Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer e-ISSN: 2548-964X
Vol. 7, No. 2, Februari 2023, hlm. 854-862 http://j-ptiik.ub.ac.id

Penerapan *Websocket* pada Sistem *Live Chat* berbasis Web (Studi Kasus *Website* Kwikku.com)

Lius Alviando[1], Adhitya Bhawiyuga[2], Dany Primanita Kartikasari[3]

Program Studi Teknik Informatika, Fakultas Ilmu Komputer, Universitas Brawijaya
Email: [1]liusalviando@gmail.com, [2]bhawiyuga@ub.ac.id, [3]dany.jalin@ub.ac.id

Abstrak

Live Chating memanfaatkan teknologi berkirim pesan secara *real time*. Seiring dengan meningkatnya pengguna layanan dan belum siapnya sistem yang ada dapat mengganggu komunikasi antar pengguna di Website Kwikku.com. Metode *Long Polling* dan S*istem Relational Database* yang digunakan terlalu banyak menggunakan *resource* dari *server* sehingga perlu adanya pembaharuan sistem dengan menggunakan protokol *Websocket*. Sistem perencanaan dan implementasi yang dibuat adalah perancangan struktur sistem dan implementasi pada pengembangan sistem *Live Chat* dengan menggunakan protokol *Websocket* dan bahasa pemrograman HTML, PHP, CSS dan Javascript. Hasil pengujian sistem menggunakan metode *load testing*. Berdasarkan hasil analisa melalui yang telah dilakukan dengan *sample* 100 hingga 500 user dan 3 repetisi dapat disimpulkan bahwa protokol *Websocket* lebih unggul jika dibandingan dengan *long polling* karena menghasilkan *delay* yang lebih kecil dan memiliki rata-rata efisiensi sebesar 52,181%. Protokol *Websocket* dengan menggunakan *non-relational database* dapat menjadi solusi untuk mengatasi permasalahan tingginya *delay* pada sistem

Kata kunci: *kwikku.com,live chat,websocket,long polling, real time*

Abstract

Live Chat utilizes technology to send messages in real time. As service users increase and the existing system is not yet ready, it can disrupt communication between content creators and community members. The Long Polling method and the Relational Database System that are used use too many resources from the server, it is necessary to update the system using the Websocket protocol. The planning and implementation system created is the design of the system structure and implementation of the Live Chat system development using the Websocket protocol and the programming languages HTML, PHP, CSS and Javascript. The results of testing the system using the load testing method. Based on the results of the analysis that has been carried out with a sample of 100 to 500 users and 3 repetitions, it can be concluded that the Websocket protocol is superior when compared to long polling because it produces smaller delays and has an average efficiency of 52.181%. The Websocket protocol using a non-relational database can be a solution to overcome the problem of high delay on the system

Keywords: *kwikku.com,chat,websocket, long polling, real time*

1. PENDAHULUAN

Kwikku.com merupakan produk dari PT Kwikku Media Nusantara yang merupakan anak perusahaan dari Falcon Pictures. Layanan yang diberikan berupa *website social media* dengan fitur utama *storytelling*. Kwikku memiliki 3 jenis karya yang dapat dinikmati pengguna yaitu novel, artikel dan komik yang dapat diakses melalui web browser dan aplikasi mobile Android dan iOS. Fitur yang dapat dinikmati oleh pengguna terdiri dari Kwikku Now yaitu layanan yang dibuat bertujuan untuk memberikan berita-berita terkini dan pengguna dapat mencari informasi yang berasal dari berbagai Media Parner Official. Kwikku Novel, pengguna dapat mengunggah dan mengunduh karya novel, yang telah memiliki dengan beberapa penerbit besar di Indonesia seperti Mizan Group dan Falcon Publishing. Kwikku juga menyediakan *Webtoon* atau komik digital original secara ekslusif, dimana hasil karya *creator* komik dapat dipublikasikan pada aplikasi ini.

Kwikku memiliki fitur media sosial yang dapat mendukung para *content creator*

untuk berkomunikasi langsung dengan pembaca serta mempromosikan karya mereka. Kegiatan tersebut ditunjang melalui fitur *Group Live Chat* dan *Video Broadcasting* yang dapat digunakan juga sebagai sarana publikasi karya-karya novel baru. Fitur tersebut digunakan pula untuk *event meet & greet* antara *public figure* dan penggemar, sehingga penggemar dapat berinteraksi dengan *public figure* dan *content creator* melalui *live chat*. *Live chat* adalah interaksi atau diskusi antara seorang ahli, orang terkenal dan anggota masyarakat yang melibatkan pengirimiman pesan melalui internet pada saat yang sama. Fitur *live chat* pada *platform* Kwikku penting untuk menunjang komunikasi antara penggemar dengan *content creator* sehingga dibutuhkan respon yang cepat dan *real time* atas *request* yang dikirimkan agar komunikasi dapat berjalan dengan baik. Apllikasi Kwikku semakin berkembang dan jumlah pengguna semakin bertambah dimana rata-rata peningkatannya sebanyak 21.000/tahun pada dua tahun terakhir. Fitur *Live Chat* yang ada masih belum siap menerima banyaknya request dari pengguna. Hal tersebut ditandai dengan adanya *delay* signifikan yaitu rata-rata 857,6ms untuk 200 pengguna pada waktu yang bersamaan pada kolom *chat*. Banyanknya pengguna terkadang menyebabkan *server down* dikarenakan *overload* kapasitas. Berdasarkan hasil analisis kondisi tersebut disebabkan oleh penggunaan metode *Long Polling* dan S*istem Relational Database* dimana penggunaanya terlalu banyak memakai *resource* dari *server*. Oleh karena itu, perlu adanya pembaharuan sistem dengan menggunakan protokol *Websocket*.

