Institut
Teknologi
Sepuluh Nopember

TUGAS AKHIR - KI141502

RANCANG BANGUN APLIKASI WEB LELANG ONLINE *(E-AUCTION)* BERBASIS KERANGKA KERJA LARAVEL

RONAULI SILVA NATALENSIS SIDABUKKE
NRP 5113100142

Dosen Pembimbing I
Rully Soelaiman, S.Kom, M.Kom

Dosen Pembimbing II
Rizky Januar Akbar, S.Kom., M.Eng

JURUSAN TEKNIK INFORMATIKA
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya, 2017

(Halaman ini sengaja dikosongkan)

TUGAS AKHIR - KI141502

RANCANG BANGUN APLIKASI WEB LELANG ONLINE *(E-AUCTION)* BERBASIS KERANGKA KERJA LARAVEL

RONAULI SILVA NATALENSIS SIDABUKKE
NRP 5113100142

Dosen Pembimbing I
Rully Soelaiman, S.Kom, M.Kom

Dosen Pembimbing II
Rizky Januar Akbar, S.Kom., M.Eng

JURUSAN TEKNIK INFORMATIKA
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya, 2017

(Halaman ini sengaja dikosongkan)

UNDERGRADUATE THESIS - KI141502

E-AUCTION WEB APPLICATION DESIGN AND APPLICATION BASED ON LARAVEL FRAMEWORK

RONAULI SILVA NATALENSIS SIDABUKKE
NRP 5113100142

Supervisor I
Rully Soelaiman, S.Kom, M.Kom

Supervisor II
Rizky Januar Akbar, S.Kom., M.Eng

Department of INFORMATICS
Faculty of Information Technology
Institut Teknologi Sepuluh Nopember
Surabaya, 2017

(Halaman ini sengaja dikosongkan)

LEMBAR PENGESAHAN

RANCANG BANGUN APLIKASI WEB LELANG ONLINE *(E-AUCTION)* BERBASIS KERANGKA KERJA LARAVEL

TUGAS AKHIR

Diajukan Guna Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Bidang Studi Algoritma Pemrograman
Program Studi S1 Jurusan Teknik Informatika
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember

Oleh:
RONAULI SILVA NATALENSIS SIDABUKKE
NRP: 5113100142

Disetujui oleh Dosen Pembimbing Tugas Akhir :

Rully Soelaiman, S.Kom, M.Kom .....
NIP: 197002131994021001 (Pembimbing 1)

Rizky Januar Akbar, S.Kom., M.Eng .....
NIP: 198701032014041001 (Pembimbing 2)

SURABAYA
Juni 2017

(Halaman ini sengaja dikosongkan)

RANCANG BANGUN APLIKASI WEB LELANG ONLINE *(E-AUCTION)* BERBASIS KERANGKA KERJA LARAVEL

Nama	:	**RONAULI SILVA NATALENSIS SIDABUKKE**
NRP	:	**5113100142**
Jurusan	:	**Teknik Informatika FTIf**
Pembimbing I	:	**Rully Soelaiman, S.Kom, M.Kom**
Pembimbing II	:	**Rizky Januar Akbar, S.Kom., M.Eng**

Abstrak

Industri e-commerce berkembang dengan pesat di Indonesia, seiring dengan meningkatnya jumlah pengguna internet dan menjamurnya bisnis online atau sering disebut online shop. Salah satu jenisnya adalah lelang online, yaitu metode jual beli yang mengintegrasikan mekanisme lelang dengan Internet.

Dalam interaksi antara pelaku lelang online (penjual dan pembeli) pasti terjadi kegagalan/ketidakpuasan dalam transaksi lelang online.Berangkat sebuah paper yang membahas mengenai analisa kesalahan dan strategi lewat survey terhadap pengguna aplikasi lelang online di Taiwan, penulis membangun aplikasi lelang online yang disertai dengan tambahan fitur maupun saran dari paper tersebut.

Tidak hanya berdasarkan paper rujukan, penulis juga menganalisa aplikasi e-commerce yang umum digunakan di Indonesia baik user experience maupun alur transaksi, dan menambahkan beberapa fitur agar lebih sesuai dengan transaksi jual-beli online yang umum di Indonesia. Dengan aplikasi ini, diharapkan kegagalan dalam transaksi online dapat diperbaiki dan membuka peluang lelang online untuk meramaikan industri e-commerce di Indonesia.

***Kata-Kunci**: lelang online, strategi*

E-AUCTION WEB APPLICATION DESIGN AND APPLICATION BASED ON LARAVEL FRAMEWORK

Name	:	**RONAULI SILVA NATALENSIS SIDABUKKE**
NRP	:	**5113100142**
Major	:	**Informatics FTIf**
Supervisor I	:	**Rully Soelaiman, S.Kom, M.Kom**
Supervisor II	:	**Rizky Januar Akbar, S.Kom., M.Eng**

Abstract

E-commerce industry is growing rapidly in Indonesia, along with the increasing number of internet users and number of online shops is also growing. One of e-commerce type is online auction, a buy and sell method that integrates auction mechanism and the Internet.

In the interaction between online auction actors (buyers and sellers), inevitable failure/dissatisfaction of online auction transactions sometimes found. Started by analysing paper about online auction application typologies and strategies through an application's users survey, author want to build online auction application along with additional ideas and suggestions from the paper.

Author also analyzed and considering user experience, design and transaction flow local e-commerce platforms that are commonly used in Indonesia, in purpose to make the application suits Indonesian's users better. Furthermore, author hopes that this applications can reduce/prevent the expected failures in online transactions and open up online auction opportunity to enliven the e-commerce industry in Indonesia.

***Keyword**: online auction, typologies and strategies*

KATA PENGANTAR

Puji Syukur kepada Tuhan yang Maha Esa, atas berkatNya penulis dapat menyelesaikan buku berjudul **Rancang Bangun Aplikasi Web Lelang Online *(E-Auction)* Berbasis Kerangka Kerja Laravel**.

Selain itu, pada kesempatan ini penulis menghaturkan banyak terima kasih kepada pihak-pihak yang tanpa mereka, penulis tidak akan dapat menyelesaikan buku ini dengan baik :

1. ***Daddy Jesus*** - atas segala berkat, limpahan karunia, kesempatan dan rancangan jalanNya-lah penulis masih diberi nafas kehidupan, waktu, tenaga dan pikiran untuk menyelesaikan buku ini. *Thank you, Big Daddy.*
2. **Papa dan Mama** yang selalu menguatkan, menasehati, dan luar biasa sabar dalam mengingatkan penulis agar tidak lupa menjaga kesehatan dan tidak lupa ke gereja selama masa studi.
3. **Yth Bapak Rully Soelaiman** yang mengajarkan penulis *how to think scientifically* juga bimbingan, nasehat, saran dan memberikan penulis sisi pemikiran dan perspektif lain terhadap setiap masalah.
4. **Yth Bapak Rizky Januar Akbar** sebagai dosen pembimbing yang memberi bimbingan, saran teknis dan administratif, diskusi dan pemecahan masalah dalam pembuatan dan penulisan buku tugas akhir.
5. **Keluarga XL Future Leader Scholarship Camp Batch 5 & KSE ITS** yang telah menyadarkan, memberikan semangat dan inspirasi untuk terus melanjutkan tugas akhir di saat penulis kehilangan semangat.
6. **Keluarga Admin Lab. Pemrograman (2014 - 2017)**, yang telah memberikan penulis banyak pengalaman, pengetahuan dan cerita-cerita untuk dikenang.
7. **Keluarga Pengpro *Furions* dan HMTC Optimasi 2016** , yang mengajarkan penulis tentang cara berorganisasi, cara berbicara di depan publik, dan banyak lagi.
8. Bang Christo yang sudah jadi inspirasi dan semangat kuliah

penulis sejak tahun pertama masa studi penulis sampai saat ini.

9. **Keluarga Alumni Budi Mulia Siantar-Surabaya angkatan 2013** , teman setia disaat suka maupun duka.
10. Serta semua pihak yang tidak tertulis - yang telah turut membantu penulis dalam menyelesaikan Tugas Akhir ini.

Penulis menyadari bahwa Tugas Akhir ini masih memiliki banyak kekurangan. Oleh karena itu, penulis berharap kritik dan saran dari pembaca sekalian untuk memperbaiki buku ini ke depannya.

Surabaya, Juli 2017

Ronauli Silva N. Sidabukke

DAFTAR ISI

(Halaman ini sengaja dikosongkan)

DAFTAR TABEL

(Halaman ini sengaja dikosongkan)

DAFTAR GAMBAR

(Halaman ini sengaja dikosongkan)

DAFTAR KODE SUMBER

(Halaman ini sengaja dikosongkan)

BAB I

PENDAHULUAN

Pada bab ini akan dipaparkan mengenai garis besar Tugas Akhir yang meliputi latar belakang, tujuan, rumusan dan batasan permasalahan, metodologi pembuatan Tugas Akhir, dan sistematika penulisan.

1.1 Latar Belakang

Transaksi jual beli saat ini sudah dapat dilakukan lewat berbagai cara, antara lain menggunakan *e-commerce*, atau lewat *social media*, atau bisa dengan melelang di aplikasi lelang *online*. Sedikit berbeda dengan teknik penjualan di lelang online, karena aplikasi ini dapat diakses oleh banyak orang, tentu saja pelelang (*auctioneer*) tidak terbatas pada ruang lelang saja, tapi bisa berasal dari manapun selama mereka mengakses aplikasi tersebut. Lelang *online* ini tentu saja mendatangkan banyak manfaat, selain biaya yang lebih efisien dan hemat, dan juga tidak menguras waktu karena siapapun, kapanpun, dimanapun dapat mengajukan penawaran ataupun melelang barangnya tanpa harus pergi ke instansi tertentu dan melakukan lelang dengan cara konvensional.

Aplikasi serupa telah banyak, namun banyak aspek yang kurang dalam aplikasi tersebut, seperti informasi dari lelang tidak *reliable* (misal: stok barang ternyata sudah habis), alur proses yang tidak jelas sehingga membingungkan pengguna aplikasi, informasi yang kurang jelas, dan produk yang didapatkan ternyata tidak sesuai dengan informasi pada saat produk dilelang (*bad information*) (?, ?).

Dan dari masalah teknis aplikasi, beberapa sumber menyatakan bahwa ketidakjelasan alur proses yang kurang diperhatikan oleh para developer aplikasi lelang *online* menjadi beberapa alasan yang kuat mengapa lelang online masih kurang diminati (?, ?).

Diharapkan, dengan adanya aplikasi ini, beberapa kelemahan yang masih ada pada aplikasi lelang *online* saat ini dapat diperbaiki, dan juga dapat dapat membantu proses *online* yang ada di Indonesia, dan juga mampu memperbaiki citra aplikasi lelang *online* sehingga mampu meningkatkan minat masyarakat terhadap lelang *online*.

1.2 Rumusan Masalah

Rumusan masalah yang diangkat dalam tugas akhir ini adalah sebagai berikut:

1. Bagaimana membangun aplikasi lelang online berbasis web?
2. Bagaimana rancangan arsitektur aplikasi dan fitur yang menganalisa kelemahan aplikasi serupa dan strategi penyelesaian sesuai dengan paper acuan (?, ?)?

1.3 Batasan Masalah

Dari permasalahan yang telah diuraikan di atas, terdapat beberapa batasan masalah pada tugas akhir ini, yaitu:

1. Aplikasi berbasis web dengan bahasa pemrograman PHP.
2. Aplikasi berbasis kerangka kerja Laravel.
3. Basis data yang digunakan adalah PostgreSQL.
4. Aplikasi tidak mencakup proses pembayaran.

1.4 Tujuan

Tujuan dari pengerjaan Tugas Akhir ini adalah:

1. Membangun aplikasi lelang online berbasis web yang lebih kredibel sesuai dengan paper yang dijadikan acuan pada tugas akhir ini.

1.5 Metodologi

Langkah-langkah yang ditempuh dalam pengerjaan Tugas Akhir ini yaitu:

1. **Studi Literatur & Observasi**
   Pada tahap ini dilakukan pengumpulan dan penggalian informasi lewat literatur maupun artikel-artikel dari internet, yang diperlukan dalam proses perancangan dan implementasi sistem.
2. **Analisa dan Perancangan Sistem**
   Pada tahap ini, dilakukan analisa dan pendefinisian kebutuhan sistem yang digunakan untuk masalah yang dihadapi. Tahapan-tahapannya adalah sebagai berikut:
   a analisa aktor yang terlibat dalam sistem;
   b perancangan model kasus penggunaan;
   c perancangan bisnis proses dalam aplikasi;
   d analisa masalah-masalah yang sering muncul saat penulis membuat aplikasi sebelumnya;
   e perancangan dan desain arsitektur aplikasi; danj
   f perancangan antarmuka aplikasi.
3. **Implementasi**
   Tahap ini merupakan implementasi dari rancangan yang telah dibuat pada tahap sebelumnya.
4. **Pengujian dan Evaluasi** Pada tahap ini, dilakukan pengujian terhadap aplikasi terhadap fungsionalitas dan non-fungsionalitas aplikasi.

1.6 Sistematika Penulisan

Buku Tugas Akhir ini bertujuan untuk mendapatkan gambaran dari pengerjaan Tugas Akhir ini. Selain itu, diharapkan dapat berguna untuk pembaca yang tertarik untuk melakukan pengembangan lebih lanjut. Secara garis besar, buku

Tugas Akhir terdiri atas beberapa bagian seperti berikut ini:

Bab I Pendahuluan

Bab ini berisi latar belakang masalah, tujuan dan manfaat pembuatan Tugas Akhir, permasalahan, batasan masalah, metodologi yang digunakan, dan sistematika penyusunan Tugas Akhir.

Bab II Dasar Teori

Bab ini membahas beberapa teori penunjang yang berhubungan dengan pokok pembahasan dan mendasari pembuatan Tugas Akhir ini.

Bab III Analisa dan Perancangan Sistem

Bab ini membahas mengenai analisa dan perancangan aplikasi. Perancangan aplikasi meliputi perancangan data, arsitektur, proses dan struktur program.

Bab IV Implementasi

Bab ini berisi deskripsi lengkap implementasi aplikasi.

Bab V Pengujian dan Evaluasi

Bab ini membahas pengujian untuk menguji apakah aplikasi sudah tepat sasaran pada kebutuhan fungsional dan non-fungsional yang dirumuskan pada tahap Analisa dan Perancangan Sistem (Bab III).

Bab VI Kesimpulan dan Saran

Bab ini berisi kesimpulan dari hasil pengujian yang dilakukan, dan membahas saran beserta *further enchancements* untuk pengembangan sistem lebih lanjut

Daftar Pustaka

Merupakan daftar referensi yang digunakan untuk mengembangkan Tugas Akhir. **Lampiran** Merupakan bab tambahan yang berisi hal-hal terkait yang penting dalam aplikasi ini.

BAB II

LANDASAN TEORI

Pada bab ini akan dibahas mengenai teori-teori yang menjadi dasar dari pembuatan Tugas Akhir. Juga, teori-teori tersebut ditulis secara berurutan untuk mempermudah penulis mencari teori yang ingin dicari.

2.1 Amazon Web Service

Amazon Web Services adalah sekumpulan layanan-layanan berbasis Cloud Computing yang di sediakan oleh Amazon sejak tahun 2002. Meskipun salah satu perusahaan raksasa internet ini sering kita kenal untuk membeli buku dan lagu, namun sekarang Amazon telah menambah layanannya dalam hal infrastrutktur cloud computing. Amazon Web Services ini menyediakan layanan-layanan nya yang saling terintegrasi dan mudah kustomisasi(?, ?).

Dalam website resminya, disebutkan bahwa AWS dapat membantu aplikasi menjadi lebih cepat, lebih aman, dan menghemat *costs* dengan *scaling performance* menggunakan teknologi *cloud computing*(?, ?).

2.2 *Concurrency*

Konkurensi adalah bisa dikatakan sebagai suatu fitur di mana database management system(DBMS) mengijinkan banyak transaksi pada saat bersamaan untuk mengakses data yang sama. Dalam melakukan konkurensi dibutuhkan suatu Concurency Control Mechanism (CCM) agar transaksi yang dilakukan oleh banyak user pada suatu sistem di dalam waktu yang bersamaan tidak saling "mengganggu" dan tidak menghasilkan inconsistency data.

Tiga masalah umum yang muncul dalam konkurensi adalah sebagai berikut:

1. *Lost Update Problem*
   Masalah operasi update yang sukses dari seorang pengguna kemudian ditimpali oleh operasi update dari pengguna lain.
2. *Uncomited dependency problem* (ketergantungan yg tidak sukses/modifikasi sementara)
   Masalah terjadi saat suatu transaksi membaca data dari transaksi lain yg belum dicommit.
3. *Inconsistent analysis problem* (?, ?)

2.3 *Data Growth*

Yang dimaksud dalam *data growth* pada *section* ini adalah seberapa cepat perkembangan jumlah data yang disimpan oleh server. Data tersebut bisa berupa *row* dalam *database* ataupun data *gambar, video dll* (?, ?).

Sejak tahun 2000s, perkembangan data meningkat pesat dan memunculkan bisnis penyedia *data storage* dan *networking equipment*.

Sebagai ilustrasi, pada buku referensi tercatat bahwa *data equilibrium flow* firma-firma *e-Commerce* pada dekade 1990-2000 meningkat hingga 1.5 *billion gigabytes* setiap tahunnya(?, ?).

2.4 JSON *Web Token*

JSON Web Token atau lebih dikenal dengan JWT, merupakan sebuah token berbentuk JSON yang padat-informasi (ukurannya), informasi mandiri untuk ditransmisikan antar pihak yang terkait. Token tersebut ini dapat diverifikasi dan dipercaya karena sudah di-sign secara digital. Token JWT bisa di-sign dengan menggunakan secret (algoritma HMAC) atau pasangan public / private key (algoritma RSA) (?, ?).

2.5 Laravel

Laravel adalah *framework* PHP yang dikembangkan pertama kali oleh Taylor Otwell. Walaupun termasuk baru, namun komunitas pengguna laravel sudah berkembang pesat dan mampu menjadi alternatif utama dari sejumlah *framework* besar seperti CodeIgniter dan Yii. Laravel oleh para *developer* disetarakan dengan CodeIgniter dan FuelPHP namun memiliki keunikan tersendiri dari sisi *coding*. Laravel memiliki beberapa keunggulan, diantaranya:

1. Sintaks yang sederhana dan *programmer-fiendly*
2. Tersedia *generator* yang canggih dan memudahkan, Artisan CLI
3. Fitur *Schema Builder* untuk berbagai *database*
4. Fitur *Migration* dan *Seeding* untuk berbagai *database*
5. Fitur *Query Builder* yang powerful
6. Eloquent ORM (*Object Relational Mapping*)
7. Fitur pembuatan *package* dan *bundle*
8. *Dependency Injection* (?, ?)

2.6 Laravel Dusk

Laravel Dusk adalah sebuah fitur baru yang ditujukan untuk *functional testing*, yang baru diluncurkan dan dibenamkan secara *default* pada Laravel versi 5.4. Dalam *site* dokumentasinya, disebutkan bahwa Laravel Dusk menyediakan *browser automation & testing API* yang ekspresif dan mudah digunakan. Secara otomatis, Dusk tidak memerlukan instalasi JDK atau Selenium pada *host*, namun menggunakan *ChromeDriver standalone*, namun juga dapat menggunakan driver Selenium yang kompatibel (?, ?).

2.7 Lelang

Lelang adalah proses membeli dan menjual barang atau jasa dengan cara menawarkan kepada penawar, menawarkan tawaran harga lebih tinggi, dan kemudian menjual barang kepada penawar harga tertinggi. Dalam teori ekonomi, lelang mengacu pada beberapa mekanisme atau peraturan perdagangan dari pasar modal (?, ?).

Lelang menurut sejarahnya berasal dari bahasa Latin *auctio* yang berarti peningkatan harga secara bertahap. Para ahli menemukan bahwa dalam literatur Yunani, lelang telah dikenal 450 tahun sebelum Masehi. Jenis lelang yang populer saat itu antara lain adalah karya seni, tembakau, kuda, budak dan sebagainya(?, ?).

2.8 Lelang Daring / Lelang *Online*

Lelang adalah proses membeli dan menjual barang atau jasa dengan cara menawarkan kepada penawar, menawarkan tawaran harga lebih tinggi, dan kemudian menjual barang kepada penawar harga tertinggi. Dalam teori ekonomi, lelang mengacu pada beberapa mekanisme atau peraturan perdagangan dari pasar modal.

Sementara lelang daring atau lelang melalui internet muncul seiring dengan perkembangan internet. Barang atau jasa yang diperjualbelikan dipasang di situs dan peserta lelang dapat mengikuti acara lelang secara daring. Perusahaan lelang yang berhasil menggunakan sarana internet salah satunya adalah *Ebay*. Di Indonesia, lelang melalui internet (online) sudah dipelopori oleh pemerintah dengan situs lelang online yang dapat diakses melalui website resmi Kemenkeu (?, ?). Berikut adalah beberapa istilah yang ada dalam lelang online:

1. BID atau *Bidding*, artinya: Menawarkan

2. BIN (*Buy In Now*) artinya: Beli sesuai harga yang telah ditawarkan penjual
3. INC (*Increment*) artinya: Minimum kenaikan *bid* setelah *bid* sebelum nya (?, ?)

2.9 MongoDB

MongoDB (dari ”humongous”) adalah sistem basis data berorentasi dokumen lintas platform. Diklasifikasikan sebagai basis data ”NoSQL”, MongoDB menghindari struktur basis data relasional tabel berbasis tradisional yang mendukung JSON seperti dokumen dengan skema dinamis (MongoDB menyebutnya sebagai format BSON), membuat integrasi data dalam beberapa jenis aplikasi lebih mudah dan lebih cepat. Dirilis di bawah kombinasi dari GNU Affero General Public License dan Lisensi Apache, MongoDB adalah perangkat lunak bebas dan sumber terbuka(?, ?).

2.10 Node.js

Node.js adalah platform perangkat lunak pada sisi-server dan aplikasi jaringan. Ditulis dengan bahasa javascript dan bisa dijalankan pada Windows, Mac OS X dan Linux tanpa perubahan kode program. Node.js memiliki pustaka server HTTP sendiri sehingga memungkinkan untuk menjalankan webserver tanpa menggunakan program webserver seperti Apache atau Lighttpd (?, ?).

2.11 NoSQL

NoSQL adalah istilah yang dikenal dalam teknologi komputasi untuk merujuk kepada kelas yang luas dari sistem manajemen basis data yang diidentifikasikan dengan tidak

mematuhi aturan pada model sistem manajemen basis data relasional yang banyak digunakan.

NoSQL tidak dibangun terutama dengan table dan umumnya tidak menggunakan SQL untuk memanipulasi data, sehingga sering ditafsirkan sebagai “tidak hanya SQL” (?, ?).

2.12 npm / *Node Package Manager*

NPM memiliki dua fungsi utama, yaitu sebagai repositori online yang berisi banyak package atau module untuk aplikasi NodeJS dan yang kedua adalah sebuah utilitas baris perintah (command line) yang digunakan untuk menginstal paket-paket yang dibutuhkan dan juga untuk mengelola versi dan ketergantungan package dari NodeJS. Dengan NPM Anda akan mudah mencari, menginstal, uninstall aplikasi atau module/package Node.js(?, ?).

