc.update({
          id: lawanPeer,
        })
      })
    }
  })
}

// menerima ice candidates dari signaling server
const receiveICECandidates = (lawanPeer, peerSaya) => {
  // memantau perubahan jika ada ice candidates yang baru ditambah
  meetRef.collection(lawanPeer).where('id', '==', peerSaya).onSnapshot(snapshot => {
    snapshot.docChanges().forEach(async change => {
      if (change.type === 'added' && change.doc.id !== 'SDP') {
        const data = change.doc.data()
        // menambahkan ice candidate ke remote
        await peerConnection.addIceCandidate(new RTCIceCandidate(data))
      }
    })
  })
}
```

Gambar 8 script mengirim dan menerima *ice candidates*

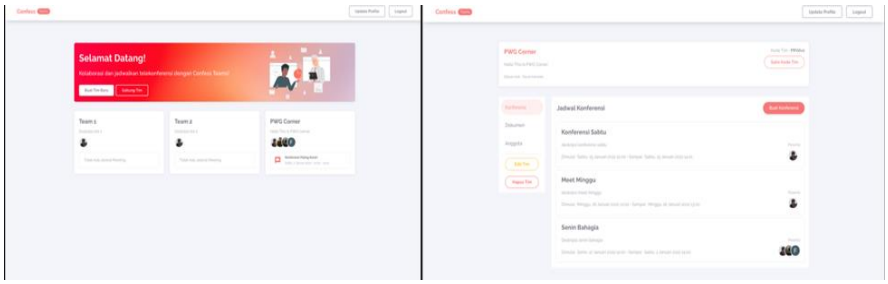
Gambar 8 merupakan script untuk mengirim dan menerima *ice candidates*. *Front end* merupakan bagian penting sistem yang dapat mempengaruhi pengalaman pengguna dari sisi tampilan visual dan interaksi pengguna. Sebuah *front end* tidak hanya harus memberikan tampilan visual dan *layout* yang menarik, namun harus memudahkan pengguna dalam menyelesaikan sebuah pekerjaan atau masalah, sehingga dapat memberikan pengalaman pengguna yang baik. *Front end* sistem informasi pada penelitian ini dibangun menggunakan Vue.js,

yaitu *framework* berbasis JavaScript yang dapat digunakan untuk membuat sebuah aplikasi yang progresif dan dinamis. Tampilan sistem informasi yang sudah dibangun adalah sebagai berikut:



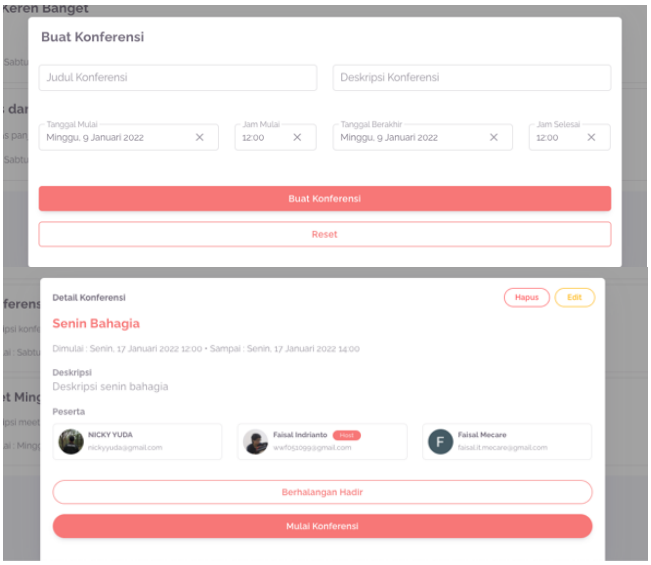
Gambar 9 Tampilan Halaman Awal dan Halaman Konferensi

Halaman awal terbagi menjadi dua kolom seperti yang terlihat pada gambar 9. Kolom kiri berfungsi untuk membuat konferensi versi *personal* dan di kolom kanan untuk mengakses sistem informasi versi tim, yang terdiri dari tombol buat tim, gabung tim, masuk ke *dashboard*, serta indikator jika sudah *log in* dan terdapat sebuah tautan untuk *log out.* Halaman konferensi terdiri dari tampilan video dari peserta sesi konferensi dan di bagian bawah terdapat aksi aksi kontrol dari proses berjalannya konferensi. Terdapat tombol untuk keluar dari konferensi, mematikan dan menyalakan input suara, mematikan dan menyalakan input kamera, mengakses *chat*, membagikan dokumen, dan membagikan tampilan layar / *screen sharing*.