Websocket merupakan protokol komunikasi *client-server* yang didesain untuk *web browser*, namun saat ini dapat digunakan pada *client* atau *server* apapun. Protokol *Websocket* menerapkan komunikasi *full-duplex* sehingga memungkinkan lebih banyak interaksi antara *client* dan situs *web*, memfasilitasi konten *live*, aplikasi *real-time* dan komunikasi dua arah (Zhangling & Mao, 2012). *Websocket* membutuhkan *resource* yang lebih sedikit apabila dibandingkan dengan HTTP *long polling* dikarenakan proses *handshake* hanya dilakukan satu kali di awal(Liu & Sun, 2012). Optimisasi dari segi penyimpanan data juga diperlukan untuk mempercepat komunikasi *client* dan *server*. *Relational Database* yang digunakan saat ini memakan *resource* yang besar apabila dibandingkan dengan sistem *Non-relational* Database. *Non-relational Database* atau biasa disebut Nosql lebih unggul dari segi *cos*t, performa, dan kemampuan untuk mengelola data dalam skala besar (Kunda & Phiri, 2017).*Websocket* memiliki sistem *Full-Duplex* yang berarti klien dan *server* dapat mengirim dan menerima pesan di seluruh *channel*. *Long Polling* memiliki sistem *Half-Duplex* yang berarti bahwa siklus *request-response* baru diperlukan setiap kali klien ingin mengirimkan pesan ke *server*. *Long polling* biasanya menghasilkan *latency* rata-rata yang sedikit lebih tinggi dan daripada *Websockets*. *Websockets* mendukung kompresi per-pesan. *Long Polling* biasanya beroperasi dalam *batch* yang secara signifikan dapat meningkatkan efisiensi kompresi pesan.

Berdarkan uraian tersebut maka diperlukan adanya pembaharuan sistem *Live Chat* pada *Website* Kwikku. Tujuan yang diharapkan dari penelitian ini adalah menganalisis efisiensi Protokol *Websocket* dibandingkan dengan metode *Long Polling* pada aplikasi *live chat* pada *website* Kwikku.com.

2. LANDASAN KEPUSTAKAAN

2.1 Live Chat

Live chat merupakan sebuah fitur pada sebuah *platform* yang melibatkan pengguna untuk berkomunikasi baik kepada sistem itu sendiri maupun antar pengguna. Sebuah sistem memiliki harus memperhatikan beberapa hal berikut untuk dapat disebut sebagai sistem *live chat*: (1) *Reliability & availability*, Sistem harus mampu melaksanakan task sesuai dengan keinginan pengguna. (2) *Responsiveness*, Sistem harus memberikan informasi terkait proses pelaksanaan task serta harus dapat selalu siap untuk melayani pengguna. (McLean & Osei-Frimpong, 2017)

2.2 *Websocket*

Websocket merupakan protokol komunikasi yang memiliki sistem komunikasi dua arah dari *server* ke c*lient* dan *client* ke *server*. Protokol ini dikembangkan oleh HTML 5. *Websocket* dirancang untuk digunakan pada web browser. Namun, seiring berjalannya waktu teknologi ini juga dapat digunakan pada perangkat bergerak dengan bahasa pemrograman Android dan *Swift*. *Websocket* mempunyai kemampuan untuk memberikan update secara *real-time* yang sebelumnya menggunakan metode *long polling*. Keuntungan utama menggunakan *Websocket* adalah

mengurangi kebutuhan sumber daya baik di sisi klien maupun server. *Websocket* menggunakan HTTP sebagai mekanisme transport, komunikasi tidak seketika berakhir setelah respon diterima oleh *client*, melainkan selama koneksi masih terbuka *client* dan *server* dapat salaing mengirim pesan secara *asynchronous* (Lombardi, 2015).