2.13 PostgreSQL

PostgreSQL adalah sebuah produk *database* relasional yang termasuk dalam kategori *free open source software* (*FOSS*). PostgreSQL terkenal karena fitur-fitur yang advanced dan pendekatan rancangan modelnya menggunakan paradigma *object-oriented*, sehingga sering dikategorikan sebagai *Object Relational Database Management System* (ORDBMS). Beberapa fitur PostgreSQL adalah sebagai berikut:

1. *Inheritance*, dimana satu table dapat diturunkan model dan beberapa karakteristik dari table lainnya.
2. *Multi-Version Concurrency Control* dimana user diberi data snapshot ketika suatu perubahan dilakukan sampai commit.
3. *Rules*, dimana suatu *query* DML yang dikirimkan ke server akan mengalami penulisan ulang (*rewrite*). Ini terjadi sebelum diproses oleh *query planner*.

4. dan berbagai fitur lainnya (?, ?)

2.14 Redis

Redis adalah *open source*, struktur data yang ditempatkan di memori, digunakan sebagai *database*, *cache* dan *message broker*. Redis mendukung struktur data seperti *string, sets, hash, lists* dan *sorted sets*. Sama seperti cache, setiap key diisi oleh value. Tapi kelebihannya, Redis bisa digunakan untuk melakukan operasi dari value tersebut. Cara terbaik untuk memahami redis adalah membuat model aplikasi tanpa memikirkan bagaimana caranya untuk menyimpan data di dalam *database* (?, ?).

2.15 *Repository Pattern*

Repository Pattern adalah sebuah pola dalam struktur *software engineering* yang memisahkan *data management layer* ke dalam sebuah layer tersendiri - yang di*handle* oleh sebuah bagian struktur yang disebut repository.

Jika menggunakan *pattern* ini, semua kode spesifik yang terkait dengan *persistence logic* dan implementasi akses data berhenti sampai di *repository* (*controller* hanya me*redirect request* dan validasi *request*)(?, ?).

Dalam sebuah referensi, disebutkan bahwa: ”*The repository pattern covers large centralized transaction-oriented databases, the blackboard systems used for some AI applications, and systems with predetermined execution patterns in which different phases add information to a single complex data structures (e.g., compilers). These variants differ chiefly in their control structure.*” (?, ?).

2.16 SendGrid

SendGrid adalah sebuah *customer comumunication platform* untuk email *marketing* dan transaksional yang berbasis di Denver, Colorado.

SendGrid menyediakan layanan pengiriman email yang berbasis *cloud* kepada pihak bisnis. Layanan yang ditawarkan sangat beragam, mulai dari *shipping notifications*, *friend requests* dan lain lagi.

Selain itu, juga dapat *handling* ISP monitoring, *domain keys, feedback loops*, dan juga memberikan report *opened mails, unsubscribes, bounces dan spam reports*. Pada tahun 2012, SendGrid menggaet Twilio dan menambahkan layanan integrasi SMS, Suara dan *push notification*. (?, ?)

2.17 *Service Worker*

Service worker adalah skrip yang dijalankan browser Anda di latar belakang, terpisah dari laman web, yang membuka pintu ke berbagai fitur yang tidak memerlukan laman web atau interaksi pengguna. Saat ini, service worker sudah menyertakan berbagai fitur seperti pemberitahuan push dan sinkronisasi latar belakang. Di masa mendatang, service worker akan mendukung hal-hal lainnya seperti sinkronisasi berkala atau geofencing. Fitur inti yang didiskusikan dalam tutorial adalah kemampuan mencegat dan menangani permintaan jaringan, termasuk mengelola cache respons lewat program(?, ?).

2.18 SMTP / *Simple Mail Transfer Protocol*

SMTP adalah suatu protokol yang digunakan untuk mengirimkan pesan e-mail antar server, yang bisa dianalogikan sebagai kantor pos. Ketika kita mengirim sebuah e-mail,

komputer kita akan mengarahkan e-mail tersebut ke sebuah SMTP server, untuk diteruskan ke mail-server tujuan (?, ?).

2.19 Socket.io

Socket.io adalah *library* Javascript untuk aplikasi web yang bersifat *realtime*. Socket.io menjembatani antara komunikasi dua arah antara *web clients* dan *server*. Socket.io terbagi menjadi dua bagian, yaitu *client-side library* yang berjalan di browser client, dan *server-side library* yang menggunakan Node.js. Kedua komponen tersebut mempunyai API yang sama. Seperti Node.js, Socket.io juga bersifat *event-driven*. Socket.IO menggunakan protokol *websocket* dengan *polling* sebagai opsi *fallback*. Meskipun Socket.IO merupakan 'pembungkus' untuk soket web, namun ia memiliki banyak fitur, antara lain broadcast ke banyak soket, dan I/O yang asinkronus (?, ?).

2.20 *Test Script*

Test Script dalam dunia *software testing* adalah set instruksi atau sekumpulan baris kode yang akan melakukan *testing* terhadap fungsionalitas sistem dengan target tertentu (?, ?).

Ada beberapa jenis *script test*:

1. *Manual testing*, atau lebih sering disebut *test cases*
2. *Automated Testing*

2.21 *Vendu Reglement*

Lelang dilegalisasi & resmi masuk Indonesia dalam perundang-undangan sejak 1908, yaitu dengan berlakunya *Vendu Reglement*, Stbl. 1908 No. 189 dan *Vendu Instructie*, Stbl 1908 No. 190. *Vendu Reglement* ini berisikan peraturan-peraturan dasar lelang yang berlaku hingga saat ini, dan menjadi dasar hukum penyelenggaraan lelang di Indonesia(?, ?).

2.22 Vue.js / Vue

Vue adalah sebuah *framework* Javascript yang *progressive* dan bersifat *open-source* untuk membangun *user interface*. Integrasi kedalam project yang menggunakan *library* Javascript lain menjadi lebih mudah dengan Vue karena Vue memang didesain untuk *incrementally adoptable*. Vue juga dapat berfungsi sebagai *web application framewowrk* untuk membangun sebuah *single-page applications*.

Dalam 2016 *Javascript Survey*, Vue mendapatkan 89% untuk kategori *developer satisfaction rating*. Vue mengakumulasikan 98 *stars Github* setiap hari, dan menduduki peringkat ke-10 *Project Github* dengan bintang terbanyak *all the time*(?, ?).

2.23 Whitelist

Whitelist sendiri adalah memindahkan daftar alamat atau domain dari pengirim email dengan harapan agar muncul pada kotak pesan email utama Anda. Sederhananya, hal ini dilakukan untuk menghindari agar pesan email yang dikirimkan oleh pengirim pesan sebenarnya tidak terbaca sebagai spam (?, ?).

BAB III

ANALISA DAN PERANCANGAN SISTEM

3.1 Analisa Sistem

Pada subbab ini, penulis akan memaparkan analisa terhadap kebutuhan baik fungsional dan non-fungsional, teknis dan serta pengaruhnya dalam perancangan aplikasi pada subbab **??**. Subbab Analisa Sistem akan dipaparkan secara sistematis sebagai berikut:

a. analisa kebutuhan dasar;
b. analisa aspek bisnis;
c. analisa kebutuhan fungsional;
d. analisa kebutuhan non-fungsional; dan
e. rancangan arsitektur, strktur dan teknologi yang digunakan.

.

3.1.1 Deskripsi Umum Aplikasi

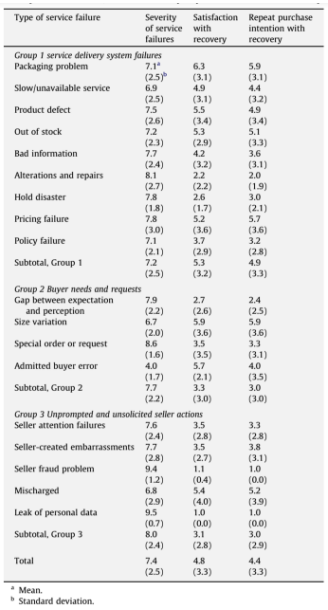
Bid! Bid! Bid! Istilah tersebut pasti tidak asing bagi para pecinta lelang online. Lelang sendiri adalah proses membeli dan menjual barang atau jasa dengan cara menawarkan kepada penawar, menawarkan tawaran harga lebih tinggi, dan kemudian menjual barang kepada penawar harga tertinggi (Wikipedia.com). Di Indonesia, sistem lelang sudah digunakan sejak jaman Hindia Belanda dimana saat itu sistem lelang pertama kali diperkenalkan di Indonesia dan biasa digunakan lelang terhadap aset-aset pejabat atau pemerintah yang dimutasi pada saat itu (Artikel sejarah lelang, 2011).

Dalam proses bisnisnya sendiri, lelang cukup sederhana, yaitu pengguna dapat melelang dan dapat menjual barang untuk dilelang. Yang berarti, pengguna dapat memanajemen riwayat lelang dan memanajemen barang yang ia daftarkan untuk dilelang. Hal ini akan didefinisikan lebih lanjut dalam subbab **??**, yaitu subbab Spesifikasi Kebutuhan Fungsional.

3.1.2 Analisa Paper Rujukan

Bercermin terhadap aplikasi *e-commerce* yang telah ada, masalah yang paling sering dialami adalah ketidakpuasan pengguna. Salah satu indikator bahwa suatu perusahaan dikatakan memiliki ketidakpuasan pelanggan adalah karena kegagalan dalam pelayanannya. Seorang pelanggan sangat mungkin memutuskan untuk komplain setelah mengalami ketidakpuasan terhadap layanan suatu perusahaan, dan jika tidak ditangani dengan baik, hal ini bisa berakibat fatal terhadap reputasi dan kepercayaan pengguna terhadap aplikasi tersebut.

Oleh karena itu, sebuah paper mengangkat topik ini khusus dalam bidang aplikasi lelang online, menganalisa kegagalan dan ketidakpuasan pengguna, beserta solusi-solusi yang ditawarkan oleh pengguna aplikasi untuk memperbaiki kegagalan pelayanan tersebut.



Type of service failure	Severity of service failures	Satisfaction with recovery	Repeat purchase intention with recovery
Group 1 service delivery system failures			
Packaging problem	7.1[a]	6.3	5.9
	(2.5)[b]	(3.1)	(3.1)
Slow/unavailable service	6.9	4.9	4.4
	(2.5)	(3.1)	(3.2)
Product defect	7.5	5.5	4.9
	(2.6)	(3.4)	(3.4)
Out of stock	7.2	5.3	5.1
	(2.3)	(2.9)	(3.3)
Bad information	7.7	4.2	3.6
	(2.4)	(3.2)	(3.1)
Alterations and repairs	8.1	2.2	2.0
	(2.7)	(2.2)	(1.9)
Hold disaster	7.8	2.6	3.0
	(1.8)	(1.7)	(2.1)
Pricing failure	7.8	5.2	5.7
	(3.0)	(3.6)	(3.6)
Policy failure	7.1	3.7	3.2
	(2.1)	(2.9)	(2.8)
Subtotal, Group 1	7.2	5.3	4.9
	(2.5)	(3.2)	(3.3)
Group 2 Buyer needs and requests			
Gap between expectation and perception	7.9	2.7	2.4
	(2.2)	(2.6)	(2.5)
Size variation	6.7	5.9	5.9
	(2.0)	(3.6)	(3.6)
Special order or request	8.6	3.5	3.3
	(1.6)	(3.5)	(3.1)
Admitted buyer error	4.0	5.7	4.0
	(1.7)	(2.1)	(3.5)
Subtotal, Group 2	7.7	3.3	3.0
	(2.2)	(3.0)	(3.0)
Group 3 Unprompted and unsolicited seller actions			
Seller attention failures	7.6	3.5	3.3
	(2.4)	(2.8)	(2.8)
Seller-created embarrassments	7.7	3.5	3.8
	(2.8)	(2.7)	(3.1)
Seller fraud problem	9.4	1.1	1.0
	(1.2)	(0.4)	(0.0)
Mischarged	6.8	5.4	5.2
	(2.9)	(4.0)	(3.9)
Leak of personal data	9.5	1.0	1.0
	(0.7)	(0.0)	(0.0)
Subtotal, Group 3	8.0	3.1	3.0
	(2.4)	(2.8)	(2.9)
Total	7.4	4.8	4.4
	(2.5)	(3.3)	(3.3)

[a] Mean.
[b] Standard deviation.

Gambar 3.1 Fatalitas kegagalan dalam aplikasi Lelang Online, Kepuasan terhadap Perbaikan Pelayanan dan *Repeat Purchase Intention* setelah Perbaikan Layanan

Dalam gambar diatas, dijabarkan beberapa jenis kegagalan yang pernah dialami oleh pengguna aplikasi serta fatalitas/pengaruh buruk kegagalan tersebut terhadap kepercayaan pengguna.

Recovery strategy	No. (%)	No. of satisfactory recovery	No. of dissatisfactory recovery	Satisfaction with recovery	Retention with recovery
Correction	280 [a] (32.3) [b]	228[a] (81.4)[b]	52[a] (18.6)[b]	7.2[c] (2.4)[d]	6.6[c] (2.7)[d]
Correction plus	82 (9.5)	77 (93.9)	5 (6.1)	7.7 (2.1)	7.6 (2.5)
Discount	26 (3.0)	23 (88.5)	3 (11.5)	7.0 (2.2)	6.0 (2.9)
Replacement	14 (1.6)	2 (14.3)	12 (85.7)	3.4 (2.7)	3.5 (2.7)
Store credit	16 (1.8)	9 (56.3)	7 (43.7)	6.1 (2.9)	6.4 (2.7)
Apology	54 (6.2)	20 (37.0)	34 (63.0)	3.7 (2.6)	3.4 (2.6)
Refund	67 (7.7)	42 (62.7)	25 (37.3)	6.4 (2.9)	5.2 (3.1)
Unsatisfactory correction	182 (21.0)	0 (0)	182 (100.0)	1.9 (1.3)	1.8 (1.6)
Failure escalation	39 (4.5)	0 (0)	39 (100)	1.3 (0.9)	1.3 (0.9)
Nothing	107 (12.4)	0 (0)	107 (100)	1.7 (1.2)	1.7 (1.6)
Total	867 (100)	407 (46.9)	460 (53.1)	4.8 (3.3)	4.4 (3.3)

[a] Number.
[b] %.
[c] Mean.
[d] Standard deviation.

Gambar 3.2 Kategori Perbaikan terhadap Kegagalan Pelayanan Lelang Online

Maka berdasarkan hasil analisa tersebut, fitur-fitur yang ditambahkan selain daripada fitur dasar aplikasi lelang online sebagai *added vaue* adalah sebagai berikut:

1. Fitur chatting, untuk mengurangi kemungkinan *Bad Information* dimana ekspektasi dan persepsi terhadap barang yang dilelang antara pembeli dan penjual tidak sama dan *Special Needs*,
2. Fitur pemberian kupon voucher (*Discount and Correction*

Plus) yang bisa berupa *free shipping* atau *discount*.

3.1.3 *Bussiness Aspects of Software Engineering*

Lelang merupakan salah satu metode pertukaran barang dan jasa dengan metode penetapan harga yang berbeda dengan perdagangan. Oleh karena itu, lelang juga termasuk dalam kategori bisnis. Yang menarik adalah, ketika bisnis digabungkan dengan teknologi atau yang sering disebut *e-commerce*, hal yang sekedar pertukaran barang bertransformasi menjadi sebuah sistem interaktif yang kompleks dimana tujuan utamanya adalah menarik pengunjung/pengguna untuk menyelesaikan sebuah transaksi. Hal ini tentu sangat krusial, penting, dan tertantang untuk menyelesaikannya.

Dalam mencapai kesuksesan dan tingkat kompetitif yang tinggi, haruslah menyediakan layanan dengan kesan *user experience (UX)* yang positif bagi para penggunanya. Morville (?, ?, p. 27) , dalam studi yang dilakukannya, menyebutkan bahwa UX tercakup dalam 7 aspek esensial, yaitu

a. *useful*
b. *usable*
c. *findable*
d. *desirable*
e. *accessible*
f. *credible*.

Hasil-hasil temuan penting yang menarik dalam pengaruh *user experience*, adalah sebagai berikut dikutip dari sebuah sumber adalah sebagai berikut:

a. *User tend to leave if a page loads more than 3 seconds*;
b. *79% of users won't return if the web's performance and experience is poor*;
c. *44% of users will tell the poor experiences to their friends*.

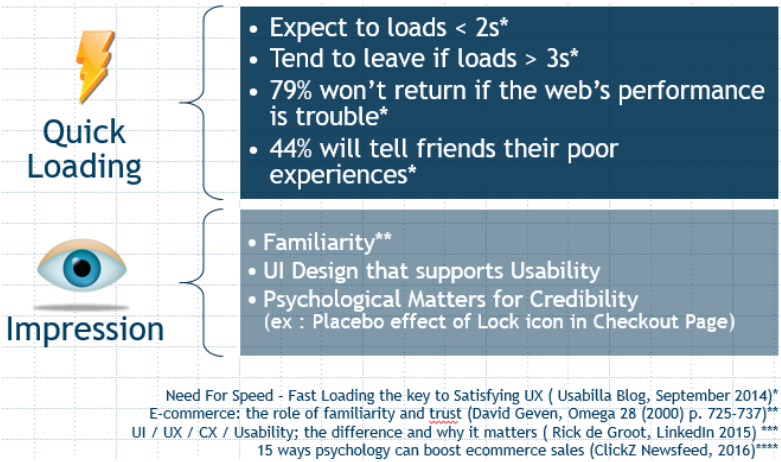
Selain dari faktor *user experience* dan *performance*, beberapa hal yang menjadi poin penting dan menarik dalam beberapa studi

yang terkait adalah sebagai berikut:

a *Familiarity* - yang dapat didefinisikan sebagai tingkat familier atau kesamaan dengan sistem sejenis ternyata dapat membangun *trust* sehingga mensugesti pengguna untuk menyelesaikan transaksi yang dilakukan;

b *Usability* yang memudahkan pengguna dalam menyelesaikan transaksi; dan

c Aspek-aspek psikologi seperti pemilihan warna, penggunaan *icon* yang sesuai, seperti *icon* gembok pada halaman pembayaran ternyata dapat mengesankan *security* pada pengguna.



Gambar 3.3 Visualisasi aspek bisnis dalam *software engineering*

Dari hasil temuan ini, dapat disimpulkan bahwa *user experiences, performances, usability* dan psikologi memiliki pengaruh besar dalam kesuksesan lelang online dalam menarik hati para penggunanya. Hal ini akan mempengaruhi definisi kebutuhan fungsionalitas yang akan dibahas dalam subbab **??**.

3.1.4 *Technical Analysis*

Untuk membuat sebuah aplikasi yang sukses, tentunya banyak sekali aspek yang harus diperhatikan. Selain kualitas aplikasi yang akan dibuat, juga ketahanannya terhadap perubahan karena *e-commerce* adalah sesuatu yang sangat cepat berubah karena kompetitor yang sangat kompetitif dan dorongan tehnologi yang membuat efektifitas dan efisiensi menjadi lebih baik.

Dari aspek *software engineering* sendiri, *software engineering* dimaksudkan untuk menunjang/*support* pengembangan *software* daripada *individual programming*. Hal ini mencakup: **a)** *evolution* **b)** *design* **c)** *supporting program specification* (?, ?)

Product characteristics	Description
Maintainability	Software should be written in such a way so that it can evolve to meet the changing needs of customers. This is a critical attribute because software change is an inevitable requirement of a changing business environment.
Dependability and security	Software dependability includes a range of characteristics including reliability, security, and safety. Dependable software should not cause physical or economic damage in the event of system failure. Malicious users should not be able to access or damage the system.
Efficiency	Software should not make wasteful use of system resources such as memory and processor cycles. Efficiency therefore includes responsiveness, processing time, memory utilization, etc.
Acceptability	Software must be acceptable to the type of users for which it is designed. This means that it must be understandable, usable, and compatible with other systems that they use.

Gambar 3.4 *Essential attributes of good software*

Berdasarkan kriteria tersebut, maka setiap poin perlu diperhatikan agar dapat mengembangkan sebuah aplikasi yang tidak hanya sukses, tapi juga bertahan dalam kompetisi. Dalam istilah bisnis, hal ini disebut dengan *risk management & planning*.

3.1.5 Analisa Pribadi Penulis

Dalam pemaparan selanjutnya, penulis akan

3.1.5.1 Analisa *User Experience* dari E-Commerce di Indonesia

Selama masa pengerjaan aplikasi, penulis sering menganalisa dan memperhatikan kebiasaan-kebiasaan yang umum di website *e-commerce* di Indonesia. Salah satu yang paling sering dianalisa oleh penulis adalah adalah situs Tokopedia. Dalam pengembangannya, *user interface* aplikasi akan dipengaruhi analisa ini, yang dijabarkan seperti berikut:

1. Halaman yang muncul bukanlah eagerloading, tapi *lazy loading*
   Ini adalah solusi cerdas untuk mengakali *delay loading item* yang sudah pasti jumlahnya sangat banyak (maka butuh *query* yang tentunya memakan waktu cukup lama), namun juga memainkan faktor psikologi / *user behaviour* pengguna dengan membiarkan pengguna melihat tahap demi tahap halaman 'diisi'.
2. *User Interface* yang sederhana dan pemilihan warna yang *soft*

3.1.5.2 Analisa Keamanan pada koneksi Soket

Untuk mengakomodasi fitur yang bersifat *realtime*, dibutuhkan koneksi ke soket secara terus menerus. Hal ini tentu dapat menjadi sasaran empuk *security* karena jika tidak diamankan, maka dapat menjadi peluang besar bagi para pihak yang tidak berkepentingan untuk merusak proses bisnis aplikasi.

Namun, jika dalam setiap koneksi soket harus mengirimkan *credentials*, hal ini tentu menjadi tidak praktis dan malah lebih berbahaya karena membiarkan data-data sensitif seperti *password* dan *username* berlalu-lalang di jaringan internet.

Selain itu, *disadvantage*nya adalah ketidakpraktisan untuk selalu meng*query* database setiap kali ada koneksi, tentu saja ini memperlambat kerja *database* dan menambah waktu *delay*. Maka dari itu, penulis mengidentifikasi poin-poin penting berikut :

- Hindari *query* database untuk *autentikasi* yang sifatnya masif
- Menggunakan mekanisme authentikasi yang menggunakan *credentials* karena rentan dengan masalah keamanan
- Mencari metode yang lebih efektif, cepat untuk autentikasi selain

3.1.5.3 Analisa *Best Practice* dalam Struktur Perangkat Lunak

Pada dasarnya, Laravel adalah kerangka kerja MVC. Namun, ada banyak fitur yang ada dalam aplikasi Lelang Online ini yang tidak terakomodasi dalam MVC, misal sebagai berikut :

1. Sistem Verifikasi lewat Email - yang berarti aplikasi harus berinteraksi dengan SMTP server
2. Sistem *Generate* Token JWT.io, dimana dalam proses *Generate Token* sama sekali tidak ada database dilibatkan.

Jika fitur-fitur tersebut 'dipaksa' dimuat ke dalam MVC, maka tentu saja strukturnya menjadi ganjil, dan muncul *code smell* berikut :

1. *Large Class*, dimana terdapat satu buah file yang sangat panjang (biasanya merupakan entitas utama, dalam hal ini contohnya barang/*item*)
2. *Inappropriate Intimacy*, dimana terdapat satu kelas yang menyimpan *logic* yang tidak seharusnya ia simpan
3. *Duplicated Code*

Dari hasil analisa ini, penulis mengidentifikasikan strategi-strategi yang akan diterapkan dalam rancangan struktur aplikasi pada subbab **??**, yaitu sebagai berikut:

1. Penggunaan Repository Pattern
   Memisahkan antara Data Processing Layer dan View Layer - agar lebih rapi, terstruktur, hal ini juga dapat menghindari *Duplicated Code*.
2. Penambahan Komponen : Service dan Provider
   Untuk memisahkan *logic* aplikasi yang terkait dengan akses*eksternal services*. Tujuannya, agar jika kedepannya terdapat perbaikan fitur/penambahan fitur, lebih mudah melakukan *traceback* terhadap file/kelas yang bertanggungjawab terhadap fitur tersebut.