Gambar 10. Tampilan Halaman Daftar Tim dan Detail Tim

Gambar 10 merupakan halaman daftar tim berisi daftar tim yang tergabung oleh pengguna dan terdiri dari *header* atas yang berisi ucapan selamat datang, tombol buat tim, dan gabung tim. Lalu di bawah *header*, terdapat kartu daftar tim yang berisi nama tim, deskripsi tim, anggota tim, dan jadwal konferensi terbaru (jika ada). Pada halaman detail tim, terdapat informasi mengenai detail tim, seperti judul tim, deskripsi tim, dan kode tim. Lalu di bagian bawah terbagi menjadi dua kolom, yang di sebelah kiri terdapat menu navigasi *tab* untuk melihat daftar item terkait. Secara default, menu navigasi yang aktif adalah daftar jadwal telekonferensi. Pada bagian tersebut terdapat judul, tombol untuk menampilkan form membuat jadwal konferensi, dan daftar jadwal konferensi. Pada kartu jadwal konferensi berisi informasi mengenai judul konferensi, deskripsi konferensi, waktu konferensi, dan peserta konferensi.



Gambar 11. Komponen Form dan Detail Konferensi

Komponen *form* konferensi pada gambar 11 diatas berisi inputan untuk data judul konferensi, deskripsi konferensi, dan tanggal serta jam mulai hingga berakhir dari jadwal konferensi. Lalu di bagian bawah kolom inputan terdapat tombol untuk menyimpan data yang sudah diisi ke *database* dan tombol untuk mereset ulang *form*. Komponen detail konferensi dapat ditampilkan ketika pengguna menekan salah satu kartu konferensi. Komponen ini berisi judul konferensi, waktu konferensi, deskripsi konferensi, serta peserta konferensi. Terdapat juga tombol untuk mengubah data konferensi, hapus konferensi, serta tombol untuk bergabung atau keluar dari jadwal konferensi.

3.3 Pengujian

Hasil pengujian fungsionalitas *black box* adalah sebagai berikut:

Tabel 1. Pengujian fungsionalitas sistem

Skenario Pengujian	Masukan	Ekspektasi	Hasil
Daftar akun menggunakan Google Auth	Menekan tombol daftar.	Akun baru akan terbuat dan dapat digunakan untuk masuk ke sistem.	Valid
Log in menggunakan akun Google	Menekan tombol *log in*.	Memilih akun Google yang digunakan untuk masuk ke sistem lalu berhasil ter *log in*.	Valid
Mengubah data akun	Menginput nama dan memilih foto, lalu menekan tombol *update* pada form edit akun.	Nama dan foto akun berhasil terubah.	Valid
Log out	Menekan tombol *log out* dan menekan tombol konfirmasi.	Berhasil keluar dari sistem dan tidak terautentikasi.	Valid
Membuat konferensi	Menekan tombol "confess sekarang" pada halaman *home*.	Jika berhasil akan masuk ke halaman konferensi dan ruang konferensi bisa dibagikan.	Valid
Bergabung ke konferensi	Menginputkan kode ruang konferensi lalu menekan tombol *submit*.	Berhasil bergabung ke konferensi yang sudah dibuat menggunakan kode konferensi atau alamat tautan konferensi.	Valid
Membuat tim	Menekan tombol buat tim pada halaman *home*, meginputkan nama dan deskripsi tim pada form, lalu menekan tombol *submit*.	Pengguna akan menjadi *host* dari tim yang dibuat dan langsung diarahkan ke halaman detail tim jika berhasil.	Valid
Bergabung ke tim	Menekan tombol gabung tim, menginputkan kode tim, lalu menekan tombol *submit*.	Pengguna bergabung ke tim yang sudah dibuat pengguna lain dan akan terdaftar sebagai anggota dari tim tersebut.	Valid
Menampilkan daftar tim	Masuk ke halaman *dashboard*.	Pengguna masuk ke *dashboard* dan dapat melihat daftar tim yang sudah tergabung.	Valid
Menampilkan detail tim	Menekan salah satu kartu tim di halaman daftar tim.	Halaman detail tim akan tampil ketika pengguna menekan salah satu tim di halaman daftar tim.	Valid
Menampilkan jadwal konferensi	Masuk ke detail tim lalu menekan *tab* jadwal konferensi.	Jadwal konferensi yang sudah dibuat akan tampil di komponen daftar jadwal konferensi.	Valid
Menambah jadwal konferensi	Menekan tombol buat konferensi, menginputkan judul, deskripsi, dan waktu konferensi, lalu menekan tombol *submit*.	Jadwal konferensi akan tampil di komponen daftar jadwal konferensi jika berhasil disubmit.	Valid
Melihat detail konferensi	Menekan salah satu kartu jadwal konferensi.	Pengguna melihat detail jadwal konferensi dengan menekan salah satu jadwal konferensi.	Valid
Mengubah data konferensi	Di kartu detail konferensi, pengguna menekan tombol ubah konferensi, menginputkan data yang ingin diubah, lalu menekan tombol *submit*.	Data judul, deskripsi, dan waktu jadwal konferensi berhasil diubah.	Valid