2.3 Socket.io

Socket.io merupakan *library* berbasis bahasa pemrograman *Javascript*. *Library* ini merupakan pengembangan dari protokol *Websocket* native yang memiliki fitur *low-latency, bidirectional* dengan tambahan fitur *fallback* sehingga koneksi yang dibuat lebih konsisten. Socket.io juga didesain agar lebih mudah digunakan dengan *high-level syntax.* (Socket.io, 2022). Library memiliki beberapa fitur utama yang dapat membantu komunikasi menggunakan protocol websoket yaitu HTTO *long-polling fallback, Automatic recpnnection, Packet bufferin, Acknowledgements, Broadcasting, Multiplexing*

2.4 *Non-Relational Database*

Non-Relational Database atau biasa disebut NoSQL merupakan skema database baru yang menghilangkan konsep relasi antar data. Hal ini dapat menjadi kelebihan maupun kekurangan sesuai dengan kebutuhan sistem. Secara performa NoSQL dapat mengelola data dengan cepat dibandingkan dengan *Relational Database*, hal ini disebabkan data yang diakses oleh *client* diambil langsung dari data yang ada. Hal ini berbeda dengan konsep *Relational Database* yang memerlukan kompilasi data dari banyak tabel yang kemudian digabung menjadi satu. NoSql memiliki beberapa kategori yaitu *Key-Value, Document Oriented, Column Family* dan *Graph.* Konsep NoSql memiliki fitur-fitur dasar yang menjadi keunggulan yaitu:

1. *Scale-out*

Scaling out mengacu pada pencapaian kinerja tinggi dalam lingkungan terdistribusi. Basis data NoSQL memungkinkan distribusi data ke banyak *client* dengan distribusi beban pemrosesan. Banyak database NoSQL memungkinkan distribusi data otomatis ke *client* baru ketika ditambahkan ke cluster.

2. *Flexibility*

Fleksibilitas dalam hal struktur data mengatakan bahwa tidak perlu mendefinisikan skema untuk sebuah database. NoSQL tidak memerlukan skema yang telah ditentukan sebelumnya. Hal ini memungkinkan pengguna untuk menyimpan data dari berbagai struktur di tabel yang sama. Namun, dukungan untuk kueri tingkat tinggi seperti SQL tidak didukung oleh sebagian besar database NoSQL (Biswajeet Sethi, Samaresh Mishra, 2014).

2.5 MongoDB

Mongo DB merupakan salah satu Non-Relational Database dengan kategori *Document Oriented*. Data pada MongoDB merupakan sebuah dokumen yang memiliki komposisi key-value seperti data dalam bentuk JSON. Bentuk data seperti ini merupakan data *native* pada bahasa pemrograman secara umum sehingga memiliki tingkat kompatibilitas yang tinggi. MongoDB juga mereplika bahasa kueri yang mendukung CRUD (*Create Read Update Delete*). (Gaikwad, 2021).

2.6 Java Script

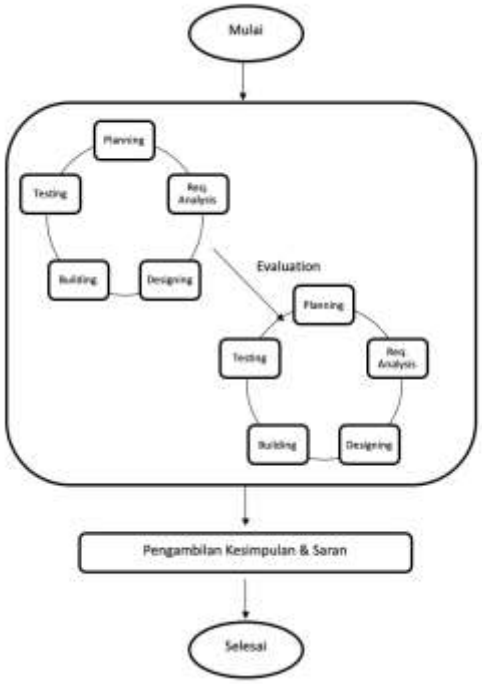
JavaScript adalah bahasa pemrograman berbentuk kumpulan *script* yang berjalan pada suatu dokumen HTML. JavaScript dapat menyempurnakan tampilan dan sistem pada halaman *web-based application* yang dikembangkan. Karakterstik dari bahasa pemrograman ini adalah bahasa pemrograman berjenis high-level programing, bersifat client-side, berorientasi pada objek, bersifat loosely typed (Mariko,2019).