3.1.5.4 Analisa Aplikasi Serupa

Selama pengerjaan aplikasi, penulis menganalisa aplikasi serupa. Penulis menemukan aplikasi yang kurang lebih alur bisnis/alur penggunaan aplikasinya serupa yaitu : Carousell. Penulis melihat beberapa kesamaan antara sifat transaksi aplikasi tugas akhir saya dengan aplikasi tersebut, yaitu:

1. Sama-sama tidak mengakomodasi pembayaran
2. Sama-sama tidak adanya kepastian harga (bedanya, pada Carousell yang terjadi adalah *bargaining*

Sehingga dalam alur proses nya, banyak diadaptasi dari Carousell, agar pengguna dapat lebih familiar dan *predictability*nya lebih tinggi jika diadaptasi dari *E-commerce* lainnya yang lebih umum digunakan oleh pengguna.

3.1.5.5 Analisa Aplikasi Serupa

Selama pengerjaan aplikasi, penulis menganalisa aplikasi serupa. Penulis menemukan aplikasi yang kurang lebih alur bisnis / alur penggunaan aplikasinya serupa yaitu : Carousell.

Penulis melihat ada beberapa kesamaan antara sifat transaksi aplikasi tugas akhir saya dengan aplikasi tersebut, yaitu :

1. Sama-sama tidak mengakomodasi pembayaran

2. Sama-sama tidak adanya kepastian harga (bedanya, pada Carousell yang terjadi adalah *bargaining*

Sehingga dalam alur proses nya, banyak diadaptasi dari Carousell, agar pengguna dapat lebih familiar dan *predictability*nya lebih tinggi jika diadaptasi dari *E-commerce* lainnya yang lebih umum digunakan oleh pengguna.

3.1.5.6 Analisa Penyimpanan Data

Untuk penyimpanan data, terdapat 2 jenis data yang sifatnya cukup berbeda, yaitu sebagai berikut:

1. **Data transaksional disimpan di DBMS SQL - *Relational***
   Data yang sifatnya *transaksional*, seperti data *bidding*, data pengguna, dan lain sebagainya. Untuk data ini, lebih baik jika menggunakan database Postgre, untuk menjaga integritas data dan *integrity checking* juga menjadi lebih baik.
2. **Data non-transaksional disimpan di DBMS NoSQL**
   Data *chatting*, data *joined rooms* kurang tepat jika disimpan dalam database transaksional karena sifat pertambahan datanya yang sangat cepat, masif dan urgensi integritas data tidak terlalu diprioritaskan (dibanding dengan data transaksional pada poin sebelumnya). Oleh karena itu, baiknya data ini disimpan pada database NoSQL dengan alasan-alasan sebagai berikut.
   - Banyaknya transaksi *read write*;
   - Ketidaksamaan frekuensi *read and write* data semua pengguna;
   - Sifat permintaan transaksi yang cepat; dan
   - Kemungkinan perubahan struktur atribut pada pesan (misal: *attachments*, *forwarding*, *replying*, dll) akan sangat menyulitkan pengembangan selanjutnya jika menggunakan database transaksional yang terpaku

pada skema database yang ditetapkan di awal pengembangan aplikasi.

3. **Data citra/gambar menggunakan layanan Pihak Ketiga**
   Sekarang telah banyak penyedia jasa *cloud computing* sebagai infrastruktur, seperti Amazon Web Service, Google Cloud Storage Google Alasan-alasan menggunakan AWS sebagai data storage untuk gambar adalah sebagai berikut :
   (a) Skalabilitas aplikasi lebih terjaga.
   Dengan memisahkan penyimpanan antara gambar dan server sehingga lebih mudah me*maintain* perkembangan aplikasi, dan lebih fokus terhadap pengembangan aplikasi.
   (b) Menyediakan *built-in* keamanan, fleksibel dan efisiensi (?, ?)
4. **Optimasi *assets***
   Dalam banyak kesempatan, penulis seringkali mendapati bahwa *delay* untuk *loading assets* lebih lama daripada *loading data* dari database. Berikut penulis akan memaparkan hasil analisa berupa penyebab dan *tackling* permasalahan tersebut.
   (a) *Useless assets* yang disertakan dalam halaman : Memisahkan *essentials assets* dan menyertakan *script* yang hanya digunakan oleh halaman tersebut.
   (b) Logika penyusunan script yang tidak efektif dan optimal (misal: ada satu script yang menyertakan file yang tidak diperlukan) : dilakukan *pre-processing* berupa *minifying, optimization, compiling, compression* terhadap *assets*
   (c) Latensi ke server yang cukup tinggi (misal: kecepatan sambungan internet yang rendah) : *Caching*, *upgrading server* agar dapat ”lebih terjangkau” secara jaringan, penerapan PWA

(*Progressive Web Apps*) untuk sisi *user experience*.

3.1.5.7 Spesifikasi Kebutuhan Fungsional

Berdasarkan deskripsi umum sistem pada subbab **??**, maka dapat disimpulkan bahwa kebutuhan fungsional dari aplikasi lelang online, yang dipaparkan dalam tabel **??**.

Tabel 3.1 Kebutuhan Fungsional Aplikasi Lelang Online

No.	Parameter	Ketersediaan
1	Manajemen Akun	Pengguna: Memperbarui dan mengubah informasi pengguna dalam akun
2	Memanajemen penawaran terhadap barang lelang	Pengguna: Menawar barang, mendapat informasi barang, mencari barang yang diinginkan, dan fitur-fitur yang mempermudah pengguna dalam penawaran barang
3	Memanajemen barang yang dilelang	Pengguna:Mendaftarkan barang untuk dilelang, melihat progress kemajuan lelang, membatalkan lelang dari pengguna tertentu
4	Memanajemen Laporan Pengguna	Pengelola: Melihat daftar laporan dari pengguna, memblokir pengguna yang melanggar ketentuan
Dilanjutkan ke halaman selanjutnya		

Tabel 3.1 – lanjutan dari halaman sebelumnya

No.	Parameter	Ketersediaan
5	Memanajemen Kupon	Pengelola: Membuat kupon, melihat daftar kupon, melihat riwayat penggunaan kupon

3.1.6 Spesifikasi Kebutuhan Non-Fungsional

Kebutuhan non-fungsional yang harus dipeuhi oleh aplikasi ini berhubungan dengan faktor-faktor sebagai berikut:

Tabel 3.2 Kebutuhan Non-Fungsional Aplikasi Lelang Online

No.	Parameter	Ketersediaan
1	Ketersediaan	Aplikasi harus dapat berjalan pada browser, tanpa dibatasi waktu dan tempat selama browser tersebut tersambung ke jaringan internet.
2	Bahasa	Bahasa yang digunakan pada antarmuka merupakan bahasa Indonesia
3	Otorisasi	Setiap pengguna hanya berhak mengakses dan mengubah data yang merupakan milik pengguna tersebut.
3	Performa	Rata-rata waktu akses halaman tidak boleh lebih dari 3 detik.

.. dilanjutkan ke halaman selanjutnya

Tabel 3.2 – lanjutan dari halaman sebelumnya

No.	Parameter	Ketersediaan
4	Portabilitas	Aplikasi dapat diakses pada semua platform selama platform tersebut terinstall web browser
4	*User Experience*	Aplikasi memberikan kesan positif terhadap pengalaman penggnuaan aplikasi
4	*Security*	Koneksi terhadap web harus terlindung *https*.
4	*Flexibility*	Aplikasi haruslah bersifat fleksibel terhadap perubahan (jika terdapat perubahan fitur di masa depan, tidak harus *merefactor* atau mengubah struktur program).

3.1.7 Tugas dan Hak Akses Aktor

Aplikasi lelang online dapat digunakan sebagai wadah bagi para pecinta lelang online untuk melakukan kegiatan lelang atau bagi para penjual atau pembeli yang ingin menjual atau membeli barang yang sesuai baik dari kualitas maupun harga. Identifikasi aktor dalam sistem lelang online dijelaskan dalam tabel **??**.

Tabel 3.3 Identifikasi aktor dalam sistem lelang online

No	Aktor	Keterangan
1	Pengguna	Pengguna yang dimaksud adalah pengguna yang menggunakan fungsionalitas lelang dalam sistem. Pengguna dapat menjadi seorang *bidder* ataupun seorang *auctioneer*, dimana pengguna dapat menjual barang untuk dilelang, dan dapat pula melelang barang
2	*Administrator*	Bertugas melihat dan mengawasi jalannya lelang, melihat laporan keluhan dari pengguna, serta melakukan *block* pengguna jika terdeteksi adanya tindakan yang tidak sesuai dengan kaidah lelang.

Tabel 3.4 Detail Tugas dan Hak Akses

Aktor	**Tugas**	**Hak Akses**	**Kemampuan yang harus dimiliki**
Pengguna	Melakukan aktivitas lelang dan aktivitas interaksi antarpengguna	Manajemen barang & aktivitas lelang dan interaksi antarpengguna.	
Dilanjutkan ke halaman selanjutnya			

Tabel 3.4 – lanjutan dari halaman sebelumnya

Aktor	Tugas	Hak Akses	Kemampuan yang harus dimiliki
Administrator	Bertugas melihat dan mengawasi jalannya lelang, melihat laporan keluhan dari pengguna, serta melakukan *block* pengguna jika terdeteksi adanya tindakan yang tidak sesuai dengan kaidah lelang.	Mengakses halaman-halaman monitoring dari sistem terpisah.	

3.2 Perancangan Sistem

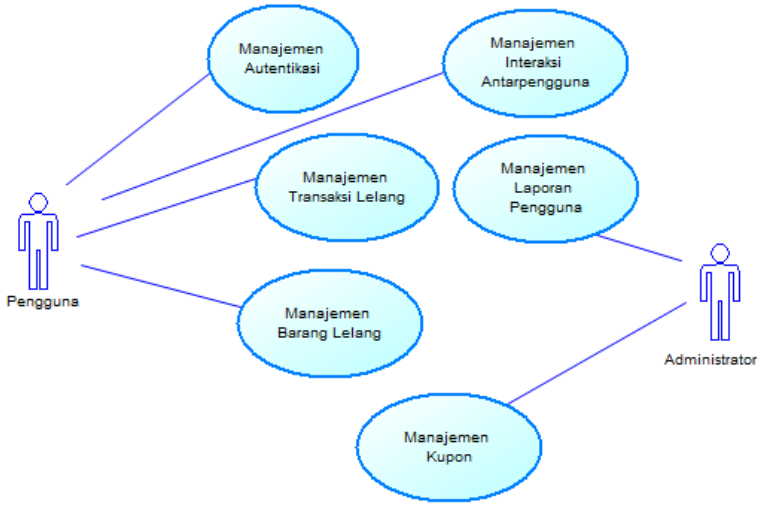
3.2.1 Spesifikasi Kasus Penggunaan

Kasus penggunaan disini dimaksudkan untuk menurunkan kebutuhan fungsional yang telah dispesifikasikan sebelumnya pada tabel **??** sebelumnya.

Daftar kasus Penggunaan dapat dilihat pada **??**.

Tabel 3.5 Tabel Kasus Penggunaan

ID Kasus Penggunaan	Kasus Penggunaan
KP-01	Manajemen Authentikasi Pengguna
KP-02	Memanajemen Transaksi Lelang
KP-03	Manajemen Barang Lelang
KP-04	Manajemen Interaksi Antar Pengguna
KP-05	Monitoring Proses Lelang
KP-06	Manajemen Voucher

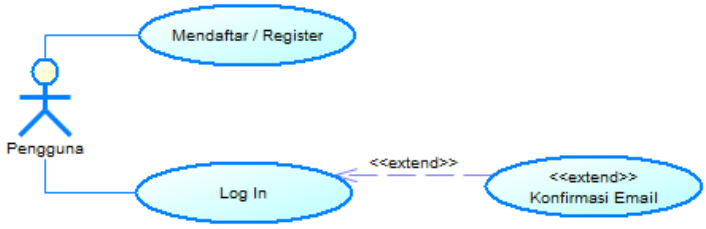


Gambar 3.5 Diagram Kasus Penggunaan Aplikasi

Selanjutnya, akan dijabarkan masing-masing Spesifikasi Kasus Penggunaan untuk semua Kasus Penggunaan yang telah

dijabarkan diatas.

3.2.1.1 KP01. Manajemen Autentikasi Pengguna



Gambar 3.6 Diagram Kasus Penggunaan Manajemen Autentikasi Pengguna

Pada kasus penggunaan ini, pengguna dapat memanajemen autentikasi dan pendaftaran ke dalam sistem.

Tabel 3.6 Tabel Kasus Penggunaan

ID Kasus Penggunaan	Kasus Penggunaan
KP-01	Manajemen Authentikasi Pengguna
KP-02	Memanajemen Transaksi Lelang
KP-03	Manajemen Barang Lelang
KP-04	Manajemen Interaksi Antar Pengguna
KP-05	Monitoring Proses Lelang
KP-06	Manajemen Voucher

Kode	UC-01.01
Nama	**Registrasi**
Aktor	Pengguna
Deskripsi	Pengguna mendaftar ke dalam akun agar masuk ke dalam sistem
Tipe	Fungsional
Precondition	Pengguna belum memiliki akun di aplikasi
Postcondition	Pengguna sudah memiliki akun terdaftar di aplikasi
Alur Kejadian Normal	
	1. Pengguna membuka Halaman Registrasi 2. Sistem menampilkan halaman yang berisi Form Registrasi 3. Pengugna mengisi form tersebut 4. Setelah selesai mengisi, pengugna mengklik tombol ”Registrasi” 5. Sistem memvalidasi data yang dimasukkan pengguna 6. Jika data valid, sistem me*redirect* ke halaman *landing page* dalam keadaan sudah terautentikasi & aun berhasil didaftarkan, dan sistem mengirimkan email konfirmasi email ke alamat email yang didaftarkan.
Alur Kejadian Alternatif	
	Data yang dimasukkan pengguna tidak valid
	??a. Sistem tidak dapat memvalidasi data yang dimasukkan pengguna. **??**b. Sistem me*redirect* ke halaman form registrasi (langkah **??**) dengan *error message*.

Tabel 3.7 Spesifikasi Kasus Penggunaan Registrasi

Kode	UC-01.02
Nama	***Login***
Aktor	Pengguna
Deskripsi	Pengguna melakukan *login* agar dapat masuk ke dalam aplikasi dalam keadaan terauthentikasi.
Tipe	Fungsional
Precondition	Pengguna masuk ke dalam aplikasi dalam keadaan belum terautentikasi
Postcondition	Pengguna masuk ke dalam aplikasi dalam keadaan sudah terautentikasi
Alur Kejadian Normal	
	1. Pengguna membuka Halaman Login 2. Sistem menampilkan halaman Login 3. Pengguna mengisi halaman sesuai *credential* yang dimiliki 4. Setelah selesai mengisi, pengguna mengklik tombol ”Login” 5. Sistem memverifikasi *credential* yang diberikan 6. Jika data benar, sistem me*redirect* ke halaman *landing page* dalam keadaan sudah terautentikasi.
Alur Kejadian Alternatif	
	Data yang dimasukkan pengguna tidak valid
	??a. Sistem tidak dapat memverifikasi *credential* pengguna. **??**b. Sistem me*redirect* ke halaman Login (langkah **??**)dengan *error message*.

Tabel 3.8 Spesifikasi Kasus Penggunaan Login

Kode	UC-01.03
Nama	**Konfirmasi Email**
Aktor	Pengguna
Deskripsi	Pengguna melakukan konfirmasi email agar status akun pengguna menjadi teraktivasi
Tipe	Fungsional
Precondition	Status akun pengguna masih belum terverifikasi
Postcondition	Status akun pengguna masih sudah terverifikasi
Alur Kejadian Normal	
	1. Pengguna membuka halaman *inbox* email pengguna di *sistem email service* yang mereka gunakan. 2. pengguna mencari dan membuka email konfirmasi yang dikirimkan oleh Lelangapa 3. Sistem *email service* pengguna menampilkan isi email konfirmasi, beserta sebuah tombol ”Konfirmasi email” 4. Pengguna mengklik tombol ”Konfirmasi Email” 5. Halaman *browser* akan di*redirect* ke URL konfirmasi email 6. Sistem menampilkan halaman *landing page* dimana status akun pengguna sudah terverifikasi.
Alur Kejadian Alternatif	
	-

Tabel 3.9 Spesifikasi Kasus Penggunaan: Konfirmasi Email

3.2.1.2 KP02. Manajemen Transaksi Lelang

Gambar 3.7 Diagram Kasus Penggunaan Manajemen Traksaksi Lelang

Pada kasus penggunaan ini, pengguna akan dapat memanajemen transaksi dan penawaran-penawaran yang ia berikan terhadap barang yang terdaftar dalam alikasi.

Kode	UC-02.01
Nama	**Melihat daftar barang yang dilelang**
Aktor	Pengguna
Deskripsi	Pengguna melihat daftar barang yang sedang dilelang
Tipe	Fungsional
Pre Condition	Sistem belum menampilkan daftar barang yang sedang dilelang
Post Condition	Sistem menampilkan daftar barang yang sedang dilelang
Alur Kejadian Normal	
	1. Pengguna mengklik *icon* aplikasi di kiri atas halaman 2. Sistem menampilkan halaman depan yang berisi daftar barang yang sedang dilelang *Ket : Pada halaman depan, ditampilkan barang sesuai dengan kategori berdasarkan waktu dan popularitas, seperti Hot Item (barang yang paling ramai transaksi bidnya), Newest Item, dll.*
Alur Kejadian Alternatif	
	-

Tabel 3.10 Spesifikasi Kasus Penggunaan : Melihat barang yang dilelang

Kode	UC-02.02
Nama	**Mencari Barang Lelang**
Aktor	Pengguna
Deskripsi	Pengguna ingin mencari barang lelang dengan kriteria nama tertentu
Tipe	Fungsional
Pre Condition	Pengguna menemukan barang lelang yang ia cari dengan kriteria *string*.
Post Condition	Pengguna menemukan barang lelang yang ia cari dengan kriteria *string* masukan.
Alur Kejadian Normal	
	1. Pengguna memasukkan kriteria *string* pencarian di *field* masukan di *Header Bar* 2. Setelah selesai, pengguna mengklik tombol ”Cari” 3. Sistem mencari barang terdaftar yang sesuai dengan kriteria masukan pengguna 4. Jika ketemu, sistem menampilkan halaman ”Hasil Pencarian” beserta barang yang sesuai dengan kriteria pengguna. 5. Pengguna lalu mengklik barang yang sesuai dengan keinginan 6. Sistem menampilkan detail barang yang sesuai dengan keinginan pengguna
Alur Kejadian Alternatif	
	Tidak ada barang terdaftar dalam sistem yang sesuai dengan kriteria pengguna.
	?? a. Sistem tidak dapat menemukan barang yang sesuai **??** b. Sistem menampilkan ”Hasil Pencarian” namun dengan keterangan ”Hasil pencarian kosong”

Tabel 3.11 Spesifikasi Kasus Penggunaan : Mencari Barang Lelang

Kode	UC-02.03
Nama	**Mencari Barang Lelang**
Aktor	Pengguna
Deskripsi	Pengguna ingin mencari barang lelang dengan kriteria nama tertentu
Tipe	Fungsional
Pre Condition	Pengguna menemukan barang lelang yang ia cari dengan kriteria *string*.
Post Condition	Pengguna menemukan barang lelang yang ia cari dengan kriteria *string* masukan.
Alur Kejadian Normal	
	1. Pengguna memasukkan kriteria *string* pencarian di *field* masukan di *Header Bar* 2. Setelah selesai, pengguna mengklik tombol ”Cari” 3. Sistem mencari barang terdaftar yang sesuai dengan kriteria masukan pengguna 4. Jika ketemu, sistem menampilkan halaman ”Hasil Pencarian” beserta barang yang sesuai dengan kriteria pengguna. 5. Pengguna lalu mengklik barang yang sesuai dengan keinginan 6. Sistem menampilkan detail barang yang sesuai dengan keinginan pengguna
Alur Kejadian Alternatif	
	Tidak ada barang terdaftar dalam sistem yang sesuai dengan kriteria pengguna.
	?? a. Sistem tidak dapat menemukan barang yang sesuai **??** b. Sistem menampilkan ”Hasil Pencarian” namun dengan keterangan ”Hasil pencarian kosong”

Tabel 3.12 Spesifikasi Kasus Penggunaan : Mencari Barang Lelang

Kode	UC-02.04
Nama	**Melihat 3 Riwayat Penawaran Lelang Barang Teratas**
Aktor	Pengguna
Deskripsi	Pengguna ingin melihat riwayat penawaran lelang terhadap barang yang ia daftarkan
Tipe	Fungsional
Pre Condition	Pengguna menemukan barang lelang yang ia cari dengan kriteria *string*.
Post Condition	Pengguna menemukan barang lelang yang ia cari dengan kriteria *string* masukan.
Alur Kejadian Normal	
	1. Pengguna membuka halaman ”Kelola Barang” 2. Pengguna mengklik barang yang ingin dilihat informasi riwayat penawaran lelangnya 3. Sistem menampilkan halaman informasi barang tersebut 4. Pengguna mengklik tombol ”Lihat Penawaran Teratas” 5. Sistem menampilkan *modal* berisi 3 riwayat penawaran lelang teratas barang tersebut.
Alur Kejadian Alternatif	
	-

Tabel 3.13 Spesifikasi Kasus Penggunaan : Melihat Riwayat Penawaran Lelang Barang

3.2.1.3 KP03. Manajemen Barang Lelang

Gambar 3.8 Diagram Kasus Penggunaan Manajemen Barang Lelang

Pada kasus penggunaan ini, pengguna akan dapat memanajemen barang yang ia daftarkan untuk dilelang, dan melihat proses monitoringnya, seperti yang dipaparkan pada penjelasan berikut.

Kode	UC-03.01
Nama	**Mendaftarkan Barang Lelang**
Aktor	Pengguna
Deskripsi	Pengguna mendaftarkan barang untuk dilelang di dalam sistem
Tipe	Fungsional
Precondition	Barang yang akan dilelang belum terdaftar dalam sistem
Postcondition	Barang yang akan dilelang sudah terdaftar dalam sistem
Alur Kejadian Normal	
	1. Pengguna dalam keadaan terautentikasi, mengklik ”Item Anda” -> ”Add Items” pada *navbar* bagian atas halaman. 2. Sistem menampilkan halaman yang berisi form pendaftaran barang 3. Pengguna mengisi form tersebut sesuai data barang 4. Setelah selesai mengisi, pengugna mengklik tombol ”Daftar Barang” 5. Sistem memvalidasi data yang dimasukkan pengguna 6. Jika data valid, sistem me*redirect* ke halaman ”Kelola Barang” dalam keadaan barang baru sudah ditambahkan.
Alur Kejadian Alternatif	
	Data barang yang dimasukkan pengguna tidak valid
	??a. Sistem tidak dapat memvalidasi data yang dimasukkan pengguna. **??**b. Sistem me*redirect* ke halaman form ”Tambah Barang” (langkah **??**) dengan *error message*.