Bergabung ke jadwal konferensi	Menekan tombol gabung konferensi.	Pengguna menekan tombol gabung konferensi dan akan terdaftar sebagai peserta konferensi.	Valid
Keluar dari jadwal konferensi	Menekan tombol berhalangan hadir.	Jika sudah bergabung, pengguna menekan tombol berhalangan hadir untuk keluar dari jadwal konferensi.	Valid
Menghapus jadwal konferensi	Menekan tombol hapus konferensi dan menekan tombol konfirmasi.	Jadwal konferensi berhasil dihapus dari daftar jadwal.	Valid

Menampilkan daftar dokumen	Masuk ke detail tim dan menekan *tab* dokumen.	Dokumen yang berhasil diunggah selama sesi konferensi akan tampil di daftar dokumen.	Valid
Melihat detail dokumen	Menekan salah satu kartu dokumen.	Pengguna menekan salah satu dokumen dan detail dokumen akan ditampilkan di komponen *document viewer*.	Valid
Menghapus dokumen	Menekan tombol hapus pada salah satu kartu dokumen.	Pengguna menghapus salah satu dokumen dan tidak akan tampil lagi di daftar dokumen.	Valid
Melihat daftar anggota tim	Masuk ke detail tim dan menekan *tab* daftar anggota.	Daftar anggota tim yang sudah tergabung akan tampil di daftar anggota.	Valid
Menambahkan anggota tim	Menekan tombol tambah anggota, menginputkan alamat email, dan menekan tombol *submit*.	*Host* menambahkan email pengguna yang sudah terdaftar di sistem dan langsung menjadi anggota tim tersebut.	Valid
Menghapus anggota tim	Menekan tombol keluarkan dari tim dan menekan tombol konfirmasi.	*Host* menghapus pengguna sehingga tidak lagi menjadi anggota tim.	Valid
Mengubah tim	Menekan tombol ubah tim, menginputkan nama dan deskripsi tim yang baru, lalu menekan tombol *submit*.	*Host* berhasil mengubah nama dan deskripsi tim.	Valid
Keluar dari tim	Menkan tombol keluar dari tim dan menekan tombol konfirmasi.	Pengguna dapat keluar dari tim dan tidak lagi terdaftar sebagai anggota tim dari tim tersebut.	Valid
Menghapus tim	Menekan tombol hapus tim dan menekan tombol konfirmasi.	*Host* menghapus tim dan tidak tampil lagi di daftar tim dari masing masing anggota tim.	Valid
Mengirimkan pesan	Masuk ke konferensi, membuka menu *chat*, menginputkan pesan, lalu menekan tombol *submit*.	Pesan berhasil diterima pengguna lain.	Valid
Menerima pesan	Membuka menu *chat*.	Pesan dari pengguna lain muncul di daftar pesan.	Valid
Membagikan tampilan layar	Menekan tombol bagikan tampilan layar lalu menekan tombol konfirmasi.	Tampilan layar pengguna akan tampil di konferensi.	Valid
Mengirimkan dokumen	Menekan tombol unggah dokumen, memilih dokumen dari perangkat pengguna, lalu menekan tombol kirim.	Dokumen yang telah dikirimkan akan muncul di daftar dokumen pada halaman konferensi dan halaman detail tim.	Valid
Melihat dokumen	Menekan salah satu kartu dokumen yang sudah dikirimkan.	Dokumen yang dikirimkan dapat dilihat di komponen *document viewer*.	Valid
Meninggalkan konferensi	Menekan tombol tinggalkan konferensi dan menekan tombol konfirmasi.	Pengguna berhasil keluar dari halaman konferensi dan pengguna tidak ada pada daftar peserta konferensi.	Valid

4. KESIMPULAN

Teknologi WebRTC dapat dimanfaatkan sebagai sarana untuk melakukan kegiatan konferensi secara *peer-to-peer* tanpa mengharuskan pengguna melakukan instalasi perangkat lunak tambahan. Kita dapat memanfaatkan kapabilitas WebRTC untuk melakukan pertukaran informasi seperti audio, video, dan *file sharing* secara langsung / *real-time* dan *peer-to-peer* untuk membangun sebuah sistem informasi telekonferensi. Sistem informasi telekonferensi yang dibuat dapat digunakan untuk menyusun jadwal pertemuan daring dan konferensi yang dapat membantu bagi tim perusahaan maupun personal untuk melakukan komunikasi jarak jauh. Berdasarkan hasil pengujian *black box* yang dilakukan, sistem telah siap diimplementasikan dan digunakan bagi personal dan perusahaan karena semua fungsionalitas dan fitur dapat berjalan dengan normal. Kelemahan dari sistem yang dibuat adalah koneksi *peer-to-peer* yang sudah dibangun terkadang mudah terputus, terutama pada jaringan pengguna yang lemah. Sebuah koneksi *peer-to-peer* memiliki kekurangan, karena secara konsep, antar *peer* harus saling terhubung satu sama lain, sehingga jika ada banyak *peer* yang terhubung, maka akan membebani penggunaan sumber daya perangkat dan jaringan pengguna. Sehingga saran bagi penelitan ini adalah penggunaan *media server* agar koneksi *peer-to-peer* yang dibangun lebih stabil dan dapat menampung lebih banyak pengguna.