2.7 Node JS

Node JS merupakan *software* yang didesain untuk mengembangkan aplikasi berbasis web dan ditulis dalam bahasa pemrograman Javascript. Node JS melengkapi fungsi Javascript yang pada umumnya berjalan pada sisi *client*, Node JS dapat berjalan pada sisi *server.* Node JS memiliki sifat *asynchronous* sehingga dapat berjalan secara fleksibel. (Stenberg, 2020)

3. METODOLOGI

Metodologi penelitian adalah alur atau tahap-tahap yang dilakukan pada penelitian. Secara umum, pembuatan sistem akan digambarkan melalui diagram 1



Gambar 1. Alur Penelitian

3.1 Planning

Pada tahap ini dilakukan penggalian permasalahan yang ada dengan melihat fakta-fakta terkait penelitian serta membaca literatur berupa jurnal dan buku pedoman lain. Permasalahan yang didapat kemudian dirumuskan menjadi suatu latar belakang penelitian skripsi ini. Selanjutnya dilakukan identifikasi apa saja variabel yang terlibat dalam masalah tersebut untuk merumuskan sebuah solusi dalam mengatasi masalah tersebut. Pada tahap ini juga akan dilakukan studi literatur untuk mendalami konsep yang digunakan dalam penelitian. Studi literatur berisikan landasan teori yang terkait dengan penelitian. Studi literatur dapat membantu untuk mengetahui dan memahami teori-teori dalam menyelesaikan permasalahan yang ada. Teori-teori pendukung ini diperoleh dari buku, jurnal, *e-book, website* dokumentasi resmi dan penelitian sebelumnya yang berkaitan dengan penelitian ini.

3.2 Requirement Analysis

Tahap ini bertujuan untuk mengetahui apa saja yang dibutuhkan untuk mengembangkan sistem. Pada tahap ini dilakukan elisitasi kebutuhan, analisa kebutuhan, dan spesifikasi kebutuhan. Elisitasi kebutuhan dilakukan untuk menggali permasalahan, fungsi-fungsi yang harus ada di dalam sistem, proses kerja sistem yang dibutuhkan serta batasan sistem. Analisis kebutuhan dilakukan untuk menganalisis kembali kebutuhan yang didapatkan pada saat elisitasi kebutuhan. Spesifikasi kebutuhan bertujuan untuk membuat kebutuhan yang telah didefinisikan sebelumnya menjadi lebih spesifik, jelas, lengkap dan konsisten. Kebutuhan yang telah didapatkan kemudian dimodelkan menjadi diagram *use case*. Diagram *use case* mendeskripsikan kebutuhan-kebutuhan dan fungsionalitas perangkat lunak. *Use case* diagram akan dijelaskan lebih rinci pada *use case scenario*.

3.3 *Designing*

Setelah mengetahui kebutuhan dan fungsionalitas sistem yang akan dibangun maka selanjutnya dilakukan perancangan suatu program aplikasi *web*. Pada tahap perancangan dilakukan pemodelan *Unified Modelling Language* (UML) secara rinci yang terdiri dari *sequence diagram, class diagram*, perancangan data, perancangan komponen, dan perancangan antarmuka. Perancangan sistem bertujuan untuk mempermudah implementasi maupun pengujian. Pembuatan *Backlog* pada aplikasi Jira juga akan dilakukan untuk mempermudah manajemen Scrum sebelum memasuki tahap implementasi.

3.4 *Building*

Tahapan *Building* dilakukan guna merealisasikan hasil dari perancangan. Implementasi sistem akan menggunakan pola *client-server*. Sisi *server* akan diimplementasikan menggunakan Node.JS dan MongoDB sedangkan sisi *client* akan dibangun menggunakan bahasa PHP *Native*, HTML, Javascript dan CSS.

3.5 *Testing*

Testing atau pengujian merupakan tahapan eksekusi sistem dengan tujuan untuk menemukan kesalahan dari sistem. Pengujian dilakukan pada kebutuhan fungsional dan non fungsional dari sistem. Pengujian dilakukan dengan metode *load testing* akan dilakukan guna untuk mengetahui dan membandingkan seberapa response time sistem *long-polling* dan *Websocket* dalam menangani beberapa jumlah user. Pada tahap ini juga akan dilakukan evaluasi bersama dengan *stakeholder* yaitu pihak Kwikku.com sehingga sistem dapat sesuai dengan kebutuhan dari Kwikku.com

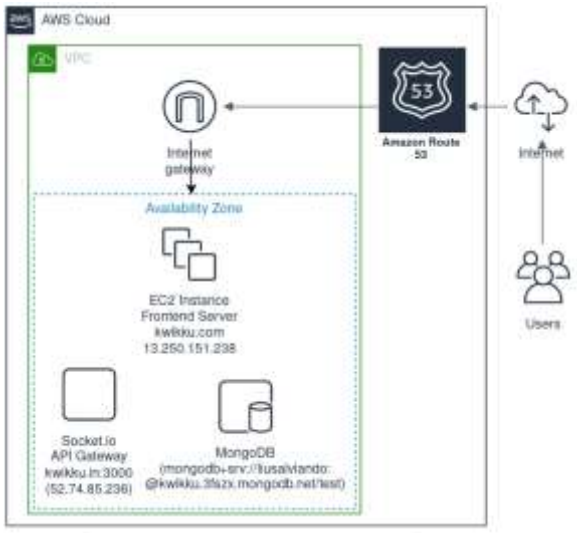
3.6 Penarikan Kesimpulan dan Saran

Penarikan kesimpulan dilakukan pada akhir penelitian setelah sistem sudah sepenuhnya di implementasi. Kesimpulan didapatkan dari hasil proses pengembangan dan menjawab dari rumusan masalah yang telah dideskripsikan. Saran didapatkan dari kekurangan dari hasil sistem pada penelitian ini yang ditujukan untuk penyempurnaan sistem kedepannya.