Tabel 3.14 Spesifikasi Kasus Penggunaan : Mendaftarkan Barang Lelang

Kode	UC-03.02
Nama	**Memperbarui informasi barang yang dilelang**
Aktor	Pengguna
Deskripsi	Pengguna memperbarui informasi barang yang sebelumnya sudah terdaftar di dalam sistem
Tipe	Fungsional
Precondition	Informasi barang belum diperbarui.
Postcondition	Informasi barang sudah diperbarui.
Alur Kejadian Normal	
	1. Pengguna dalam keadaan terautentikasi, mengklik ”Item Anda” -> ”Manage Items” pada *navbar* bagian atas halaman. 2. Sistem menampilkan halaman yang berisi daftar barang yang didaftarkan pengguna. 3. Pengguna mengklik barang yang ingin diperbarui informasinya 4. Sistem menampilkan halaman *form* ”Perbarui barang”. 5. Pengguna mengisi informasi pembaruan barang di dalam form tersebut. 6. Setelah selesai, pengguna mengklik tombol ”Simpan Pembaruan”. 7. Sistem memvalidasi data (termasuk file gambar) yang dimasukkan pengguna 8. Jika data valid, sistem me*redirect* ke halaman ”Kelola Barang” dalam keadaan barang baru sudah ditambahkan.
Alur Kejadian Alternatif	
	Data barang yang dimasukkan pengguna tidak valid
	??a. Sistem tidak dapat memvalidasi data yang dimasukkan pengguna. **??**b. Sistem me*redirect* ke halaman ”Perbarui Barang” (langkah **??**) dengan *error message*.
	Gambar yang dimasukkan pengguna tidak

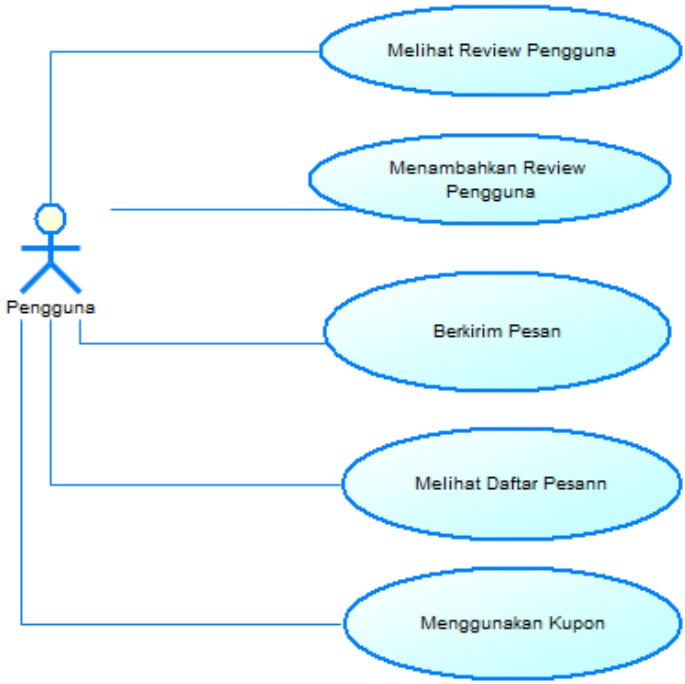
Kode	UC-03.03
Nama	**Melihat Daftar Barang yang Pernah Dilelang**
Aktor	Pengguna
Deskripsi	Pengguna hendak melihat daftar semua barang yang pernah didaftarkan untuk dilelang di dalam sistem.
Tipe	Fungsional
Precondition	Informasi daftar barang belum ditampilkan.
Postcondition	Informasi daftar barang sudah ditampilkan.
Alur Kejadian Normal	
	1. Pengguna dalam keadaan terautentikasi, mengklik ”Item Anda” -> ”Manage Items” pada *navbar* bagian atas halaman. 2. Sistem menampilkan halaman yang berisi daftar barang yang didaftarkan pengguna.
Alur Kejadian Alternatif	
	-

Tabel 3.16 Spesifikasi Kasus Penggunaan : Melihat Barang yang Pernah Didaftarkan

Kode	UC-03.04
Nama	**Melihat Detail Riwayat Penawaran Harga**
Aktor	Pengguna
Deskripsi	Pengguna hendak melihat daftar semua barang yang pernah didaftarkan dalam sistem.
Tipe	Fungsional
Precondition	Informasi daftar barang belum ditampilkan.
Postcondition	Informasi daftar barang sudah ditampilkan.
Alur Kejadian Normal	
	1. Pengguna dalam keadaan terautentikasi, mengklik ”Item Anda” -> ”Manage Items” pada *navbar* bagian atas halaman. 2. Sistem menampilkan halaman yang berisi daftar barang yang didaftarkan pengguna. 3. Pengguna mengklik barang yang ingin dilihat daftar penawaran harganya 4. Sistem menampilkan halaman detail informasi barang 5. Pengguna mengklik tombol ”Lihat Riwayat Penawaran” 6. Sistem menampilkan halaman berisi daftar riwayat penawaran.
Alur Kejadian Alternatif	
	-

Tabel 3.17 Spesifikasi Kasus Penggunaan : Melihat Riwayat Penawaran Harga

3.2.1.4 KP04. Manajemen Interaksi Antarpengguna



Gambar 3.9 Diagram Kasus Penggunaan Manajemen Interaksi Antarpengguna

Pada kasus penggunaan ini, pengguna difasilitasi untuk berinteraksi, memberikan *review* / testimoni terhadap pengguna lainnya sesuai dengan keinginan.

Kode	UC-04.01
Nama	**Melihat Review Pengguna**
Aktor	Pengguna
Deskripsi	Pengguna ingin melihat review pada pengguna tertentu
Tipe	Fungsional
Precondition	Review pengguna belum ditampilkan
Postcondition	Review pengguna berhasil ditampilkan
Alur Kejadian Normal	
	1. Pengguna mengklik *link* profil pengguna 2. Sistem menampilkan halaman profil pengguna 3. Pengguna dapat melihat *review* pengguna di bagian kiri bawah beserta rata-rata *rating* yang diberikan.
Alur Kejadian Alternatif	
	-

Tabel 3.18 Spesifikasi Kasus Penggunaan : Melihat Review Pengguna

Kode	UC-04.02
Nama	**Menambahkan Review Pengguna**
Aktor	Pengguna
Deskripsi	Pengguna ingin menambahkan review dari *transaksi* yang pernah dilakukan.
Tipe	Fungsional
Precondition	Review dari pengguna belum tercatat/tersimpan dalam sistem
Postcondition	Review dari pengguna berhasil tercatat dalam sistem
Alur Kejadian Normal	
	1. Pengguna mengklik halaman ’Riwayat Transaksi’ 2. Sistem menampilkan halaman Riwayat Transaksi yang pernah dilakukan pengguna 3. Pengguna mengklik *tab* jenis transaksi yang pernah dilakukan (Beli atau Lelang) 4. Sistem menampilkan riwayat transaksi sesuai dengan jenis transaksi yang dipilih pengguna 5. Pengguna mengklik transaksi yang ingin diberikan *review* 6. Sistem mengecek apakah *review* sudah pernah diberikan sebelumnya 7. Sistem menampilkan *modal* berisi *field input* jumlah *rating* 8. Pengguna mengisi *field* tersebut sesuai jumlah rating yang ingin diberikan 9. Setelah selesai, pengguna klik ’Next’ 10. Sistem menampilkan *modal* kedua, berisikan *field input* untuk deskripsi *review* 11. Pengguna mengisikan *field input* sesuai dengan deskripsi yang ingin diberikan 12. Setelah selesai, pengguna mengklik tombol ’Simpan Review’ 13. Sistem memvalidasi masukan dari pengguna 14. Jika tervalidasi, sistem menampilkan modal berisi informasi sukses menyimpan review

Kode	UC-04.03
Nama	**Melaporkan Barang**
Aktor	Pengguna
Deskripsi	Pengguna ingin melaporkan barang yang dianggap melanggar aturan/tidak pantas diperjualbelikan
Tipe	Fungsional
Precondition	Laporan dari pengguna belum tersimpan dalam sistem
Postcondition	Laporan dari pengguna berhasil tersimpan dalam sistem
Alur Kejadian Normal	
	1. Pengguna mengklik barang yang ingin dilaporkan 2. Sistem menampilkan halaman informasi barang 3. Pengguna mengklik tombol ”Laporkan Barang” 4. Sistem menampilkan *modal* berisi *input field* laporan 5. Pengguna mengisi *fields* tersebut sesuai dengan konten laporan yang ingin disampaikan 6. Setelah selesai, pengguna mengklik tombol ”Laporkan” 7. Sistem mengecek dan memvalidasi masukan pengguna 8. Jika valid, sistem akan menampilkan *modal* Sukses Menyimpan Laporan 9. Sistem me*redirect* pengguna kembali ke halaman di **??**
Alur Kejadian Alternatif	
	Data masukan laporan pengguna tidak valid
	??a. Sistem mendeteksi masukan pengguna tidak valid. **??**c. Sistem menampilkan kembali modal di poin **??** beserta dengan *error message*

Kode	UC-04.04
Nama	**Mengirim Pesan**
Aktor	Pengguna
Deskripsi	Pengguna akan mengirimkan pesan kepada pengguna lainnya
Tipe	Fungsional
Precondition	Pesan yang dikirimkan pengguna belum tersimpan pada sistem
Postcondition	Pesan yang dikirimkan pengguna berhasil tersimpan pada sistem
Alur Kejadian Normal	
	1. Pengguna mengklik *URL* pengguna tujuan yang ingin dikirimi pesan 2. Sistem menampilkan halaman profil pengguna tujuan 3. Pengguna mengklik tombol ”Kirim Pesan” 4. Sistem menampilkan halaman percakapan pengguna terhadap tujuan beserta riwayat percakapan pengguna dengan pengguna tujuan 5. Pengguna memasukkan pesan yang ingin dikirimkan pada *field input* yang disediakan 6. Setelah selesai, pengguna mengklik tombol ’Kirim’ 7. Sistem mengirim kepada koneksi soket 8. Jika proses pengiriman kepada soket berhasil dan tidak ada gangguan, sistem kembali menampilkan halaman pengguna dengan informasi pesan yang sudah terkirim muncul di riwayat percakapan pengguna dengan pengguna tujuan
Alur Kejadian Alternatif	
	Terjadi masalah teknis sehingga pesan tidak dapat terkirim
	??a. Sistem mendapatkan *exception* dari koneksi soket, bahwa pesan tidak dapat tersimpan

Kode	UC-04.05
Nama	**Melihat dan Membaca Pesan**
Aktor	Pengguna
Deskripsi	Pengguna ingin melihat daftar percakapan/ daftar perpesanan yang pernah dilakukan pengguna
Tipe	Fungsional
Precondition	Daftar percakapan/ daftar perpesanan belum ditampilkan
Postcondition	Daftar percakapan/ daftar perpesanan berhasil ditampilkan
Alur Kejadian Normal	
	1. Pengguna mengklik tombol ’Conversations’ di *navbar* aplikasi 2. Sistem menampilkan halaman daftar percakapan pengguna 3. Sistem memanggil fungsi AJAX untuk meminta data-data percakapan terakhir pengguna 4. Balasan dari fungsi AJAX di *load* oleh *browser* untuk selanjutnya di*parse* ke dalam HTML 5. Sistem menampilkan daftar percakapan pengguna 6. Pengguna mengklik percakapan yang ingin dilihat/dibaca 7. Sistem menampilkan etail percakapan pengguna dengan pengguna tujuan
Alur Kejadian Alternatif	
	Koneksi terhadap soket *chat* tidak dapat dibangun/bermasalah
	??a. Sistem mendapatkan *exception* dari fungsi AJAX, bahwa daftar perpesanan/percakapan tidak dapat di*load* **??**b. Sistem menampilkan kembali halaman pada poin **??**, dengan *modal* berisikan *error message*

Kode	UC-04.06
Nama	**Memasukkan Kupon pada Transaksi**
Aktor	Pengguna
Deskripsi	Pengguna ingin menggunakan kupon/voucher yang ia miliki untuk pada sebuah transaksi
Tipe	Fungsional
Precondition	Pengguna belum berhasil men*submit* kode kupon ke dalam transaksi barang
Postcondition	Pengguna berhasil men*submit* kode kupon ke dalam transaksi barang
Alur Kejadian Normal	
	1. Pengguna membuka halaman ’Riwayat Transaksi Lelang’ 2. Sistem menampilkan halaman Riwayat Transaksi Lelang pengguna 3. Pengguna mengklik tombol ’Masukkan Kupon’ pada transaksi yang diinginkan 4. Sistem mengecek permintaan penggunaan kupon 5. Jika permintaan dapat diverifikasi dan valid, sistem menampilkan *modal* berisi *input field* kupon 6. Pengguna memasukkan kupon yang ingin dimasukkan, lalu mengklik tombol ’Submit’ 7. Sistem memvalidasi kupon voucher dan status barang 8. Jika valid, sistem menerapkan penggunaan kupon ke dalam transaksi barang 9. Sistem lalu menampilkan *modal* yang berisi informasi sukses penggunaan kupon pada transaksi
Alur Kejadian Alternatif	
	-

Tabel 3.23 Spesifikasi Kasus Penggunaan : Mendaftarkan Barang Lelang

3.2.1.5 KP05. Manajemen Voucher

Gambar 3.10 Diagram Kasus Penggunaan Manajemen Kupon

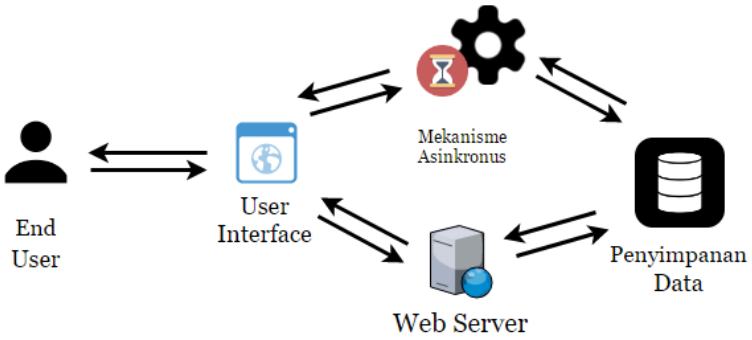
Kasus penggunaan ini seluruhnya digunakan oleh *administrator* aplikasi dan dilakukan di sistem terpisah. Kasus penggunaan ditujukan untuk mempermudah *administrator* dalam memanajemen voucher/kupon yang dibagikan oleh pengguna.

3.2.2 Identifikasi Komponen Fundamental

Berdasarkan Bab Analisa, dapat diidentifikasi dan divisualisasikan (pada gambar **??**) komponen-komponen penting dalam pembuatan aplikasi, yaitu sebagai berikut:

1. Web Server
2. Mekanisme penyimpanan data (*database* dan *data storage*)
3. *User Interface* sebagai media terhadap *end-user*
4. Mekanisme Asinkronus untuk mengakomodasi fitur *realtime*

5. Verifikasi, Autentikasi dan Otorisasi (Keamanan) dalam implementasi program
6. Implementasi *User Behavior* dan HCI



Gambar 3.11 Arsitektur dasar yang dibutuhkan untuk membangun aplikasi

3.2.3 *Technology Options*

Pada subbab sebelumnya, penulis sudah memaparkan arsitektur dasar yang dibutuhkan dalam rancang bangun aplikasi. Terkait dengan arsitektur dasar tersebut, banyak pilihan teknologi yang dapat mengimplementasikan arsitektur tersebut. Dalam pemaparan selanjutnya, akan dijelaskan alasan penulis *best practice* dalam pemilihan teknologi yang digunakan, didasarkan pada *best practices* dan pengalaman-pengalaman penulis. Keterkaitan dengan aspek-aspek yang dijelaskan sebelumnya pada subbab **??**.

3.2.3.1 NGINX sebagai Web Server

- Kelebihan
  1. Konfigurasi yang lebih *friendly* dan terstruktur
  2. Ketersediaan fitur yang lengkap & krusial (*reverse proxy*, memungkinkan skalabilitas & *load balancing*)

3. *Learning-gap* yang kecil terhadap pengalaman penulis/sudah familiar

- Opsi lainnya
  1. Apache2: Fiturnya kurang lengkap
  2. Node.js: *Learning-gap* yang besar bagi penulis/belum familiar
  3. Python: Belum Familiar, dan perlu eksplorasi fitur lebih dalam

3.2.3.2 PostgreSQL untuk Penyimpanan Data

- Kelebihan
  1. *Learning gap* yang kecil
  2. Stabil karena telah digunakan dan dikembangkan oleh banyak *developer* selama bertahun-tahun
- Opsi lainnya
  1. SQL Server: Instalasi yang kompleks, penggunaan *resource* yang cukup besar

3.2.3.3 MongoDB untuk Penyimpanan Data Nontansaksional

- Kelebihan
  1. *Learning curve* yang mudah/sintaksnya kurang lebih sama dengan sintaks *database* transaksional pada umumnya
  2. Performa yang cepat karena menggunakan BSON
  3. Fitur yang lengkap untuk *sustainability* aplikasi seperti (Replikasi, Sharding, dll)
  4. *Handling* terhadap data yang sangat besar yang cukup bagus, cocok untuk data yang masif seperti *chatting*.
- Opsi lainnya
  1. Redis: Cepat, namun penyimpanan dilakukan di RAM sehingga lebih cocok untuk penyimpanan *auth session*,

bukan untuk penyimpanan data yang sifatnya masif
2. Cassandra: *Learning-gap* yang besar, namun fiturnya lengkap untuk *data mining*

3.2.3.4 CDN sebagai *Assets Sources*

- Kelebihan
  1. Akses cepat karena besar kemungkinan asset tersebut telah di*cache* sebelumnya dalam browser pengguna
  2. Mengurangi *bandwith* server
  3. Telah dioptimasi oleh pengembang masing-masing asset.
- Opsi lainnya
  1. Disimpan dalam server: Mengurangi *bandwith* server (*cost* meningkat)

3.2.3.5 AWS S3 untuk *Content Growth Scalability*

- Kelebihan
  1. *Benefit* yang sangat *krusial*: keamanan, skalabilitas, *availability* - karena sudah di*handle* langsung oleh pengembang *cloud computing* yang ahli di bidangnya
  2. Perkembangan jumlah konten yang akan disimpan (gambar barang yang diupload pengguna) tentunya bersifat sangat masif, sehingga tidak mungkin disimpan dalam server
- Opsi lainnya
  1. Disimpan dalam server: Mengurangi performa server karena sifatnya yang memakan *resource* cukup banyak, dan menambah *cost* untuk *upgrade server storage*

3.2.3.6 SendGrid untuk SMTP Relay

- Kelebihan

1. Konfigurasi yang mudah
2. Dokumentasi yang cukup lengkap dan mudah ditemukan
3. Fitur yang lengkap
4. Adanya *free storage* dari akun Github Student Pack penulis

- Opsi lainnya
  1. MailChimp: Dokumentasi kurang lengkap, tidak ada *free storage* untuk akun penulis

3.2.3.7 VUE.JS untuk *Workloads Sharing*

- Kelebihan
  1. *Learning-gap* relatif kecil dibandingkan *Javascript tools* lainnya, karena didesain khusus untuk Laravel
  2. Adanya program utilitas (webpack) yang membuat performa Vue.js jauh lebih cepat
  3. Logika aplikasi dapat di*obfuscate* dengan webpack (*embedded* dalam Laravel)
- Opsi lainnya
  1. React: *Learning gap* dan *learning curve* yang sangat besar untuk penulis
  2. jQuery: tidak efektif karena *code smells* yang ditimbulkan cukup banyak

3.2.3.8 SOCKET.IO untuk Mekanisme Asinkronus

- Kelebihan
  1. *Learning-gap* relatif kecil dibandingkan *Javascript tools* lainnya, karena didesain khusus untuk Laravel
  2. Adanya program utilitas (webpack) yang membuat performa Vue.js jauh lebih cepat
  3. Logika aplikasi dapat di*obfuscate* dengan webpack (*embedded* dalam Laravel)

- Opsi lainnya
    1. React: *Learning gap* dan *learning curve* yang sangat besar untuk penulis
    2. jQuery: tidak efektif karena *code smells* yang ditimbulkan cukup banyak

3.2.3.9 JWT untuk Keamanan Soket

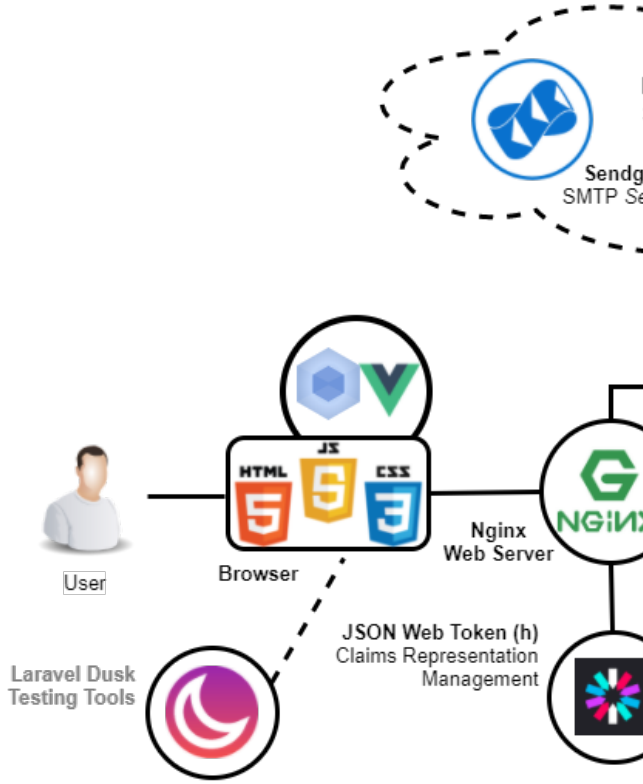
- Kelebihan
    1. *Library support* yang lengkap untuk komponen-komponen lainnya
    2. Efektif dan efisien karena tidak ada *query* ke database untuk autentikasi
- Opsi lainnya
    1. *Query* ke *database* secara konvensional: Sifat koneksi soket yang masif akan sangat memberatkan *database* jika setiap kali ada koneksi baru, harus melakukan *query database* sehingga tidak efektif
    2. *Session caching* dengan Redis: *Learning gap* yang besar

3.2.3.10 Laravel Dusk untuk *Functionality Testing Script*

- Kelebihan
    1. *Learning gap* yang kecil karena didesain sefamiliar mungkin dengan Laravel
- Opsi lainnya
    1. Selenium: *Learning curve* yang besar
    2. Phantom.js: *Learning curve* yang besar

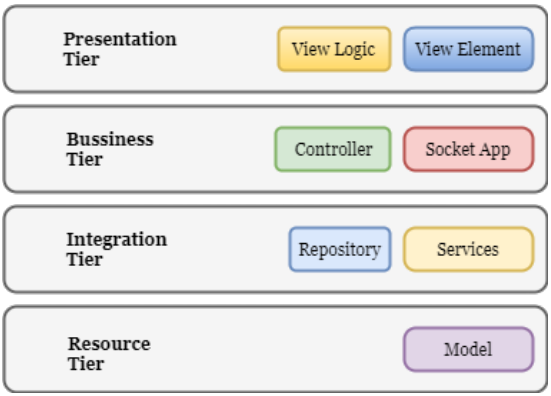
3.2.4 Arsitektur Perangkat Lunak

Arsitektur perangkat lunak dalam pengembangan aplikasi lelang online ini digambarkan pada diagram arsitektur pada gambar **??**



Gambar 3.12 *Visualisasi arsitektur dan teknologi Final yang diterapkan dalam rancang bangun aplikasi*

3.2.5 Struktur Aplikasi



Gambar 3.13 Komponen Penyusun Struktur Aplikasi

3.2.5.1 *Presentation Tier*

Presentation tier adalah gabungan komponen yang bertanggung jawab terhadap *user interface* ke secara keseluruhan. Dalam *tier* ini menjadi 2 komponen, yaitu:

1. **Komponen *View Element***
   Komponen *view* adalah struktur yang bertanggungjawab terhadap dari elemen-elemen tampilan (HTML, CSS, JS).
2. **Komponen *View Logic***
   Komponen *view logic* adalah struktur yang secara khusus bertanggungjawab terhadap logika *view*, misal pada halaman melelang barang, maka *view logic* adalah logika program saat pengguna memasukkan penawaran harga, mengklik tombol ”Tawar” hingga akhirnya penawaran tersebut diterima/diproses oleh aplikasi.