DAFTAR PUSTAKA

[1] Syarifuddin, "Telematika Dakwah di Dunia Broadcasating," *Jurnal Dakwah Tabligh*, vol. 13, no. 2, pp. 213–225, 2012.

[2] M. Katsika, I. Koutroulis, C. Zotos, M. Mitroulia, and S. Armakolas, "A social approach to teleconferencing," *Inovace a technologie ve vzdělávání*, vol. 1, no. 1, pp. 117–126, 2019.

[3] D. Halder, P. Kumar, S. Bhushan, and A. M. Baswade, "fybrrStream: A WebRTC based Efficient and Scalable P2P Live Streaming Platform," in *2021 International Conference on Computer Communications and Networks (ICCCN)*, 2021, pp. 1–9. doi: 10.1109/ICCCN52240.2021.9522198.

[4] R. Y. Rahmanda, E. Sakti Pramukantoro, and W. Yahya, "Perancangan dan Implementasi Kelas Virtual FILKOM Universitas Brawijaya dengan Memanfaatkan Teknologi WebRTC (Web Real-Time

Communication)," *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer*, vol. 2, no. 7, pp. 2721–2729, 2018.
[5] W. Elleuch, "Models for multimedia conference between browsers based on WebRTC," in *International Conference on Wireless and Mobile Computing, Networking and Communications*, 2013, pp. 279–284. doi: 10.1109/WiMOB.2013.6673373.
[6] E. Hanchett and B. Listwon, *Vue. js in Action*. Simon and Schuster, 2018.
[7] J. Wexler, *Get Programming with Node. js*. Simon and Schuster, 2019.
[8] S. Baral, "The Rising NoSql Technology," *International Journal of Advanced Engineering, Management and Science (IJAEMS)*, vol. 2, no. 11, pp. 1932–1935, 2016.
[9] L. Delia and P. Pesado, "Performance Analysis in NoSQL Databases, Relational Databases and NoSQL Databases as a Service in the Cloud," in *Computer Science–CACIC 2020: 26th Argentine Congress, CACIC 2020, San Justo, Buenos Aires, Argentina, October 5–9, 2020, Revised Selected Papers*, 2021, pp. 157–170.
[10] J. R. Sagala, "Model Rapid Application Development (RAD) Dalam Pengembangan Sistem Informasi Penjadwalan Belajar Mengajar," *Jurnal Mantik Penusa*, vol. 2, no. 1, pp. 87–90, 2018.
[11] T. Snadhika Jaya, "Pengujian Aplikasi dengan Metode Blackbox Testing Boundary Value Analysis (Studi Kasus: Kantor Digital Politeknik Negeri Lampung)," *Jurnal Informatika: Jurnal Pengembangan IT (JPIT)*, vol. 03, no. 02, pp. 45–48, 2018.
[12] N. Hidayat and K. Hati, "Penerapan Metode Rapid Application Development (RAD) dalam Rancang Bangun Sistem Informasi Rapor Online (SIRALINE)," *Jurnal Sistem Informasi (JSI) STMIK Antar Bangsa*, vol. 10, no. 1, pp. 8–17, 2021.
[13] S. Lee, "Unified Modeling Language (UML) for Database Systems and Computer Applications," *International Journal of Database Theory and Application*, vol. 5, no. 1, pp. 157–164, 2012.