3.7 Perancangan

a. Perancangan Sistem

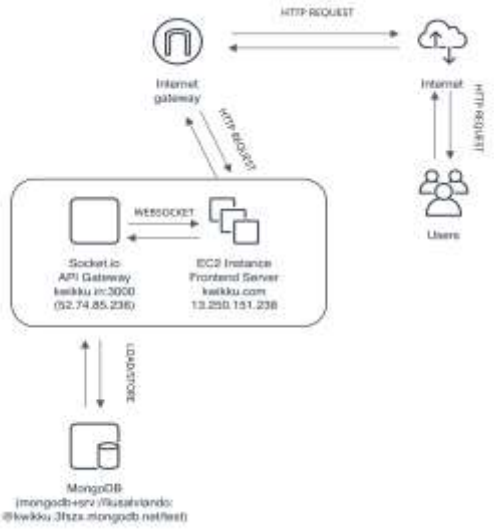
Pada perancangan dilakukan perancangan Topologi jaringan, perancangan arsitektur, perancangan perangkat perangkat, infrastruktur jaringan, pemodelan *sequence diagram*, pemodelan *class diagram*, perancangan *database* dan perancangan komponen REST API.



Gambar 2. Diagram Topologi Jaringan

Secara keseluruhan, *server* dibangun menggunakan Amazon Web Service. Terdapat tiga *server* terpisah yang akan digunakan sistem. Yang pertama adalah Web Server yang berfungsi untuk melayani bagian Front End. Kemudian *Websocket* dan API akan dibangun pada server tersendiri yang terhubung dengan Database Server.

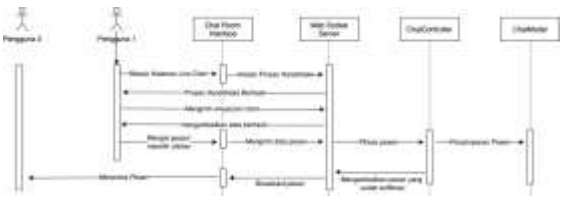
Perancangan aritektur, Pengguna melakukan HTTP request pada server kwikku.com untuk memperoleh response berupa interface sistem dan melakukan proses handshake pada server *Websocket*. Pengguna berkomunikasi dengan pengguna lainnya melalui protokol *Websocket* yang ada di server.



Gambar 3. Diagram Arsitektur

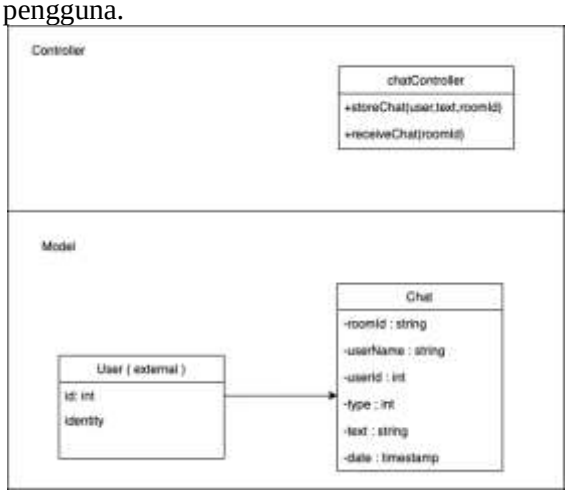
Perancangan Perangkat, Pada keseluruhan sistem yang berinteraksi terdapat 2 server dan 1 *cluster* database secara keseluruhan, yaitu *server backend* dan *server frontend. Server frontend* bertugas menampilkan data yang diterima dan akan dikirimkan oleh *client* sedangkan *server backend* bertugas menerima data dan melakukan penyimpanan data ke cluster database, serta menyediakan data saat dilakukan *request client*

Sequence diagram memodelkan hubungan antar objek dalam sistem. Pemodelan ini mendeskripsikan alur dari sistem untuk proses pengiriman dan penerimaan pesan. *Sequence diagram* tersebut menggambarkan proses dari penerimaan dan pengiriman pesan melalui *server Websocket* yang berfungsi untuk komunikasi dua arah. Setelah sinyal pesan diterima oleh *server Websocket*, data yang berupa JSON diteruskan ke controller untuk dilakukan pengolahan pesan lalu diteruskan lagi ke model untuk proses penyimpanan ke *database*. Setelah data tersimpan, maka pesan akan di-*broadcast* kepada seluruh pengguna pada *Chat Room.*



Gambar 4. *Sequence* Diagram Pengiriman dan Penerimaan Pesan

Pemodelan Class Diagram, Pemodelan ini menggambarkan kelas yang akan dibentuk pada sistem. Pada c*lass controller* terdapat sebuah *class* yaitu chatController yang akan berfungsi untuk menangani pengolahan pesan dan memanggil *class* pada *class model*. Pada *class model* terdapat sebuah *class* yaitu Chat yang berisikan skema data pada *database*. Object Chat akan terhubung dengan data pada *database* eksternal yaitu *User* yang berisikan identitas pengguna.