3.2.5.2 *Bussiness Tier*

Bussiness tier adalah gabungan komponen yang bertanggung jawab kelangsungan proses bisnis. Dalam *tier* ini, terdapat 2 komponen yaitu:

1. **Komponen *Controller***
   Komponen *controller* adalah struktur yang bertanggungjawab logika/proses bisnis aplikasi secara keseluruhan. *Controller* mengatur autentikasi, otorisasi, penerusan dan pemrosesan *request* dari pengguna kepada komponen pemroses seperti *data management layer* atau *external service layer*, lalu mengembalikan hasilnya kepada pengguna.
2. **Komponen *Socket App*** Komponen ini ditulis dengan menggunakan Node.js, dan komponen ini bertanggungjawab penuh untuk menyediakan *realtime connection* pada fitur berkirim pesan dan proses *bidding*.

3.2.5.3 *Integration Tier*

Integration tier adalah gabungan komponen yang bertanggung jawab integrasi pemrosesan data dan *external service usages*. Dalam *tier* ini, terdapat komponen-komponen berikut:

1. **Komponen *Repository***
   Komponen *repository* adalah komponen yang bertanggungjawab terhadap *data management*, dimana *controller* hanya bertugas meneruskan dan mengautorisasi *request*, maka *repository* bertanggungjawab untuk mengolah data tersebut sesuai permintaan. Lalu, *repository* bertanggungjawab meneruskan status proses (jika sifatnya keberhasilan, misal: *create* data pengguna), atau meneruskan hasil proses (misal: *request* halaman tampilkan barang, maka *repository* bertanggungjawab

meneruskan hasil *query* kepada *controller*).

2. **Komponen *Services*** Komponen ini dan bertanggungjawab penuh terhadap akses aplikasi ke *external service* seperti *SMTP Relay* yang menggunakan SendGrid.

3.2.5.4 *Integration Tier*

Integration tier adalah gabungan komponen yang bertanggungjawab integrasi pemrosesan data dan *external service usages*. Dalam *tier* ini, terdapat komponen-komponen berikut:

1. **Komponen *Repository***
   Komponen *repository* adalah komponen yang bertanggungjawab terhadap *data management*, dimana *controller* hanya bertugas meneruskan dan mengautorisasi *request*, maka *repository* bertanggungjawab untuk mengolah data tersebut sesuai permintaan. Lalu, *repository* bertanggungjawab meneruskan status proses (jika sifatnya keberhasilan, misal: *create* data pengguna), atau meneruskan hasil proses (misal: *request* halaman tampilkan barang, maka *repository* bertanggungjawab meneruskan hasil *query* kepada *controller*).

2. **Komponen *Services*** Komponen ini dan bertanggungjawab penuh terhadap akses aplikasi ke *external service* seperti *SMTP Relay* yang menggunakan SendGrid.

3.2.5.5 *Resource/Data Access Tier*

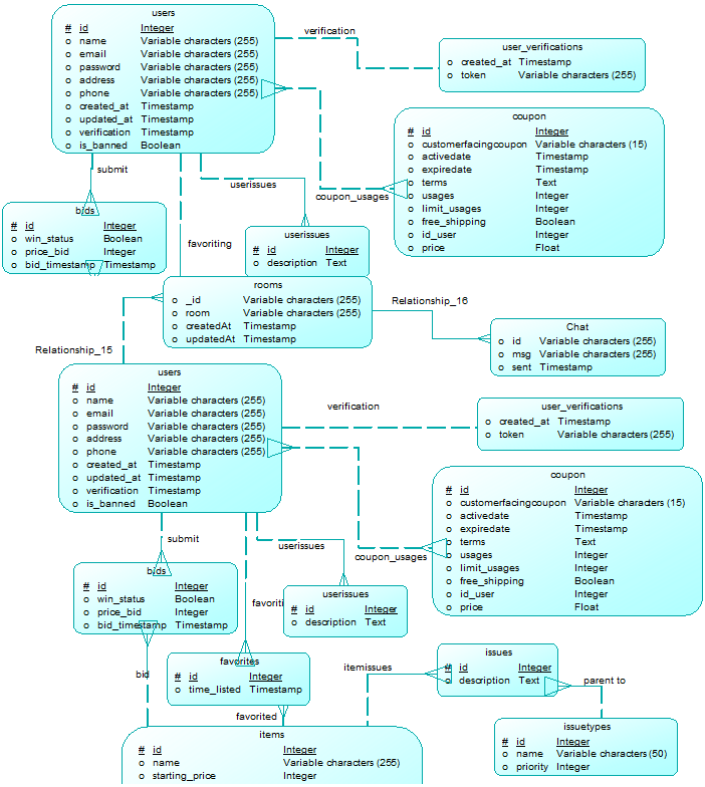
Resource tier adalah gabungan komponen yang bertanggungjawab terhadap akses ke *database* dan *storage devices*, yaitu pengaksesan dan *preprocessing* data menjadi *suitable format* bagi komponen lainnya.

1. **Komponen *Model***
   Komponen *model* adalah komponen yang bertanggungjawab terhadap sebuah tabel/entitas dalam *database* aplikasi. Tidak hanya itu, *model* juga bertanggungjawab terhadap *preprocessing* informasi, yaitu pengubahan data dari *database* menjadi *collection*/Eloquent ORM yang merupakan *suitable format* bagi pemrosesan data di tier *repository* maupun *bussiness*.

3.2.6 Perancangan *Database*

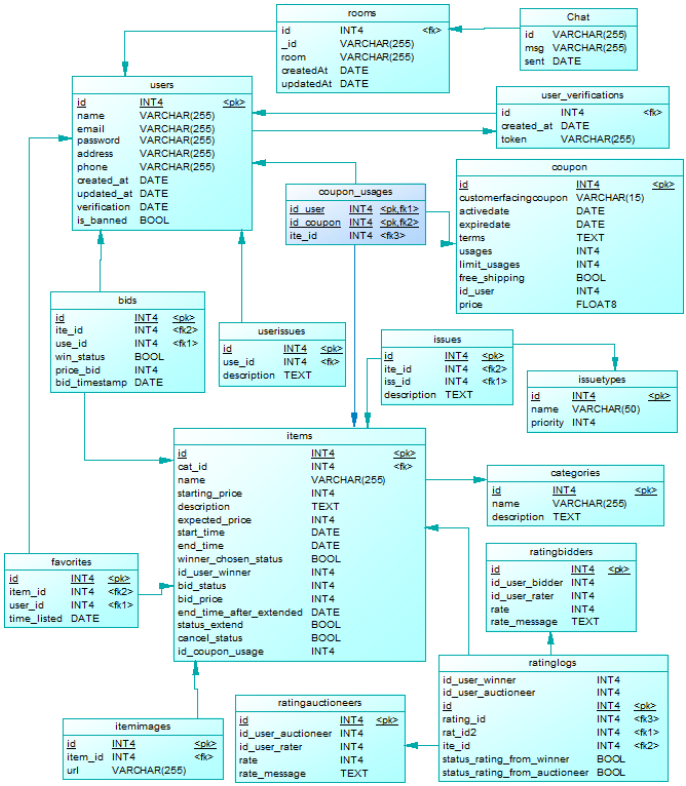
Pada subbab ini akan dijelaskan bagaimana rancangan basis data yang digunakan pada aplikasi lelang online ini. Sistem basis data yang digunakan pada aplikasi ini menggunakan dua jenis database, yaitu transaksional (menggunakan PostgreSQL) dan nontransaksional (menggunakan MongoDB). Conceptual Data Model (CDM) dan Physical data model (PDM) dari basis data sistem ini dijelaskan pada gambar

3.2.6.1 *Conceptual Database Model*



Gambar 3.14 *Conceptual Database Model* (PDM) Aplikasi

3.2.6.2 *Physical Database Model*



Gambar 3.15 *Physical Database Model* (PDM) Aplikasi

3.2.7 Kamus Data *Database* Transaksional

Subbab ini akan berisi pemaparan detail tentang kamus data dalam *database* yang bersifat transaksional. Keseluruhan terdapat 15 tabel.

3.2.7.1 Kamus data tabel *administrator*

Tabel 3.24 Spesifikasi Tabel Administrator

No	Nama Atribut	Tipe Data	Keterangan
1	id	int	PK , Auto increment
2	name	varchar(255)	Nama
3	username	varchar(255)	Username yang digunakan
4	password	varchar(255)	Password
5	created_at	timestamp	Timestamp default laravel
6	updated_at	timestamp	Timestamp default laravel
7	email	varchar(255)	Email

3.2.7.2 Kamus data tabel *bids*

Tabel 3.25 Kamus Data Tabel *bids*

No	Nama Atribut	Tipe Data	Keterangan
1	id	bigint	PK , Auto increment
2	id_user	int	FK ID user yang memberikan penawaran
3	id_item	bigint	FK ID barang yang ditawar
4	price_bid	bigint	Harga yang ditawarkan

Dilanjutkan ke halaman selanjutnya

Tabel 3.25 – lanjutan dari halaman sebelumnya

No	Nama Atribut	Tipe Data	Keterangan
5	win_status	bool	Status apakah bid ini memenangkan lelang
6	bid_timestamp	timestamp	Timestamp

3.2.7.3 Kamus data tabel *categories*

Tabel 3.26 Kamus Data Tabel *categories*

No	Nama Atribut	Tipe Data	Keterangan
1	id	bigint	PK, AutoIncrement
2	name	int	Nama Kategori
3	description	bigint	Deskripsi Kategori

3.2.7.4 Kamus data tabel *coupon*

Tabel 3.27 Kamus Data Tabel *coupon*

No	Nama Atribut	Tipe Data	Keterangan
1	coupon_id	int	PK, AutoIncr
2	customerfacing coupon	varchar(15)	String voucher yang diberikan kepada pengguna
3	activedate	timestamp	Tanggal aktif voucher
4	expiredate	timestamp	Tanggal nonaktif voucher

Tabel 3.27 – lanjutan dari halaman sebelumnya

No	Nama Atribut	Tipe Data	Keterangan
5	terms	varchar(255)	Deskripsi aturan penggunaan voucher
6	usages	bigint	Banyak penggunaan voucher
7	limit_usages	bigint	Batas penggunaan voucher
8	free_shipping	bool	Status apakah voucher berupa kupon *free shipping* atau tidak
9	price	float	Persentase diskon yang diberikan

3.2.7.5 Kamus data tabel *coupon_usages*

Tabel 3.28 Kamus Data Tabel Coupon Usages

No	Nama Atribut	Tipe Data	Keterangan
1	id	int	PK, Autoincrement
2	id_user	bigint	FK ID User yang menggunakan voucher
3	id_item	bigint	FK ID Item
4	id_coupon	int	FK ID Coupon yang digunakan

Tabel 3.28 – lanjutan dari halaman sebelumnya

No	Nama Atribut	Tipe Data	Keterangan
5	price_after	bigint	FK Harga setelah penggunaan voucher

3.2.7.6 Kamus data tabel *favorites*

Tabel 3.29 Kamus Data Tabel *favorites*

No	Nama Atribut	Tipe Data	Keterangan
1	id	bigint	PK, Autoincrement
2	id_user	int	FK ID User
3	id_item_favorite	bigint	FK ID Item
4	remove_status	bool	
5	time_listed	timestamp	timestamp

3.2.7.7 Kamus data tabel *issues*

Tabel 3.30 Kamus Data Tabel *issues*

No	Nama Atribut	Tipe Data	Keterangan
1	id	int	PK, Autoincrement
2	id_issue_type	int	FK ID tipe issue yang dilaporkan
3	id_user	int	FK ID user yang melaporkan
4	id_issued_object	int	FK ID item yang dilaporkan

Tabel 3.30 – lanjutan dari halaman sebelumnya

No	Nama Atribut	Tipe Data	Keterangan
5	description	text	Deskripsi Laporan

3.2.7.8 Kamus data tabel *issue_types*

Tabel 3.31 Kamus Data Tabel *issuetypes*

No	Nama Atribut	Tipe Data	Keterangan
1	id	int	PK, Autoincrement
2	name	varchar(50)	Nama tipe laporan
3	priority	int	Prioritas laporan

3.2.7.9 Kamus data tabel *items*

Tabel 3.32 Kamus Data Tabel *items*

No	Nama Atribut	Tipe Data	Keterangan
1	id	bigint	PK, Autoincrement
2	id_user	int	FK ID User
3	name	varchar(255)	Nama barang
4	description	text	Deskripsi barang
5	starting_price	bigint	Harga awal lelang
6	expected_price	bigint	Harga akhir yang diinginkan
7	start_time	timestamp	Awal waktu lelang

Tabel 3.32 – lanjutan dari halaman sebelumnya

No	Nama Atribut	Tipe Data	Keterangan
8	end_time	timestamp	Akhir waktu lelang
9	id_category	int	FK - ID kategori barang
10	winner_chosen_status	bool	Status terpilih pemenang lelang
11	id_user_winner	int	FK - ID user pemenang lelang
12	bid_price	bigint	Harga penawaran lelang tertinggi pada barang tersebut
13	end_time_after_extended	timestamp	Waktu akhir extended
14	status_extend	int	Status extend waktu lelang barang
15	cancel_status	bool	Status cancel lelang barang
16	id_coupon_usage	int	FK - ID kupon yang digunakan pada barang tersebut

3.2.7.10 Kamus data tabel *rating_auctioneers*

Tabel 3.33 Kamus Data Tabel *ratingauctioneers*

No	Nama Atribut	Tipe Data	Keterangan
1	id	int	PK, Autoincrement
2	id_user _auctioneer	varchar(50)	FK ID User Penjual
3	id_user_rater	int	FK ID User pemberi review
6	rate	bigint	Jumlah rating yang diberikan
7	id_item	int	FK ID barang yang dirating
8	rate_message	text	Deskripsi Rate

3.2.7.11 Kamus data tabel *ratingbidders*

Tabel 3.34 Kamus Data Tabel *ratingbidders*

No	Nama Atribut	Tipe Data	Keterangan
1	id	int	PK, Autoincrement
2	id_user_bidder	varchar(50)	FK ID User Pelelang
3	id_user_rater	int	FK ID User pemberi review
6	rate	bigint	Jumlah rating yang diberikan
7	id_item	int	FK ID barang yang dirating
8	rate_message	text	Deskripsi Rate

3.2.7.12 Kamus data tabel *ratinglogs*

Tabel 3.35 Kamus Data Tabel *ratinglogs*

No	Nama Atribut	Tipe Data	Keterangan
1	id	int	PK, Autoincrement
2	id_user_winner	varchar(50)	FK ID User pemenang lelang
3	id_user_auctioneer	int	FK ID User Penjual
4	id_item	int	FK ID barang yang dirating
5	status_rating_from_winner	int	Flag apakah transaksi sudah dirating oleh pemenang lelang
6	status_rating_from_auctioneer	bigint	Flag apakah transaksi sudah dirating oleh penjual barang
7	id_rating_bidder	int	FK ID rating di tabel ratingauctioneers
8	id_rating_auctioneer	text	FK ID rating di tabel ratingbidders

3.2.7.13 Kamus data tabel *userverifications*

Tabel 3.36 Kamus Data Tabel *userverifications*

No	Nama Atribut	Tipe Data	Keterangan
1	id_user	int	FK ID User
2	created_at	timestamp	Timestamp
3	token	varchar(255)	Token random yang digenerate sistem

3.2.7.14 Kamus data tabel *userissues*

Tabel 3.37 Kamus Data Tabel *User Issues*

No	Nama Atribut	Tipe Data	Keterangan
1	id	int	PK, Autoincrement
2	id_user	int	FK - ID user yang melaporkan
3	id_issued_user	int	FK - ID User yang dilaporkan
4	description	text	Deskripsi laporan yang diberikan

3.2.7.15 Kamus data tabel *users*

Tabel 3.38 Kamus Data Tabel *users*

No	Nama Atribut	Tipe Data	Keterangan
1	id	int	PK, Autoincrement
2	name	varchar(255)	Nama

Tabel 3.38 – lanjutan dari halaman sebelumnya

No	Nama Atribut	Tipe Data	Keterangan
3	email	varchar(50)	Email
4	password	varchar(255)	Password
5	address	text	Alamat
6	phone	varchar(50)	No. HP
7	created_at	timestamp	Timestamp
8	updated_at	timestamp	timestamp
9	username	varchar(20)	Username
10	verification	timestamp	Tanggal user terverifikasi - jika belum diverifikasi, isinya NULL
11	is_banned	timestamp	Status banned pengguna

3.2.8 Kamus Data *Database* Non-Transaksional

Subbab ini akan berisi pemaparan detail kamus data dalam *database* yang bersifat non-transaksional. Keseluruhan terdapat 3 *collections*.

3.2.8.1 Kamus data *collection* userchats

Tabel 3.39 Kamus data tabel *userchat*

No	Nama Atribut	Tipe Data	Keterangan
1	id	-	Hash generated key
2	room	varchar	Room chat
3	sender	int	FK ID user pengirim sender

Tabel 3.39 – lanjutan dari halaman sebelumnya

No	Nama Atribut	Tipe Data	Keterangan
4	msg	int	FK ID user receiver
5	sent	ISODate	timestamp pesan diterima d

3.2.8.2 Kamus data *collection* chatroom

Tabel 3.40 Kamus Data Tabel Chatroom

No	Nama Atribut	Tipe Data	Keterangan
1	id	-	Hash generated
2	room	varchar	Room chat
3	id_user_1	int	FK ID user pengirim sender
5	created_at	ISODate	timestamp pesan pertama
6	updated_at	ISODate	timestamp pesan terakhir

3.2.8.3 Kamus data *collection* itemimages

Tabel 3.41 Kamus data tabel Itemimage

No	Nama Atribut	Tipe Data	Keterangan
1	id	-	PK - Generated key
2	item_id	int	FK - ID barang
3	url	int	Url gambar

3.2.9 Kamus Data

(Halaman ini sengaja dikosongkan)

BAB IV

IMPLEMENTASI

Pada bab ini dibahas mengenai implementasi aplikasi, dibagi menjadi empat subbab yaitu :

1. Implementasi Perangkat Keras / *Deployment*
2. Implementasi Perangkat Lunak
3. Implementasi Antarmuka / *User Interface*

4.1 Implementasi Perangkat Keras/ *Deployment*

Aplikasi dideploy secara *online*, dalam sebuah *Virtual Private Server* yang di*host* oleh *Digital Ocean*. Spesifikasi VPS yang digunakan adalah sebagai berikut:

1. ***Hardware***
   (a) CPU: Intel(R) Xeon(R) CPU E5-2630L v2 @ 2.40GHz
   (b) Operating System:
   (c) RAM: 512MB
   (d) Storage Space: 20GB
2. ***Operating System***
   (a) Architecture: 64bit
   (b) Kernel Version: Linux 4.4.0-75-generic x86 64
   (c) OS Version: Ubuntu 16.04.2 LTS Xenial
3. ***Networking Stats***
   (a) Tersambung ke Internet: Ya
   (b) IP Publik: Ya
   (c) Alamat IP Publik (IPv4): 188.166.179.2
   (d) *Average Download Speed*: 1371 Mbit/s
   (e) *Average Upload Speed*: 860.12 Mbit/s
   (f) DNS: Google
4. **Domain Stats**
   (a) HTTPS Support: Yes
   (b) SSL Certificate issued by: Avast
   (c) Domain: https://Lelangapa.com

(d) Testing-purpose subdomain: https://testing.lelangapa.com
(e) Domain issued by: Namecheap

4.2 Implementasi Perangkat Lunak

Pada subbab ini, penulis akan memaparkan mengenai spesifikasi dan pemasangan perangkat lunak yang dibutuhkan dalam rancang bangun aplikasi lelang online ini.

4.2.1 Strategi *Deployment*

Pada saat instalasi dan konfigurasi beberapa komponen perangkat lunak, tidak tidak keseluruhan prosesnya berjalan dengan baik dalam sekali percobaan. Berikut paparan beberapa kesulitan dan penyelesaiannya.

4.2.1.1 NGINX

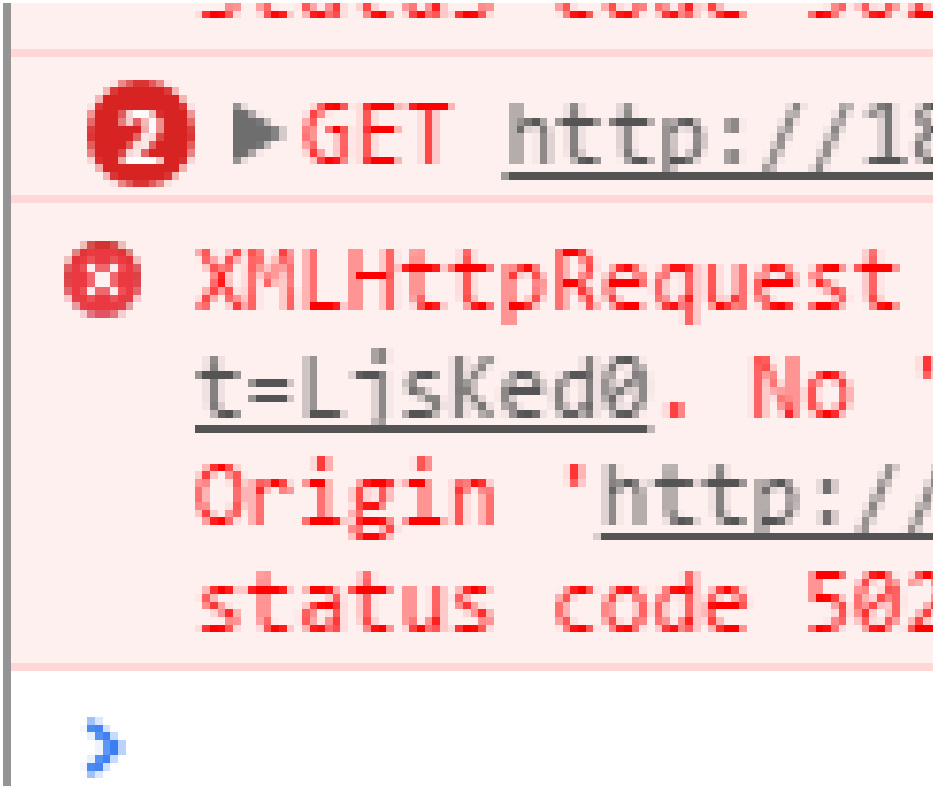
1. CORS (*Cross Origin Resource Sharing*)

   CORS adalah sebuah mekanisme yang memungkinkan sebuah *website* menggunakan *resources*(seperti skrip Javascript, *fonts*, dll) untuk diakses dari sumber lain selain *domain origin*nya.

   Secara teknis, CORS mendefinisikan *protokol*/cara browser dan server untuk berinteraksi otorisasi permintaan *resource* dari domain lain, dan juga lebih aman karena *developer* dapat mengkontrol otorisasi tersebut (daripada mengizinkan semua permintaan).