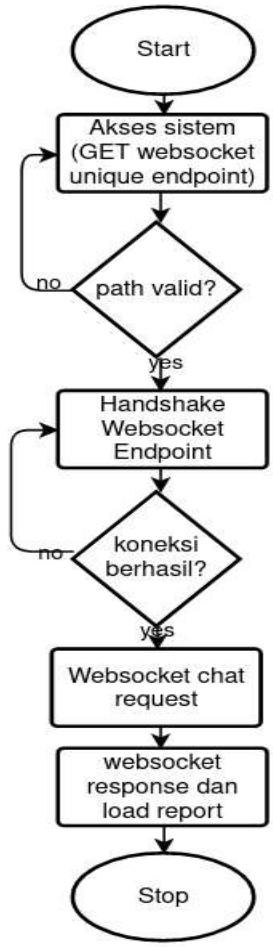


Gambar 5. Class Diagram Sistem

b. Perancangan Pengujian

Pada bagian pengujian akan dilakukan pengujian berupa *load testing*. Pengujian ini dilakukan untuk mengetahui bagaimana sistem berperilaku ketika banyak pengguna mencoba untuk mengakses data secara bersamaan. Hasil pengujian ini digunakan sebagai acuan berapa jumlah pengguna yang dapat ditangani oleh sistem dalam suatu waktu.

Load testing akan dilakukan menggunakan Apache JMeter dengan *plugin Websocket tester* dan http *request tester*. Pengujian dilakukan dengan beberapa skema yang telah disusun. Terdapat 5 skema yang masing masing diulang sebanyak 3 kali. Setiap skema secara berurutan jumlah user yang melakukan request sebanyak 100, 200, hingga 500 pada skema kelima. Setiap request user mengirimkan pesan yang berisi 140 karakter. Hasil dari pengujian akan dihitung menggunakan rumus efisiensi untuk memperoleh hasil akhir pengujian. Terdapat beberapa prosedur awal yang perlu dilakukan sebelum didapatkan hasil. Pada gambar dibawah ini adalah proses pengujian dari load testing yang dilakukan pada *Websocket*



Gambar 6. Alur *Test Case* Pengujian *Websocket*

Gambar 7. Detail *Thread* Pada *Apache Jmeter*

Pengujuan pada long polling terdiri dari dua *sub-process* yaitu pengujian pada *end point get chat* serta *end point send chat*. Untuk detail *thread* pada Apache Jmeter dapat dilihat pada gambar berikut.

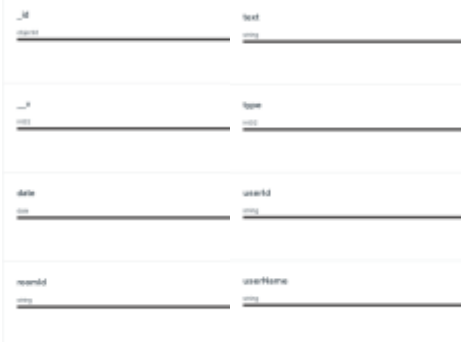
Gambar 8. Detail *Thread* Pada *Apache Jmeter*

3.8 Implementasi

Tahapan yang dilakukan pada implementasi sistem adalah implementasi *database* dan implementasi komponen REST API dan *Websocket*.

1. Implementasi Database

Basis data yang digunakan dalam proses implementasi adalah *Non-Relational Database* Mongodb. Berikut merupakan detail tabel *chat* yang dibuat.



Gambar 9. Implementasi Database

2. Implementasi Komponen REST API dan *Websocket*

Implementasi komponen REST API dan *Websocket* adalah proses melakukan coding program.

Index.js

```
io.sockets.on("connection", socket => {
    socket.on('join room', (roomId, user, socketId) => {
      ...
    })
});
```

Chat_javascript.php

```
var socket = io('https://kwikku.in:3000');
window.addEventListener('load', (event) => {
    socket.emit('join room','test');
})
```

Gambar 10.Connect and Join Room

Index.js

```
socket.on("chat",(data) => {
var filter = storeChat(data.user,data.type,data.text,roomId)
socket.to(roomId).emit("chat",data.user,data.type,filter)
})

```

chatController.js

```
const storeChat = (user,type, text, roomId) => {
    if(type == 2){
        var stickerId=text.replace('(sticker-','');
        stickerId=stickerId.replace('.png)','');
        text="https://kwikku.us/uploads/public/images/
        sticker/content/"+stickerId+".png"
    }
    process.env.TZ = 'Asia/Jakarta'
    var filteredText = filter.clean(text)
    const chat = new Chat({
        roomId: roomId,
        userName: user.userName,
        userId: user.userId,
        type: type,
        text: filteredText,
        date: new Date().getTime()
    })
    chat.save();
    return filteredText
}
```

Chat_javascript.php

```
socket.to(roomId).emit('chat message',{user,type,text});
$("#inputLiveChat").val("");
$("#chatBoxLive").animate({scrollTop:
$("#chatBoxLive")[0].scrollHeight}, 1000);
waiting = 1;
setTimeout(
      function(){
      waiting = 0;
      },2000
);
```

Gambar 11. Send Chat

chatController.js

```
const receiveChat = (roomId) => {
    Chat.find({roomId: roomId}).exec().then(
    function(resultChat){
        return(resultChat)
    })
}
```

route.js

```
router.get("/api/room/get/:roomId", async (req, res) => {
    if(req.params.roomId == '' ){
        res.send('roomId tidak boleh kosong')
    }else{
        var data={}
        var roomId = req.params.roomId
        var data = receiveChat(roomId)
        res.send(data)
    }
})
```

Chat_javascript.php

```
$.ajax({
      url: "https://kwikku.in/ api/room/get/test ",
      method: "GET",
      success: function(response){
            var data = response;
            var me="<?php echo $login_username;?>";
            for(var a=0;a<data.length;a++){
            if(me==data[a]. userName){
            var realname =
"<span class='text_blue'>"+data[a].userName+"</span>";
            }else{
            var realname =
"<span>"+data[a]. userName +"</span>";
            }
            var template = "<div><b>"+realname+" :</b>
"+data[a].text+"</div>";

      $("#listLiveChat").append(template);
            }
      $("#chatBoxLive").animate({scrollTop:
      $("#chatBoxLive")[0].scrollHeight}, 10);
      }
});
```

route.js

```
socket.on('chat message', function(msg){
      var me="<?php echo $login_username; ?>";
      if(me==msg.user.userName){
      var realname =
"<span class='text_blue'>"+msg.user.userName+"</span>";
      }else{
      var realname =
 "<span>"+msg.user.username+"</span>";
      }
      if(msg.text == "□"){
            $('#fwBtnShow').click();
      }
      var template =
"<div><b>"+realname+" :</b> "+msg.filter+"</div>";
$("#listLiveChat").append(template);
$("#chatBoxLive").animate({scrollTop:
$("#chatBoxLive")[0].scrollHeight}, 10);
});
```

***Gambar 12**.Receive Chat*

4. PENGUJIAN DAN ANALISIS

pengujian dari sistem yang dibangun. Pengujian sistem terdiri dari pengujian performa yang akan dilakukan dengan aplikasi *Apache Jmeter*.

4.1 Load Testing

Load testing digunakan untuk menguji waktu respon aplikasi *chat* berbasis *website* dengan sistem yang telah diimplementasikan. Pengujian ini dilakukan menggunakan aplikasi Apache Jmeter dengan simulasi berkala mulai dari 100 pengguna sampai 500 pengguna dengan pengulangan sebanyak 3 kali. *Testing* dilakukan sebanyak dua kali pada sistem dengan implementasi *long polling* dan pada sistem dengan implementasi *Websocket*. Hasil dari pengujian ini dinyatakan dalam *Summary Report Listener* yang menampilkan data *latency* atau *delay* tiap *request* dalam milisekon. Berikut tabel hasil pengujian yang dilakukan pada sistem dengan implementasi protokol *Websocket*:

Tabel 1. Hasil Pengujian *Websocket*

Percobaan	1				2				3			
Pengguna	Min (ms)	Max (ms)	Avg. (ms)	Std. Dev. (ms)	Min	Max	Avg	Std. Dev	Min	Max	Avg	Std. Dev
100	0	700	238	198,86	0	617	184	191,34	0	715	241	227,48
200	0	660	214	195,55	0	764	209	247,56	0	747	216	233,17
300	0	1971	804	648,59	0	1804	289	599,8	0	1929	522	477,19
400	0	2482	922	588,82	0	2079	682	737,72	0	1880	506	558,36
500	0	6808	1452	1800,89	0	5415	1159	1243,31	0	9034	1431	1505,97

Tahap selanjutnya dilakukan pengujian kedua dengan skenario pengujian yang sama. Pengujian ini dilakukan untuk menguji sistem dengan implementasi long polling. Berikut tabel hasil pengujian yang dilakukan pada sistem dengan implementasi long polling:

Percobaan	1				2				3			
Pengguna	Min (ms)	Max (ms)	Avg. (ms)	Std. Dev. (ms)	Min (ms)	Max (ms)	Avg(ms)	Std. Dev. (ms)	Min (ms)	Max (ms)	Avg (ms)	Std. Dev. (ms)
100	78	1854	865	663,86	80	1855	848	624,98	82	2874	928	697,7
200	56	2456	826	842,35	59	3363	1117	3363	51	2449	636	751,14
300	55	3366	1003	1135,52	61	3830	1299	1403,89	80	4600	1220	1318,76
400	58	4625	1345	1457,49	141	3438	1863	1032,1	118	3510	1133	1118,62
500	61	6645	2218	2296,54	142	4132	1221	1218,6	184	4472	1370	1318,24

Pada tahap akhir dilakukan pengambilan waktu *response* rata-rata dari setiap skenario dengan jumlah *user* yang berbeda pada setiap implementasi sistem kemudian dilakukan perhitungan persentase efisiensi Berikut tabel hasil perbandingan efisiensi antara kedua implementasi sistem.

Tabel 2. Perhitungan Efisiensi

Pengguna	Rata-rata		
	WS (ms)	LP (ms)	Efisiensi(%)
100	221	651	66,05223
200	233	857,67	72,83327
300	538,33	1177,67	54,28814
400	573,33	1187	51,69896
500	1346	1603	16,03244
		Rata-Rata	52,181

Pada setiap pengulangan skenario pengujian yang telah dilakukan terlihat bahwa, *Websocket* lebih baik dan efektif untuk diimplementasikan karena menghasilkan delay response time yang lebih sedikit.

5. KESIMPULAN

Berdasarkan hasil analisa melalui *load testing* yang telah dilakukan dengan *sample* 100 hingga 500 user dan 3 repetisi dapat disimpulkan bahwa protokol *Websocket* lebih unggul jika dibandingan dengan *long polling* karena menghasilkan *delay* yang lebih kecil dan memiliki rata-rata efisiensi sebesar 52,181%. Protokol *Websocket* dengan menggunakan *non-relational database* dapat menjadi solusi untuk mengatasi permasalahan tingginya *delay* pada sistem

DAFTAR PUSTAKA

Andriansyah, D. (2019). Computer Based Information System Journal Performance Dan Stress Testing Dalam Mengoptimasi Website. *Cbis Journal*, *07*(01).

Biswajeet Sethi, Samaresh Mishra, P. ku. P. (2014). A Study of NoSQL Database. *International Journal of Engineering Research & Technology*, *67*(6), 14–21.

Kaiser, L. (1989). *Adjusting for baseline: Change or percentage change? Statistics in Medicine, 8(10), 1183–1190.* doi:10.1002/sim.4780081002

Gaikwad, S. S. (2021). *Getting Started with*

Kunda, D., & Phiri, H. (2017). A Comparative

Study of NoSQL and Relational Database. *Zambia ICT Journal*, *1*(1), 1–4. https://doi.org/10.33260/zictjournal.v1i1.8

Liu, Q., & Sun, X. (2012). Research of Web Real-Time Communication Based on Web Socket. *International Journal of Communications, Network and System Sciences*, *05*(12), 797–801.

Lombardi, Andrew. (2015). *Websocket*. O'Reilly Media, Inc.

Mariko, S. (2019). Aplikasi website berbasis HTML dan JavaScript untuk menyelesaikan fungsi integral pada mata kuliah kalkulus _ Mariko _ Jurnal Inovasi Teknologi Pendidikan.pdf. In *Jurnal Inovasi Teknologi Pendidikan* (Vol. 6, Issue 1, pp. 80–91).

McLean, G., & Osei-Frimpong, K. (2017). Examining satisfaction with the experience during a live chat service encounter-implications for website providers. *Computers in Human Behavior*, *76*, 494–508.

Muhammad, P. B., Yahya, W., & Basuki, A. (2018). Analisis Perbandingan Kinerja Protokol *Websocket* dengan Protokol SSE pada Teknologi Push Notification. *Jurnal Pengembangan Teknologi Informasi Dan Ilmu Komputer (JPTIIK) Universitas Brawijaya*, *2*(6), 2235–2242.

Ozieranska, A., Skomra, A., Kuchta, D., & Rola, P. (2016). The critical factors of Scrum implementation in IT project– the case study. *Journal of Economics and Management*, *25*(3), 79–96.

Pawar, R. P. (2015). A Comparative study of Agile Software Development Methodology and traditional waterfall model. *IOSR Journal of Computer Engineering (IOSR-JCE)*, 1–8.

Pressman, R. (2014). Software Engineering: A Practitioner's Approach. In *Software Quality Engineering: A Practitioner's Approach* (Vol. 9781118592).

Socket.io. (2022, March 3). *Socket.io: How it Works*. https://socket.io/docs/v4/how-it-works/

Stenberg, D. (2020). *About Node JS*. https://nodejs.org/

Zhangling, Y., & Mao, D. (2012). A Real-Time Group Communication Architecture Based on *Websocket*. *International Journal of Computer and Communication Engineering*, *1*(4), 408–411. https://doi.org/10.7763/ijcce.2012.v1.100