   The Problem

   Masalah koneksi ke server lelang (yang berjalan pada domain yang sama, namun port yang berbeda) tidak dapat tersambung karena *error* berikut :



Gambar 4.1 *Error* CORS yang muncul pada console browser

Insight

Selama 2 minggu kurang lebih penulis mencoba cara yang ditemukan penulis dalam situs stackoverflow untuk mengkonfigurasi *server* lelang agar otorisasi CORS dapat dilakukan, namun tidak ada hasil.

Masalah ini terselesaikan setelah menggunakan fitur *reverse proxy* dari Nginx.

Solution

Strateginya adalah sebagai berikut :

(a) Server Lelang berjalan pada localhost port 3000.
(b) Kita mengatur sebuah subdomain khusus – misalkan A.domain.com
(c) Dalam NGINX , kita konfigurasi agar semua *requests* menuju A.domain.com ke aplikasi localhost yang kita maksud di poin **??**
(d) Selain meneruskan *requests*, NGINX juga akan meneruskan *reply* dari server lelang tersebut kepada *client*/*origin* yang meminta *request* tersebut.

4.2.1.2 VUE.JS

1. ***Package Dependencies***

Pada versi terbaru Laravel (5.4*), Laravel secara *default* menyertakan *package* Laravel Mix - yaitu fitur untuk *compiling assets* dengan Webpack, dengan hasil akhir *compiled assets* (terutama *script* Javascript) yang eksekusinya jauh lebih cepat, karena menggunakan V8 – sebuah *engine* Javascript yang telah dioptimasi yang bersifat *just-in-time* (JIT) yang memproduksi *machine code* dari sebuah *script* Javascript lalu dieksekusi.

Main Problem

Masalah muncul saat versi Laravel yang digunakan untuk membangun aplikasi adalah versi (5.3) – dan jika Laravelnya di*upgrade*, tidak ada jaminan bahwa *deprecated dependencies* (keadaan dimana sebuah *package* tidak di*support* oleh versi terbaru) – yang berarti harus *refactoring code* yang pasti memakan waktu lama.

Insights

Penulis menganalisa perbedaan mendasar package.json antara Laravel 5.3 dan 5.4 adalah sebagai

berikut:

a. Basis : Perubahan basis yang awalnya Gulp menjadi Webpack
b. *Dependencies* : Webpack ternyata menggunakan beberapa plugin tambahan yang tidak diakomodasi dalam package.json di versi 5.3
c. *Run Script* : Terdapat beberapa perubahan signifikan terhadap *run script alias* di versi 5.4 - dibandingkan pada versi 5.3.
d. *Additional Files* : Terdapat beberapa file konfigurasi tambahan agar proses kompilasi aset dapat berjalan dengan baik.

Solution

Penulis lalu mengoreksi dan *update package.json* dengan pendekatan *trial and error*, dan bisa terselesaikan dengan script berikut :

```
1  { "private": true,
2    "scripts":    {
3          "_comment" : "Lists of running npm
               ↪ commands defined here"
4    },
5    "devDependencies": {
6          "axios": "^0.15.3",
7          "bootstrap-sass": "^3.3.7",
8          "cross-env": "^3.2.3",
9          "jquery": "^3.1.1",
10         "laravel-mix": "0.*",
11         "lodash": "^4.17.4",
12         "vue": "^2.1.10"
13   },
14   "dependencies": {
15         "vue-resource": "^1.3.1"
16   } }
```

2. ***Dependencies Optimization Problem***

Setelah menulis beberapa *script* Vue, penulis menyadari bahwa setiap *script* Vue ternyata mempunyai *dependencies* yang sama, yaitu axios, Promise, toastr dan vue. Setiap file Vue meng*include* sebuah *script* yang berisi:

```
1  window.axios = require('axios');
2  window.toastr = require('toastr');
3  window. = require('toastr');
4  require('vue-resource');
```

Hal ini mengakibatkan semua file vue yang di*compile* ukurannya cukup besar (400kb), padahal sebenarnya di dalam setiap file tersebut sebenarnya ada yang sama. Hal ini tentu tidak efektif, karna sebenarnya hal-hal yang sama tersebut bisa dipisahkan, dan dijadikan *cache* sehingga *loading* halaman bisa jauh lebih cepat.

Insight & Solution

Setelah penulis menanyakan dan mendiskusikan di forum Slack, beberapa pengguna Vue menyarankan untuk *compile* keseluruhan *dependencies* yang digunakan kedalam satu file terpisah, dan hanya menulis logika Vue untuk setiap file Vue.

Isi file webpack.mix.js (file yang di*compile* oleh Webpack) menjadi seperti berikut.

```
1  /* dependencies all compiled into one single
       ↪  file */
2  mix.js('scripts/dependencies.js', 'public/js
       ↪ ');
```

```
3
4  /* dependencies all compiled into one single
       ↪  file */
5  mix.js('scripts/favorites.js', 'public/js');
6  mix.js('scripts/other_vue_script.js', '
       ↪ public/js');
```

Dan di HTML, untuk *including script* dituliskan seperti berikut:

```
1  <script
2   src="dependencies.js" >
3  </script>
4  <script
5   src="custom_page_script.js" >
6  </script>
```

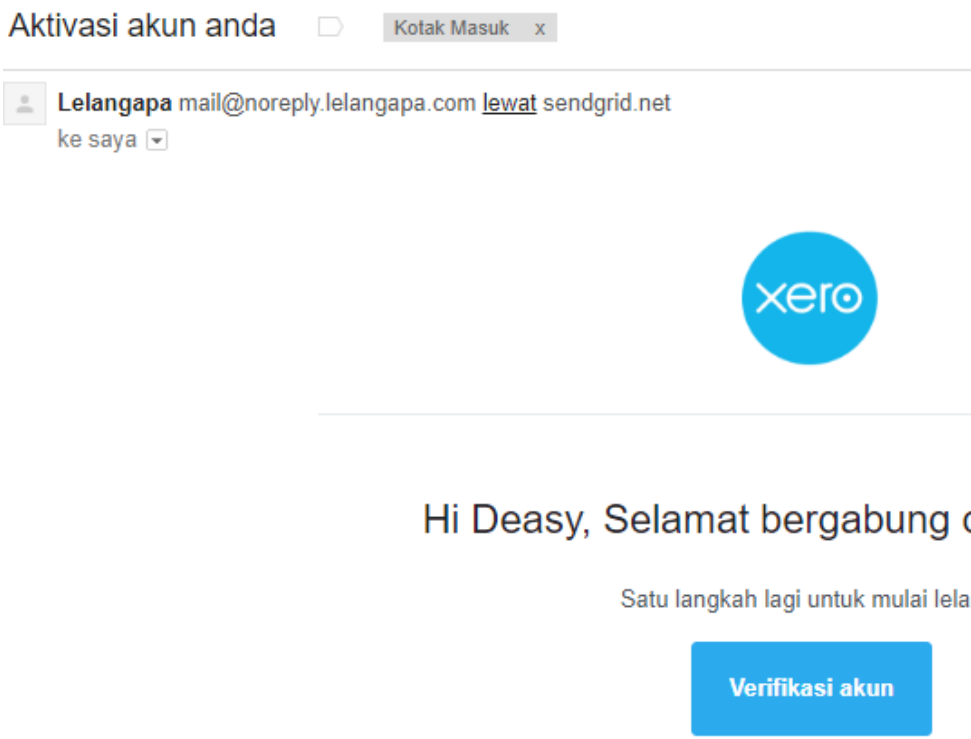
4.2.1.3 *Whitelisting* pada SendGrid

Whitelisting adalah kebalikan dari teknologi *blacklist*. Jika *blacklist* merupakan daftar dari sekumpulan web domain ataupun alamat email, dan URL yang terindikasi tidak "aman" sehingga secara otomatis akan diblokir oleh komputer maupun jaringan agar tidak dapat diakses, maka *whitelist* kebalikan dari *blacklist* yaitu daftar yang diperbolehkan untuk diakses oleh komputer atau jaringan dimana terdapat sekumpulan URL, web domain maupun alamat email yang "aman".

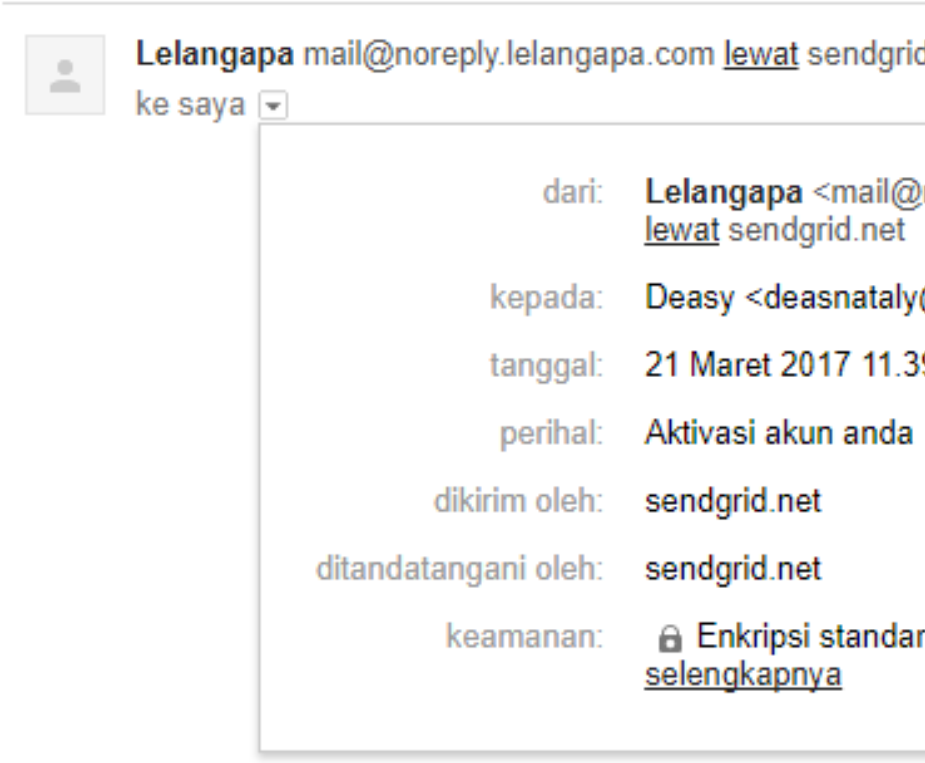
Masalah dimulai saat *email* konfirmasi akun yang dikirimkan oleh sistem aplikasi lelang online - dengan menggunakan SMTP *relay* - kepada *email pengguna*, masuk ke dalam kotak pesan sebagai *spam*. Hal ini tentu tidak baik - karena seharusnya masuk ke *inbox* sebagaimana *email* pada umumnya. Setelah penulis mencoba mencari jalan keluar, terutama SendGrid tidak menyediakan *whitelisting* dengan menggunakan *root domain* (karena penulis membeli domain lelangapa.com, penulis tidak

dapat mengirim *email* konfirmasi akun dengan menggunakan any_email_address@lelangapa.com, dan hingga buku ini ditulis penulis tetap tidak tahu alasan pastinya apa). Masalah ini baru selesai dengan menggunakan pemecahan berikut:

1. Membuat sebuah domain noreply.lelangapa.com
2. Me*register* domain *whitelisting* di pengaturan SendGrid



Gambar 4.2 *Whitelisting*



Gambar 4.3 Detail Informasi Email yang Masuk ke Kotak Masuk Pengguna

4.2.2 Diagram Kelas

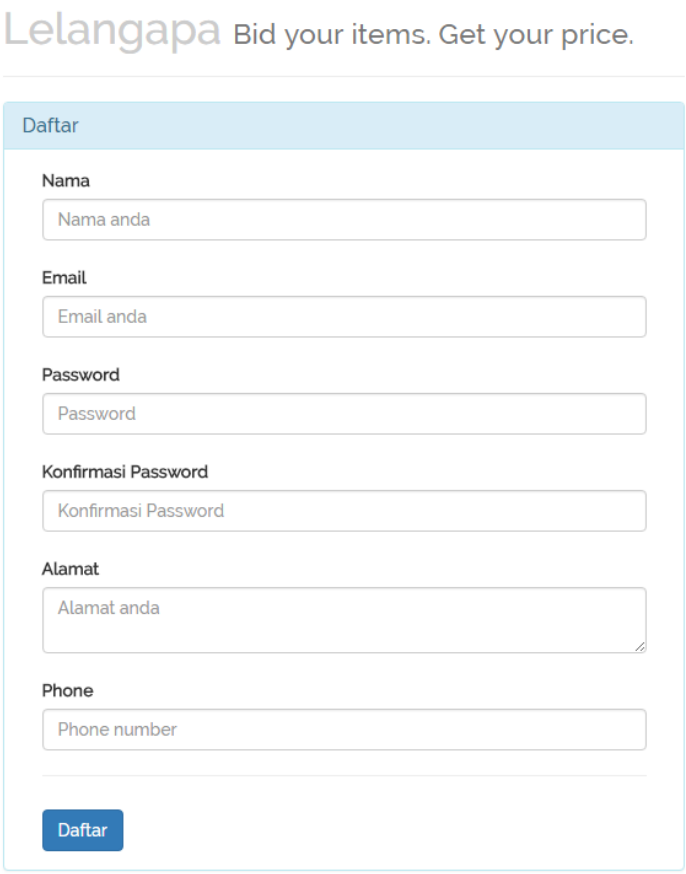
4.3 Implementasi Antarmuka / *User Interface*

4.3.1 Antarmuka Halaman Registrasi

Halaman ini dapat diakses oleh semua pengguna, baik yang belum terdaftar maupun sudah. Halaman ini menampilkan form berisi elemen *input* data diri, dan pengguna dapat mengisi lalu mengklik tombol daftar, dan untuk kasus alternatif dapat dilihat

pada tabel spesifikasi kasus penggunaan **??**.

Tidak ada *view logic* ataupun logika *UI* khusus dalam halaman ini. Kode sumber implementasi *back-end* dapat dilihat pada kode sumber **??**.



Gambar 4.4 Halaman antarmuka registrasi

```
1  /*
2   * Menampilkan halaman register
3   * method : GET
4   */
5  public function showRegistrationForm(){
6
7          return view('auth.register2');
8  }
9
10 /*
11  * validator fungsi
12  */
13 protected function validator(array $data){
14
15         return Validator::make($data, [
16                 'name' => 'required|max:255',
17                 'email' => 'required|email|max
                       ↪ :255|unique:users',
18                 'password' => 'required|min:6|
                       ↪ confirmed',
19                 'username' => 'required|unique:
                       ↪ users|min:5',
20                 'phone' => 'numeric',
21         ]);
22 }
23
24  /*
25   * Dipanggil saat mengklik tombol daftar
26   * method : POST
27   */
28 public function register(Request $request){
29
30
31         /*      validasi data */
32         $this->validator($request->all())->
               ↪ validate();
33
```

```
34          event(new Registered($user = $this->
                ↪ create($request->all())));
35
36          $this->guard()->login($user);
37
38          /* notify activationService
39          to send activation mail to user's email
                ↪ */
40          $this->activationService->
                ↪ sendActivationMail($user);
41
42          return $this->registered($request, $user)
                ↪  ?
43             : redirect($this->redirectPath());
44
45  }
```

Kode Sumber IV.1 Kode Sumber Antarmuka Registrasi

4.3.2 Implementasi Halaman *Login*

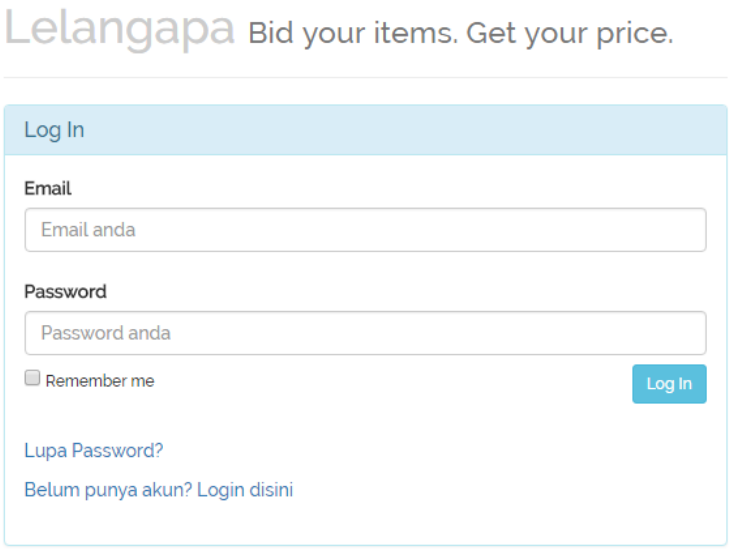
Halaman ini dapat diakses oleh semua pengguna, baik yang belum terdaftar maupun sudah, dengan pengecualian pengguna tidak dalam keadaan sudah *login*. Halaman ini menampilkan form berisi elemen *input* email dan *password*, dan pengguna dapat mengisi lalu mengklik tombol *login*, dan untuk kasus normal dan alternatif dapat dilihat pada tabel spesifikasi kasus penggunaan **??**.

Tidak ada *view logic* dalam halaman ini. Kode sumber implementasi *back-end* dapat dilihat pada kode sumber **??**.

```
1  public function showLoginForm(){
2          /*Menampilkan halaman login
3            Method : GET */
4          return view('auth.login2');
5  }
```

```
6
7   public function login(Request $request){
8           /*      Setelah klik tombol login,
9                   masuk ke dalam fungsi ini
10                  Method : POST */
11          $this->validateLogin($request);
12          return $this->sendLoginResponse($request)
                ↪ ;
13  }
```

Kode Sumber IV.2 Kode Sumber Antarmuka Registrasi



Gambar 4.5 Halaman antarmuka

4.3.3 Melihat daftar barang yang dilelang

Halaman ini hanya dapat diakses oleh pengguna yang sudah terdaftar dan sudah login ke dalam sistem. Halaman ini menampilkan form berisi elemen *input* data diri, dan pengguna dapat mengisi lalu mengklik tombol daftar, dan untuk kasus normal dan alternatif dapat dilihat pada tabel spesifikasi kasus penggunaan **??**.

Terdapat view logic khusus pada halaman ini yang ditulis menggunakan Vue dan di*compile* dengan menggunakan webpack, yang akan dicantumkan dalam kode sumber **??**. Kode sumber implementasi *back-end* dapat dilihat pada kode sumber **??**.

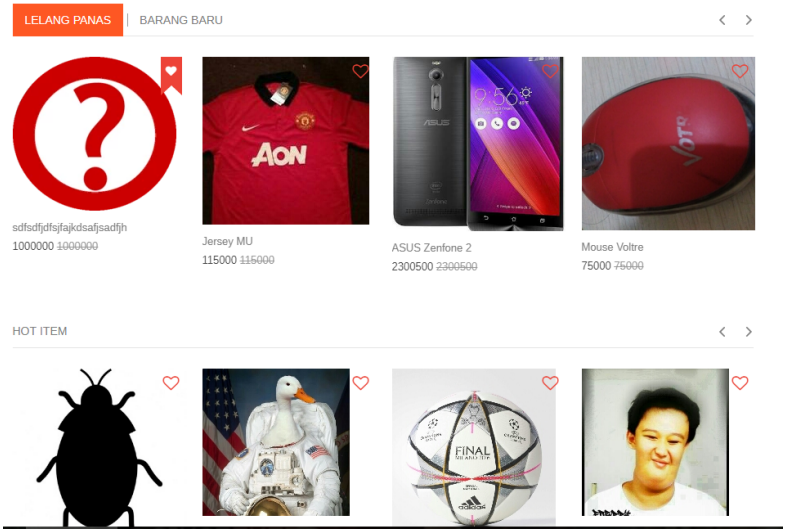
```
1  /*      file : app/Http/Controllers/
       ↪ HomeController*/
2   public function index(){
3           /*      method : GET */
4
5           /*      variabel berisi id barang
6                   yang disort dari tanggal
                        ↪ perbaruan
7                   secara descending */
8           $data['items'] = Item::all()->sortByDesc
                ↪ ('created_at');
9
10          /*      variabel berisi id barang
11                   yang sedang aktif proses lelang
12                   menggunakan repository :
                         ↪ itemRepository */
13          $data['activebid'] = $this->
                ↪ itemRepository->getActiveItem();
14
15
16          return view('pages.general.landing',
                ↪ $data);
17  }
```

Kode Sumber IV.3 Kode Sumber *Back-end* Melihat Daftar Barang

```
<div>
  <img :src="imgUrl"  class="img-responsive" />

  <div class="ribbon" v-if="isFavorited"
      ↪ @click.prevent="unFavorite(item)" >
   <div class="border-ribbon"></div>
   <i class="fa fa-heart"></i>
  </div>

  <div class="unribbon" v-else @click.prevent="
      ↪ favorite(item)">
   <div class="border-ribbon"></div>
   <i class="fa fa-heart-o"></i>
  </div>
</div>

export default {
    props: ['item', 'favorited'],
    data: function() {
        return {
            isFavorited: '',
            imgUrl : 'http://URL_GAMBAR_DEFAULT'
        }
    },
    mounted() {
        this.isFavorited = this.isFavorite ? true
            ↪  : false;
    },
    created() {
        axios.get("get/img/item/" + this.item)
            .then( response => {
             if(response.data.replace(/\s+/g, '')
                 ↪  != '' ){
              var url =  /*rewrite image url*/;
```

```
31                  this.imgUrl = url.replace(/\s+/g, '
                        ↪ ');
32                  console.log(this.imgUrl);
33              }
34          });
35      },
36      computed: {
37          isFavorite()
38          {
39              return this.favorited;
40          }
41      },
42      methods: {
43          favorite(item){
44              axios.post('/ajax/favourite/'+item)
45                  .then(response =>
                        ↪ this.isFavorited = true)
46              .catch(response => console.log(
                    ↪ response.data));
47          },
48          unFavorite(item) {
49              axios.post('/ajax/favourite/un/'+item
                    ↪ )
50                  .then(response =>
                        ↪ this.isFavorited = false)
51              .catch(response => console.log(
                    ↪ response.data));
52          }
53      }
54  }
```

Kode Sumber IV.4 Kode Sumber Vue Melihat Daftar Barang



Gambar 4.6 Halaman antarmuka

4.3.4 Mencari barang yang diinginkan

4.3.5 Menawar/melelang barang

Halaman ini hanya dapat diakses oleh pengguna yang sudah terdaftar. Halaman ini menampilkan halaman informasi barang, dan sebuah elemen *input* harga , dan pengguna dapat mengisi lalu mengklik tombol daftar, dan untuk kasus normal dan alternatif dapat dilihat pada tabel spesifikasi kasus penggunaan **??**.

Sebagai ringkasan dari ketiga logika tersebut, visualisasi pada gambar **??** akan membantu menggambarkan keseluruhan proses logika secara ringkas. Masing-masing logika tersebut dapat dijabarkan sebagai berikut:

1. Logika *back-end* ditulis menggunakan PHP yang dicantumkan dalam kode sumber **??**;

2. Logika *view* ditulis menggunakan jQuery yang dicantumkan dalam kode sumber **??**; dan
3. Logika proses lelang, berjalan diatas socket yang berjalan diatas Node.js dengan bantuan Socket.io yang dicantumkan dalam kode sumber **??**

Gambar 4.7 Halaman Antarmuka Implementasi Kasus Penggunaan Menawar/melelang barang

```
1  /*      file : app/Http/Controllers/
       ↪ ItemController*/
2  public function show($id){
3          /*      method : GET */
4
5          /*      tampilkan halaman */
6      $data['item'] = $this->itemRep->find($id);
7
8      if($data['item']->bid_status == 1 )
9      {
10         $data['currentPrice'] = $this->itemRep->
               ↪ getCurrentPrice($id);
```

```
11      }
12
13      /*  Jika barang yang disubmit sudah selesai/
            ↪ dimenangkan,
14          tambahkan variabel winner ke view */
15      if($data['item']->bid_status == -1 && $data['
            ↪ item']->winner_chosen_status)
16          $data['winnername'] = User::find($data['
                ↪ item']->id_user)->name;
17
18      /*  tampilkan halaman */
19      return view('pages.'.$this->pageFolder.'.
            ↪ detail', $data);
20  }
```

Kode Sumber IV.5 Kode Sumber *Back-end* Menampilkan Halaman Lelang Barang

```
1   /*
2    file : bidserver_https.js
3    menggunakan dependencies : socketio (ioServer),
         ↪ https, http, fs dan express
4   */
5   socket.on('submitbid', function(bidjson) {
6
7       // parameter (bidjson) adalah JSON yang
            ↪ terdiri dari :
8       // id_bidder, id_item, dan harga_bid
9
10      var bidobj = JSON.parse(bidjson);
11      biddingAPI.submitBid(bidobj, function(status,
            ↪  result)
12      {
13        //  status adalah return value
14        //  jika status = 1, broadcast ke room
15        //  untuk mengupdate pemenang lelang saat
              ↪ tersebut.
16        if (status=="1")
```

```
17          {
18              //jika sukses
19              var messageObject = {};
20              var tokenArray;
21
22                // fungsi untuk mengconstruct
23                // return message
24              constructBroadcastMsg(messageObject);
25
26                // broadcast ke semua yang
27                // join ke room lelang tersebut.
28              io.to(result.item_id_return).emit('
                    ↪ bidsuccess', result);
29
30          }
31          else if (status=="0"){
32            // jika gagal,
33            // maka send ke sender bahwa bid failed
34            socket.emit('bidfailed', { bidstatus: "
                    ↪ failed" });
35          }
36      });
37
38  });
```

Kode Sumber IV.6 Kode Sumber Logika Lelang (menggunakan Node.js)

```
1   // script ini berjalan
2   // saat tombol Tawar diklik
3
4   $('.tawar').click(function()
5   {
6       var penawaran = $('#submitted_price').val();
7       var priceNow = $('.priceongoing').html();
8       if(penawaran==""){
9           alertNoPriceSubmitted();
10      }
```

```
11      else if(penawaran < priceNow ){
12          alertInvalidBid();
13      }
14      else {
15          var JSONToSend;
16          constructJson(JSONToSend);
17
18          // submit bid ke socket
19          socket.emit('submitbid', JSONToSend);
20
21          // tunggu return dari socket
22          // jika hasilnya bid failed, maka
                ↪ tampilkan pesan error
23          socket.on('bidfailed', function (result)
                ↪ {
24              alertFailedBid();
25          });
26
27          // jika hasilnya bid sukses,
28          // tampilkan pesan sukses (toast)
29          // dan render ulang halaman (ubah isi
                ↪ elemen)
30          socket.on('bidsuccess', function (result)
                ↪  {
31              alertSuccessfulBid();
32              renderPageWithNewPrice(result);
33          });
34      }
35  });
```

Kode Sumber IV.7 Kode Sumber Logika UI (menggunakan jQuery)

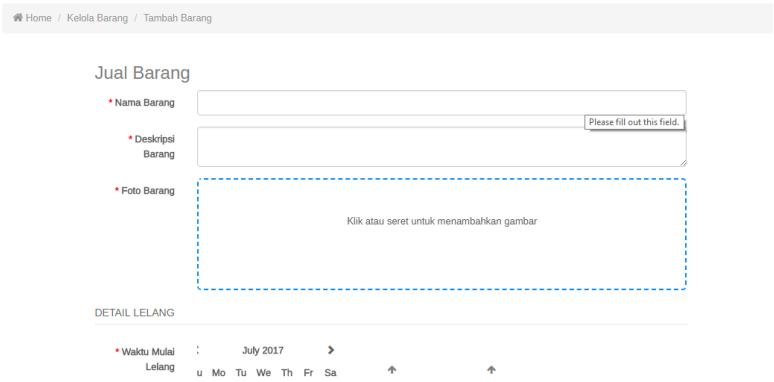
4.3.6 Mendaftarkan Barang untuk Dilelang

Halaman ini hanya dapat diakses oleh pengguna yang sudah terdaftar dan sudah login ke dalam sistem. Halaman ini menampilkan form berisi elemen *input* informasi barang, dan

pengguna dapat mengisi lalu mengklik tombol tambahkan barang, dan untuk kasus normal dan alternatif dapat dilihat pada tabel spesifikasi kasus penggunaan **??**.

Terdapat logika *view* khusus pada implementasi kasus penggunaan ini, karena adanya kebutuhan untuk *upload multiple images* untuk setiap barang dan adanya penggunaan layanan pihak ketiga (AWS Storage Service) sebagai tempat penyimpanan gambar. Kode-kode sumber terkait dengan implementasi kasus penggunaan Mendaftarkan Barang untuk Dilelang adalah sebagai berikut:

1. Logika *back-end* ditulis menggunakan PHP yang dicantumkan dalam kode sumber **??**;
2. Logika *view* ditulis menggunakan jQuery yang dicantumkan dalam kode sumber **??**; dan
3. Logika *upload*/*storage* data gambar, yang ditulis menggunakan PHP dicantumka pada kode sumber **??**



Gambar 4.8 Halaman antarmuka registrasi

```
1  /*
2      Fungsi create() dipanggil untuk menampilkan
           ↪ halaman tambah barang
```

```
3       Fungsi store() dipanggil saat form di halaman
            ↪  tambah barang diklik
4       File : ItemController
5   */
6   public function create(){
7       /*  method : GET */
8       return view('pages.'.$this->pageFolder.'.
            ↪ create', $data);
9   }
10
11
12  /*
13      Fungsi store() dipanggil oleh ajax lewat
            ↪ jQuery
14      untuk memvalidasi dan insert data ke dalam
            ↪ database
15  */
16  public function store(Request $request)
17  {
18      /*  method : POST */
19      $unserialize = $this->unserializeForm(
            ↪ $request['data']);
20      /*  Validating data */
21      $validate = Validator::make($unserialize,
            ↪ $this->itemRep->rules(), $this->itemRep
            ↪ ->messages());
22
23      if(false){
24          /* Jika gagal, return false dan error
                ↪ message
25              ke view
26           */
27          return ['success'=>false, 'msg' =>
                ↪ $validate->messages()];
28      }
29      else{
30          /*
31              Jika sukses, return true
```

```
32              dan id barang yang berhasil diinsert
33              untuk selanjutnya diproses oleh
                    ↪ browser
34              untuk upload gambar barang ke URL
                    ↪ terpisah.
35           */
36          return ['success' => true, 'id' =>  10];
37      }
38  }
```

Kode Sumber IV.8 Kode Sumber *Back-end* Mendaftarkan Barang untuk Dilelang

```
1   /*
2       Fungsi ini berfungsi untuk \textit{update}
            ↪ gambar barang, diteruskan kepada \
            ↪ textit{script} terpisah
3       Pada file : ImageController
4       Method : Any
5   */
6   public function uploadItemImage(Request $request)
        ↪ {
7       //passed here if csrf token is already passed
8       $_POST['image'] = $request->file('image');
9       $_POST['id_user'] = Auth::user()->id;
10      $_POST['ext'] = $request->file('image')->
            ↪ extension();
11
12      require ('/home/img/upload-aws.php');
13
14      return $res;
15  }
```

Kode Sumber IV.9 Kode Sumber *Back-end* Mendaftarkan Barang untuk Dilelang

```
1   /*      Fungsi ini berfungsi untuk \textit{upload
        ↪ } gambar lewat script terpisah (untuk
```

```
    ↪ alasan integrasi data dengan sistem yang
    ↪ dibuat oleh partner tugas akhir)
*/
require 'vendor/autoload.php';

use Aws\S3\S3Client;

$collection = (new MongoDB\Client("mongodb://
    ↪ 127.0.0.1:27017"))->lelangkita->itemimages;
if ($_SERVER['REQUEST_METHOD']=='POST'){
        $image = $_POST['image'];
        $userid = $_POST['id_user'];
        $ext = $_POST['ext'];
        $itemid = $_POST['itemid'];
        $isMainImage = false;

        if (isset($_POST['is_main_image']) && !
            ↪ empty($_POST['is_main_image']) &&
            ↪ $_POST['is_main_image'] == "true")
            ↪ {
                $isMainImage = true;
        }

        $year = date("Y");
        $month = date("m");
        $date = date("d");
        $unique = uniqid();
        $filename = $userid."_".$unique.".".$ext;
        $path = "img/".$year."/".$month."/".$date
            ↪ ."/".$userid;
        $imgpath = $path."/".$filename;

  $decoded_image = base64_decode($image);

  try {
    $s3client = S3Client::factory(array(
      'credentials' => array(
        'key' => 's3\_key\_credentials',
```

```
33          'secret' => 's3\_secret\_credentials'
34        ),
35        'profile' => 's3\_profile',
36        'region' => 's3\_selected\_server\_regopm'
37      ));
38
39      $upload = $s3client->putObject(
40        array(
41          'Bucket' => 'img-s7.lelangapa.com',
42          'Key' => $imgpath,
43          'Body'   => $decoded_image,
44          'ContentEncoding' => 'base64',
45          'ContentType' => 'image/'.$ext,
46          'ACL'    => 'public-read'
47        )
48      );
49
50      $insertToDB = $collection->insertOne([
51                  'item_id' => $itemid,
52                  'url' => $imgpath,
53                          'is_main_image' =>
                                ↪ $isMainImage
54          ]);
55
56                  if ($isMainImage == true) {
57                          $indexImageURL = "https:
                                ↪ //src-api.lelangapa
                                ↪ .com/apis/index/
                                ↪ submit/image";
58                          $headers = array(
59                            'Content-Type:
                                  ↪ application/json'
60                          );
61                          $post_data = array(
62                                  "id_item" =>
                                        ↪ $itemid,
63                                  "main_image_url"
                                        ↪ => $imgpath
```

```
64                          );
65                          $ch = curl_init();
66                          curl_setopt($ch,
                                ↪ CURLOPT_URL,
                                ↪ $indexImageURL);
67                          curl_setopt($ch,
                                ↪ CURLOPT_HTTPHEADER,
                                ↪  $headers);
68                          curl_setopt($ch,
                                ↪ CURLOPT_RETURNTRANSFER
                                ↪ , 1);
69                          curl_setopt($ch,
                                ↪ CURLOPT_POST, true)
                                ↪ ;
70                          curl_setopt($ch,
                                ↪ CURLOPT_POSTFIELDS,
                                ↪  json_encode(
                                ↪ $post_data));
71                          curl_setopt($ch,
                                ↪ CURLOPT_SSL_VERIFYPEER
                                ↪ , FALSE);
72                          $indexresult = curl_exec(
                                ↪ $ch);
73                          curl_close($ch);
74                  }
75
76      if ($upload) {
77        $res = array('status' => 'success', 'result
              ↪ ' => '1');
78                  echo "success";
79      }
80      else {
81        $res = array('status' => 'failed', 'result'
              ↪  => '0');
82                  echo "Failed";
83      }
84    }
85    catch(Exception $e) {
```

```
86          exit($e->getMessage());
87      }
88  }
```

Kode Sumber IV.10 Kode Sumber *Back-end* Upload Gambar Barang

4.3.7 Memperbarui Barang

Halaman ini dapat diakses oleh semua pengguna, baik yang belum terdaftar maupun sudah. Halaman ini menampilkan form berisi elemen *input* data barang yang sebelumnya sudah terisi data-data barang sebelumnya, dan pengguna dapat memperbarui data lalu mengklik tombol Perbarui, dan untuk kasus alternatif dapat dilihat pada tabel spesifikasi kasus penggunaan **??**.

Pada implementasinya, kasus penggunaan ini kurang lebih sama dengan kasus pengunaan Mendaftarkan Barang (subbab **??**) karena adanya penggunaan layanan pihak ketiga (AWS Storage Service) sebagai tempat penyimpanan gambar. Kode-kode sumber terkait dengan implementasi kasus penggunaan Mendaftarkan Barang untuk Dilelang adalah sebagai berikut:

1. Logika *back-end* ditulis menggunakan PHP yang dicantumkan dalam kode sumber **??**;
2. Logika *view* ditulis menggunakan jQuery yang dicantumkan dalam kode sumber **??**; dan
3. Logika *update image*/*storage* data gambar, yang ditulis menggunakan PHP dicantumkan pada kode sumber **??**

```
1  /*
2       Fungsi edit() dipanggil untuk menampilkan
            ↪ halaman perbarui barang
3       Fungsi update() dipanggil saat form di
            ↪ halaman update barang disubmit
4       File : ItemController
5  */
6  public function edit($id)
```

```
7  {
8          /*      method : GET */
9      $data['item'] = $this->itemRep->find($id);
10
11     /*  Jika barang sedang aktif lelang,
12         tidak dapat diedit (redirect kembali)
               ↪ */
13     if($data['item']->bid_status==1){
14         return redirect()->back()->with('
               ↪ errorItem','Maaf barang yang sedang
               ↪  dilelang tidak dapat diedit!');
15     }
16     /*  otorisasi       */
17     if($data['item']->id_user!=Auth::user()->id){
18         return redirect()->back()->with('
               ↪ errorItem','Maaf anda tidak
               ↪ terotorisasi!');
19     }
20
21     return view('pages.'.$this->pageFolder.'.
           ↪ update', $data);
22 }
23
24
25 /*
26     Fungsi update() dipanggil oleh ajax lewat
           ↪ jQuery
27     untuk memvalidasi dan update data ke dalam
           ↪ database
28 */
29 public function store(Request $request)
30 {
31     /*  method : POST */
32     $unserialize = $this->unserializeForm(
           ↪ $request['data']);
33     /*  Validating data */
34     $validate = Validator::make($unserialize,
           ↪ $this->itemRep->rules(), $this->itemRep
```

```
            ↪ ->messages());
35
36      if(false){
37          /* Jika gagal, return false dan error
                ↪ message
38              ke view
39           */
40          return ['success'=>false, 'msg' =>
                ↪ $validate->messages()];
41      }
42      else{
43          /*
44              Jika sukses, return true
45              dan id barang yang berhasil diinsert
46              untuk selanjutnya diproses oleh
                    ↪ browser
47              untuk upload gambar barang ke URL
                    ↪ terpisah.
48           */
49          return ['success' => true, 'id' =>  10];
50      }
51  }
```

Kode Sumber IV.11 Kode Sumber *Back-end* Memperbarui Barang

```
1   /* Fungsi ini berfungsi untuk \textit{update}
        ↪ gambar barang, diteruskan kepada \textit{
        ↪ script} terpisah
2   */
3   init: function() {
4                   var submitButton =
                        ↪ document.querySelector("#
                        ↪ click");
5                   var myDropzone = this; // closure
6                   console.log(
                        ↪ myDropzone.files.length);
7    submitButton.addEventListener("click", function
        ↪ () {
```

```
8       swal({
9           text: 'Memperbarui data barang anda...',
10          allowOutsideClick: false,
11          showConfirmButton: false,
12          onOpen: function (){
13              var dataSubmission = $('.submititem'
                    ↪ ).serialize();
14              var startime='&start_time=' +
                    ↪ addTimebasedTZ($('#start_time'
                    ↪ ).data().date);
15              var endtime = '&end_time=' +
                    ↪ addTimebasedTZ($('#end_time').
                    ↪ data().date);
16              dataSubmission += startime;
17              dataSubmission += endtime;
18              $.ajax({
19                  type: "POST",
20                  url: '{{ route('item.update') }}
                        ↪ ',
21                  data: { _token: '{{csrf_token()
                        ↪ }}', data : dataSubmission
                        ↪  },
22                  success: function( msg ) {
23                      console.log (msg);
24                      if(msg.success == false){
25                          $('.asdf').css('display'
                                ↪ ,'block');
26                          $('.asdf').empty();
27                          $.each(msg.msg, function
                                ↪ (i, val){
28                              $('.asdf').append(
29                                  '<li>'+val+'</li
                                        ↪ >'
30                              );
31                          });
32                          swal('Oops','Maaf, data
                                ↪ anda belum valid.
                                ↪ Silahkan cek
```

```
                    ↪ kembali','error');
33
34                  }
35                  else{
36                      if(
                        ↪ myDropzone.files.length
                        ↪  >0){
37                      iditem = msg.id;
38                      myDropzone.processQueue
                            ↪ ();
39                  }
40
41                      else{
42
43                          swal({
44                              title: 'Sukses!'
                                ↪ ,
45                              type:'success',
46                              allowOutsideClick
                                ↪  : false,
47                              showConfirmButton
                                ↪  : false,
48                              text: "Sukses
                                ↪ menambahkan
                                ↪  barang!",
49                              timer: 1000
50                          }).then(function ()
                                ↪ {
51                              document.location
                                ↪  = "{{ url
                                ↪ ('item/')
                                ↪ }}" +
                                ↪ iditem;
52                          });
53                      }
54                  }
55
56              }
```

```
57                  });
58              }
59          });
60      });
61
62      this.on("processing", function() {
63          swal('Uploading image..');
64      });
65
66      this.on("sending", function(file, xhr, data) {
67          if(iditem != 0){
68              data.append("_token", "{{ csrf_token()
                    ↪ }}");
69              data.append("itemid", iditem);
70          }
71      });
72  },
73  /*
74   saat gagal error gambar,
75   bagian fungsi error dipanggil
76  */
77  error: function(file, response) {
78      swal({
79          title: 'Oops',
80          type:'error',
81          allowOutsideClick: false,
82          showConfirmButton:false,
83          text: "Maaf ada kesalahan upload gambar,
                    ↪ silahkan upload gambar di halaman
                    ↪ edit gambar.",
84          timer: 2000
85      }).then(function () {
86          document.location = "{{ url('item') }}/"
                    ↪ + iditem;
87      });
88
89  },
90  /*
```

```
91   saat sukses update gambar,
92   bagian fungsi error dipanggil
93  */
94  success: function(file,done) {
95      swal({
96          title: 'Sukses!',
97          allowOutsideClick: false,
98          showConfirmButton:false,
99          type:'success',
100         text: "Sukses memperbarui barang!",
101         timer: 1000
102     }).then(function () {
103         document.location = "{{ url('item/') }}"
                ↪ + iditem;
104     });
105 }
106
107 });
```

Kode Sumber IV.12 Kode Sumber *View* Memperbarui Barang

```
1   /*      Fungsi ini berfungsi untuk \textit{upload
        ↪ } gambar lewat script terpisah (untuk
        ↪ alasan integrasi data dengan sistem yang
        ↪ dibuat oleh partner tugas akhir)
2   */
3
4   use Aws\S3\S3Client;
5
6   $collection = (new MongoDB\Client("mongodb://
        ↪ 127.0.0.1:27017"))->lelangkita->itemimages;
7   if ($_SERVER['REQUEST_METHOD']=='POST'){
8           constructFilePath();
9           $imgpath = $path."/".$filename;
10
11          $decoded_image = base64_decode($image);
12
13    try {
```

```
14      $s3client = S3Client::factory(array(
15        'credentials' => array(
16          'key' => 's3\_key\_credentials',
17          'secret' => 's3\_secret\_credentials'
18        ),
19        'profile' => 's3\_profile',
20        'region' => 's3\_selected\_server\_region'
21      ));
22
23      $upload = $s3client->putObject(
24        array(
25          'Bucket' => 'img-s7.lelangapa.com',
26          'Key' => $imgpath,
27          'Body'   => $decoded_image,
28          'ContentEncoding' => 'base64',
29          'ContentType' => 'image/'.$ext,
30          'ACL'    => 'public-read'
31        )
32      );
33
34      $insertToDB = $collection->insertOne([
35                  'item_id' => $itemid,
36                  'url' => $imgpath,
37                          'is_main_image' =>
                                ↪ $isMainImage
38          ]);
39
40                  if ($isMainImage == true) {
41                          $indexImageURL = "https:
                                ↪ //src-api.lelangapa
                                ↪ .com/apis/index/
                                ↪ submit/image";
42                          $headers = array(
43                            'Content-Type:
                                  ↪ application/json'
44                          );
45                          $post_data = array(
46                                  "id_item" =>
```

```
                                        ↪ $itemid,
47                                    "main_image_url"
                                        ↪ => $imgpath
48                              );
49                              $ch = curl_init();
50                              curl_setopt($ch,
                                    ↪ CURLOPT_URL,
                                    ↪ $indexImageURL);
51                              curl_setopt($ch,
                                    ↪ CURLOPT_HTTPHEADER,
                                    ↪  $headers);
52                              curl_setopt($ch,
                                    ↪ CURLOPT_RETURNTRANSFER
                                    ↪ , 1);
53                              curl_setopt($ch,
                                    ↪ CURLOPT_POST, true)
                                    ↪ ;
54                              curl_setopt($ch,
                                    ↪ CURLOPT_POSTFIELDS,
                                    ↪  json_encode(
                                    ↪ $post_data));
55                              curl_setopt($ch,
                                    ↪ CURLOPT_SSL_VERIFYPEER
                                    ↪ , FALSE);
56                              $indexresult = curl_exec(
                                    ↪ $ch);
57                              curl_close($ch);
58                      }
59          }
60          else {
61            $collection->updateOne(
62              ['_id' => new \MongoDB\BSON\ObjectID(
                    ↪ $imageid)],
63              ['$set' => array('url' => $imgpath, '
                    ↪ is_main_image' => $isMainImage)]
64            );
65          }
66
```

```
67      if ($upload) {
68        $res = array('status' => 'success', 'result
              ↪ ' => '1');
69                echo "success";
70      }
71      else {
72        $res = array('status' => 'failed', 'result'
              ↪  => '0');
73                echo "Failed";
74      }
75    }
76    catch(Exception $e) {
77      exit($e->getMessage());
78    }
79  }
```

Kode Sumber IV.13 Kode Sumber *Back-end* Upload Gambar Barang

4.3.8 Melihat Barang yang Didaftarkan

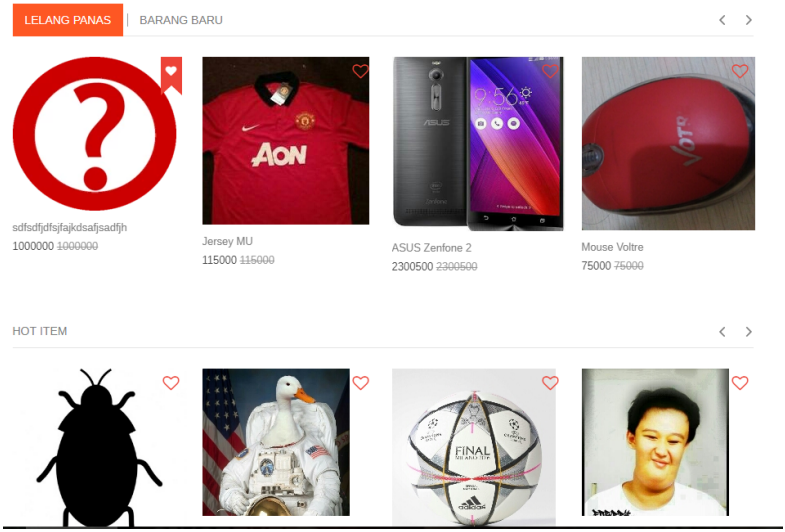
Halaman ini hanya dapat diakses oleh pengguna yang sudah terdaftar dan sudah login ke dalam sistem. Halaman ini berisi daftar barang yang pernah didaftarkan pengguna, sesuai dengan spesifikasi kasus penggunaan **??**.

Tidak ada *view logic* ataupun logika *UI* khusus dalam halaman ini. Kode sumber implementasi *back-end* dapat dilihat pada kode sumber **??**.

```
1  /*
2          file : app/Http/Controllers/
               ↪ ItemController
3          terkait dengan ItemRepository
4  */
5   public function index(){
6           /*      method : GET */
7
```

```
8       $data['items'] = $this->itemRep->getUserItem
            ↪ ();
9       return view('pages.'.$this->pageFolder.'.
            ↪ index',$data);
10  }
11
12  /*      file : app/Http/Controllers/
        ↪ ItemRepository */
13  public function getUserItem()
14  {
15      return Auth::user()->item()->orderBy('
            ↪ end_time','desc')->get();
16  }
```

Kode Sumber IV.14 Kode Sumber *Back-end* Melihat Barang yang Pernah Didaftarkan



Gambar 4.9 Halaman antarmuka

4.3.9 Melihat riwayat harga yang ditawarkan pada barang yang dilelang

4.3.10 Melihat Review Pengguna Lainnya

4.3.11 Menambahkan Review

4.3.12 Melaporkan Barang/Pengguna

4.3.13 Mengirimkan pesan

Halaman ini hanya dapat diakses oleh pengguna yang sudah terdaftar dan masuk/*login* ke dalam sistem. Pada halaman ini terdapat elemen-elemen halaman *chatting* pada umumnya, yaitu elemen *input* pesan, tombol Kirim, dan riwayat beberapa pesan sebelumnya. Spesifikasi kasus penggunaan dapat dilihat pada tabel **??**.

Terdapat logika *view* dan alur proses khusus dikarenakan sifat pengiriman dan penerimaan pesan yang realtime, sehingga dibangun diatas Node.js dan menggunakan Socket.io. Sebagai ringkasan dari ketiga logika tersebut, visualisasi pada gambar **??** akan membantu menggambarkan keseluruhan proses logika secara ringkas. Masing-masing logika tersebut dapat dijabarkan sebagai berikut:

1. Logika *back-end* ditulis menggunakan PHP yang dicantumkan dalam kode sumber **??**;
2. Logika *view* ditulis menggunakan jQuery yang dicantumkan dalam kode sumber **??**; dan
3. Logika proses pengiriman & penerimaan pesan, berjalan diatas socket yang berjalan diatas Node.js dengan bantuan Socket.io yang dicantumkan dalam kode sumber **??**

Gambar 4.10 Halaman Antarmuka Implementasi Kasus Penggunaan Mengirimkan Pesan

```
1  /*      file : app/Http/Controllers/
       ↪ ChatController */
2  public function chat($id_user)
3          /*      method : GET */
4
5      if(intval($id)){
6          $data['from'] = Auth::user()->id;
7          $data['to'] = $id;
8          $data['user'] = User::findOrFail($id);
9
10         if($data['user']){
11             return view('pages.user.chat', $data)
                   ↪ ;
12         }
13     }
14
15     /*  Jika ada parameter error,
16         redirect kembali dengan pesan error */
17     return redirect('/')->with('error','User yang
           ↪  anda cari tidak dapat ditemukan.');
18
19 }
```

Kode Sumber IV.15 Kode Sumber *Back-end* Mengirimkan Pesan

```
1  /*
2   file : chatserver\_https.js
3   menggunakan dependencies : socketio (ioServer),
        ↪ jwt, https, http, fs dan express
4  */
5  socket.on('join-room', function(room) {
6   /*
7    fungsi ini dipanggil saat
8    pertama kali pengguna membuka halaman chat
9    dimana pengguna masuk ke dalam room percakapan
10  */
11     var parsedRoom = parseRoom(room);
12
13
14  /*
15   Jika room ini belum pernah dimasuki/ chat
        ↪ perdana
16   maka ditambahkan ke tabel joinedroom
17  */
18     insertToRoomCollection(roomcollection,
           ↪ parsedRoom, false, function() {
19          socket.join(parsedRoom.room);
20
21          //
22          var cb = function(err, chat) {
23            if (chat!=[]){
24              socket.emit('chathistory', chat);
25            }
26          };
27
28   /*
29    Broadcast 5 percakapan terakhir
30    dalam room tersebut
31    ke pengguna yang baru saja join room.
```

```
32    */
33          collection.find({ room : parsedRoom.room
                ↪ }).sort({ sent : -1 }).limit(50).
                ↪ toArray(cb);
34      });
35
36  });
37
38
39  socket.on('send', function(data) {
40   /*
41    parameter yang masuk adalah JSON dengan
          ↪ konstruksi berikut :
42    { room: , body : }
43   */
44
45      var msgParse = JSON.parse(data);
46      var parsedRoom = parseRoom(msgParse.room);
47
48      //constructing inserting query
49      var insertQuery = {
50          room : parsedRoom.room,
51          sender : parsedRoom.from,
52          msg : msgParse.body,
53          sent : new Date()
54      };
55
56      //insert kedalam database
57      collection.insert(insertQuery, function(err,
            ↪ o) {
58          var ll = parsedRoom.from.toString();
59          if (err) io.to(ll).emit('send-status', {
                ↪ status : false});
60          else io.to(ll).emit('send-status', {
                ↪ status : true });
61      });
62
63      // update room information
```

```
64      // bahwa room ini sudah diperbarui
65      insertToRoomCollection(roomcollection,
            ↪ parsedRoom, true);
66
67      //broadcast ke user yang tergabung ke room
            ↪ percakapan
68      io.to(parsedRoom.room).emit('new-msg',
            ↪ insertQuery);
69
70  });
```

Kode Sumber IV.16 Kode Sumber Logika Lelang (menggunakan Node.js)

```
1   var room = '{{ Auth::user()->id }}-{{ $user->id
        ↪ }}';
2   var prepstat = '{ "room" : "' + room + '", "body"
        ↪  : "';
3   var closetag ='"}';
4
5   /*
6    Masuk ke dalam room
7    */
8   socket.emit('join-room', room);
9
10
11  /*
12   Menerima riwayat perpesanan dan merender
         ↪ pesan-pesan tersebut ke view dengan fungsi
         ↪  bantu appendNewChat()
13   */
14  socket.on('chathistory', function(data){
15      console.log(data);
16      $(data).each(function(index,value){
17          appendNewChat(value);
18      });
19  });
20
```

```
21  $('.sendMessage').click(function(){
22   /*
23    constructing JSON message
24    concate previous statement with message's body
25    and close tag
26    */
27      var msg = prepstat + $('.body').val() +
            ↪ closetag ;
28
29      /* disable input pesan untuk sementara */
30      $(".body").attr("disabled", true);
31
32      /*kirimkan pesan */
33      socket.emit('send', msg );
34
35      /* menunggu status pengiriman pesan */
36      socket.on('send-status', function(data){
37       /* Jika gagal, enable input pesan dengan
             ↪ tidak menghapus pesan yang belum jadi
             ↪ dikirimkan sebelumnya */
38          if(!data.status){
39              swal('Oops','Pesan anda tidak dapat
                    ↪ terkirim, silahkan coba lagi','
                    ↪ error');
40          }
41          /* Jika sukses, enable input pesan dengan
                ↪  elemen input pesan yang sudah
                ↪ dikosongkan */
42          else if(data.status){
43              $('.body').val('');
44          }
45
46      }, function(){
47          $(".body").attr("disabled", false);
48      });
49  });
50
51  /*
```

```
52   Bagian ini dieksekusi saat ada pesan baru masuk
         ↪ ke dalam room. appendNewChat() adalah
         ↪ fungsi bantu merender view untuk
         ↪ menampilkan pesan baru
53   */
54  socket.on('new-msg', function (value) {
55      appendNewChat(value);
56  });
```

Kode Sumber IV.17 Kode Sumber Logika Pengiriman & Penerimaan Pesan (menggunakan jQuery)

4.3.14 Melihat Kotak Pesan

4.3.15 Memasukkan kupon

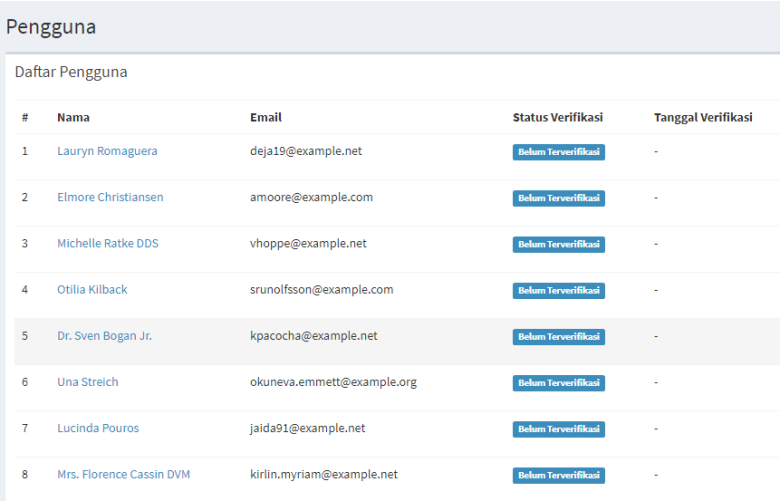
4.3.16 Melihat daftar pengguna

Halaman ini hanya dapat diakses oleh *administrator* yang sebelumnya sudah login. Detail kasus penggunaan dapat dilihat tabel spesifikasi kasus penggunaan **??**.

Tidak ada *view logic* ataupun logika *UI* khusus dalam halaman ini. Kode sumber implementasi *back-end* dapat dilihat pada kode sumber **??**.

```
1
2   /**
3    * File : app/Http/Controllers/UserController
4    * Menampilkan halaman daftar pengguna
5    * langsung diFetch dari base Model User
6    * Method : GET
7    */
8   public function user()
9   {
10      $data['data'] = User::paginate(20);
11      return view('pages.user.index', $data);
12  }
```

Kode Sumber IV.18 Kode Sumber Antarmuka Registrasi



Gambar 4.11 Halaman Antarmuka Kasus Penggunaan Melihat Daftar Pengguna

4.3.17 Melihat daftar laporan pengguna

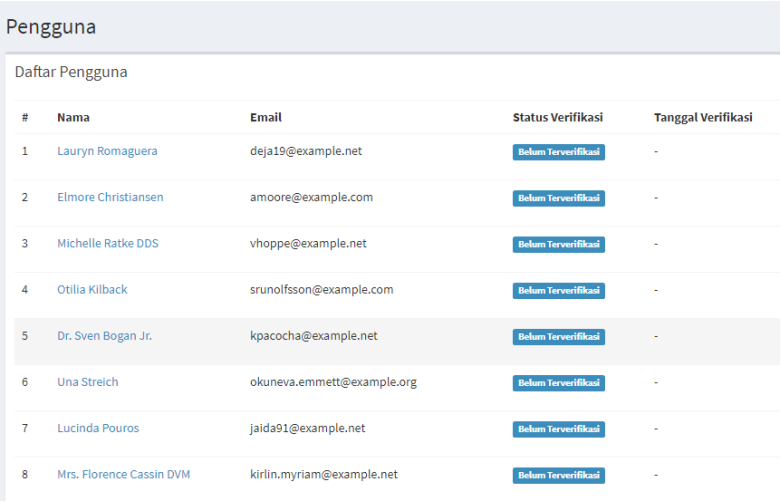
Halaman ini hanya dapat diakses oleh *administrator* yang sebelumnya sudah login. Detail kasus penggunaan dapat dilihat tabel spesifikasi kasus penggunaan **??**.

Tidak ada *view logic* ataupun logika *UI* khusus dalam halaman ini. Kode sumber implementasi *back-end* dapat dilihat pada kode sumber **??**.

```
1
2  /**
3   * File : app/Http/Controllers/UserController
4   * Menampilkan halaman daftar pengguna
```

```
5    * langsung diFetch dari base Model User
6    * Method : GET
7    */
8   public function user()
9   {
10      $data['data'] = User::paginate(20);
11      return view('pages.user.index', $data);
12  }
```

Kode Sumber IV.19 Kode Sumber Antarmuka Registrasi



Gambar 4.12 Halaman Antarmuka Kasus Penggunaan Melihat Daftar Pengguna

4.3.18 Memblock pengguna

Halaman ini hanya dapat diakses oleh *administrator* yang sebelumnya sudah login. Detail kasus penggunaan dapat dilihat tabel spesifikasi kasus penggunaan **??**.

Tidak ada *view logic* ataupun logika *UI* khusus dalam

halaman ini. Kode sumber implementasi *back-end* dapat dilihat pada kode sumber **??**.

```
1
2  /**
3   * File : app/Http/Controllers/UserController
4   * Menampilkan halaman daftar pengguna
5   * langsung diFetch dari base Model User
6   * Method : GET
7   */
8  public function user()
9  {
10     $data['data'] = User::paginate(20);
11     return view('pages.user.index', $data);
12 }
```

Kode Sumber IV.20 Kode Sumber Antarmuka Registrasi

4.3.19 Menambahkan voucher

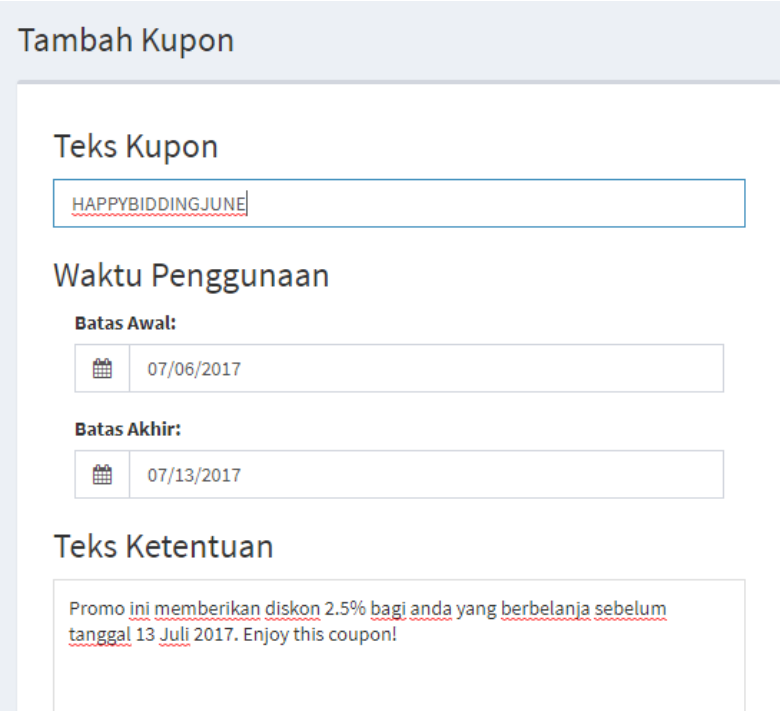
Halaman ini hanya dapat diakses oleh *administrator* yang sebelumnya sudah login. Halaman ini menampilkan form berisi elemen *input* informasi voucher, dan setelah selesai lalu mengklik tombol Tambahkan Voucher, dan untuk kasus alternatif dapat dilihat pada tabel spesifikasi kasus penggunaan **??**.

Tidak ada *view logic* ataupun logika *UI* khusus dalam halaman ini. Kode sumber implementasi *back-end* dapat dilihat pada kode sumber **??**.

```
1
2  /** File : app/Http/Controllers/CouponController
3   * Menampilkan halaman tambah kupon
4   * Method : GET
5   */
6  public function create()
7  {
```

```
8       $data['user'] = $this->userRepository->
            ↪ checkbox();
9       return view('pages.coupon.create',$data);
10  }
11
12  /**
13   * Store data kupon yang dimasukkan
14   * fungsi ini dijalankan ketika form disubmit
15   * terkait dengan CouponRepository
16   */
17
18  public function store(Request $request)
19  {
20      $data = $request->all();
21      $data['limit_usages'] = $data['usages'];
22      $ss = $this->couponRepository->storeCoupon(
            ↪ $data);
23
24      if($ss) return redirect('coupon')->with('
            ↪ success',true);
25      else return redirect('coupon')->with('success
            ↪ ',false);
26  }
27
28
29  /**
30   * File CouponRepository
31   */
32  public function storeCoupon($data)
33  {
34          /* Menggunakan base model Laravel */
35          /* return status store data ke dalam
                ↪ database */
36      return Coupon::create($data);
37  }
```

Kode Sumber IV.21 Kode Sumber Antarmuka Registrasi



Gambar 4.13 Halaman Antarmuka Kasus Penggunaan Menambahkan Voucher

4.3.20 Melihat daftar voucher

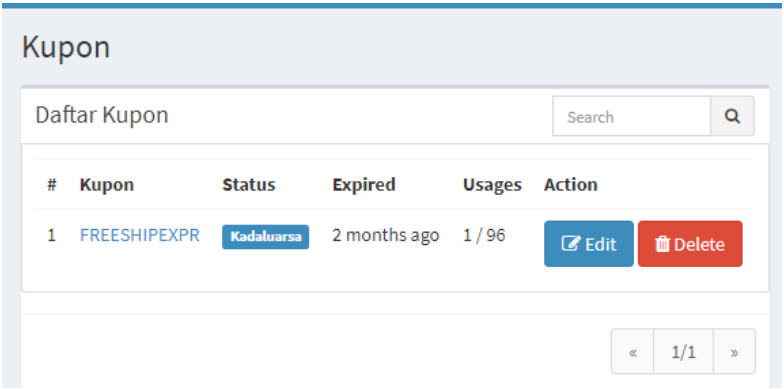
Halaman ini hanya dapat diakses oleh *administrator* yang sebelumnya sudah login. Detail kasus penggunaan dapat dilihat tabel spesifikasi kasus penggunaan **??**.

Tidak ada *view logic* ataupun logika *UI* khusus dalam halaman ini. Kode sumber implementasi *back-end* dapat dilihat pada kode sumber **??**.

```
1
2  /** File : app/Http/Controllers/CouponController
3   * Menampilkan halaman daftar kupon
```

```
4    * tidak terkait dengan CouponRepository
5    * langsung diFetch dari base Model Coupon
6    * Method : GET
7    */
8
9   public function index()
10  {
11      $data['data'] = Coupon::orderBy('activedate',
            ↪  'desc')->paginate(20);
12      return view('pages.coupon.index', $data);
13
14  }
```

Kode Sumber IV.22 Kode Sumber Antarmuka Registrasi



Gambar 4.14 Halaman Antarmuka Kasus Penggunaan Melihat Daftar Voucher

4.3.21 Memperbarui Voucher

Halaman ini hanya dapat diakses oleh *administrator* yang sebelumnya sudah login. Detail kasus penggunaan dapat dilihat tabel spesifikasi kasus penggunaan **??**.

Tidak ada *view logic* ataupun logika *UI* khusus dalam

halaman ini. Kode sumber implementasi *back-end* dapat dilihat pada kode sumber **??**.

```
1
2  /** File : app/Http/Controllers/CouponController
3   * Menampilkan halaman perbarui kupon
4   * Method : GET
5   */
6  public function edit($id)
7  {
8      try{
9          $data['user'] = $this->userRepository->
               ↪ checkbox();
10         $data['coupon'] = $this->couponRepository
               ↪ ->find($id);
11         return view('pages.coupon.edit', $data);
12     }
13     catch (\Exception $e){
14         abort(403);
15     }
16 }
17 /**
18  * Update data kupon yang dimasukkan
19  * fungsi ini dijalankan ketika form di halaman
       ↪ edit  kupon disubmit
20  * terkait dengan CouponRepository
21  */
22
23 public function update(Request $request, $id)
24 {
25
26     $data = $request->all();
27     if($this->couponRepository->update($id, $data
           ↪ )){
28         return redirect('coupon')->with('success
               ↪ ','Sukses memperbarui kupon!');
29     }
30     else return redirect('coupon')->with('error
```

```
            ↪ ','Mohon maaf, gagal memperbarui kupon.
            ↪  Silahkan coba lagi');
31  }
32
33
34  /**
35   * File CouponRepository
36   */
37  public function update($id, $data)
38  {
39      $coupon = $this->find($id);
40      return $coupon->update($data);
41
42  }
```

Kode Sumber IV.23 Kode Sumber Implementasi *Back-end* Kasus Penggunaan Memperbarui Voucher

(Halaman ini sengaja dikosongkan)

BAB V

PENGUJIAN DAN EVALUASI

5.1 Pengujian

Pada subbab ini, penulis akan memaparkan pengujian terhadap aplikasi. Pengujian yang dilakukan adalah pengujian fungsionalitas, dimana penulis menggunakan tool Laravel Dusk sebagai *testing code* untuk menguji fungsionalitas aplikasi.
Dikarenakan keterbatasan waktu, dan atas saran dari pembimbing, penulis tidak menuliskan *testing script* untuk keseluruhan fungsionalitas yang sudah pasti teruji, seperti Login (sudah menggunakan *facade* Laravel), transaksi CRUD dll.
Pada pemaparan ini, penulis mengidentifikasikan fungsionalitas utama dalam aplikasi lelang ini adalah sebagai berikut :

1. Pengujian Fungsionalitas Lelang
    (a) Pengujian Penawaran Lelang
2. Pengujian Fungsionalitas Voucher
    (a) Pengujian Penggunaan Voucher

Pada bagian ini juga, penulis menuliskan *summary* pengujian fungsionalitas ini pada .

5.1.1 Pengujian Fungsionalitas Lelang

Pada pengujian ini, terdapat beberapa skenario pengujian yang dipaparkan dalam tabel berikut :

5.1.2 Pengujian Fungsionalitas Voucher

Pada pengujian ini, terdapat beberapa skenario pengujian yang dipaparkan dalam tabel berikut :

5.1.3 *Summary* Pengujian

5.2 Evaluasi

Pada subbab ini, penulis akan memaparkan hasil analisa terhadap aplikasi, perspektif non-IT terhadap pengerjaan maupun lingkup pekerjaan dari aplikasi Lelang Online ini.

5.2.1 Evaluasi Aspek Kebutuhan

5.2.2 Evaluasi Aspek Teknis

5.2.3 Evaluasi Aspek *Bussiness Engineering*

5.2.3.1 Evaluasi Regulasi Terkait

5.2.3.2 *Market Analysis*

5.2.3.3 *Competitors Competitiveness*

5.2.3.4 Evaluasi *Startup Burn Rate*

5.2.4 Evaluasi Aspek Analisa dan Desain

5.2.5 Evaluasi Aspek Implementasi

5.2.6 Evaluasi Aspek Pengujian

5.2.7 *Summary* Evaluasi

5.2.8 *Further Enchancements*

5.2.9 *Further Readings*

BAB VI

PENUTUP

Bab ini membahas kesimpulan yang dapat diambil dari tujuan pembuatan sistem dan hubungannya dengan hasil uji coba dan evaluasi yang telah dilakukan. Selain itu, terdapat beberapa saran yang bisa dijadikan acuan untuk melakukan pengembangan dan penelitian lebih lanjut.

6.1 Kesimpulan

Dari proses perancangan, implementasi dan pengujian terhadap sistem, dapat diambil beberapa kesimpulan berikut:

1. Kesimpulan 1
2. Kesimpulan 2
3. Kesimpulan 3

6.2 Saran

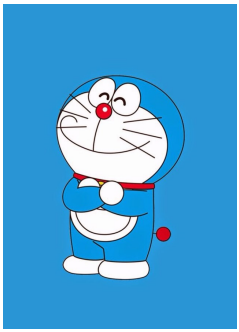
Berikut beberapa saran yang diberikan untuk pengembangan lebih lanjut:

- Menggunakan mekanisme *Queue* sebagai *countermeasure* dari masalah *occurence* (di Laravel sudah ada disediakan *base class*nya sendiri).
- Mengikutsertakan pihak yang menguasai/spesialisasi di bidang hukum untuk menetapkan peraturan-peraturan terkait
- Saran 2
- Saran 3

(Halaman ini sengaja dikosongkan)

BIODATA PENULIS

Ronauli



Silva NS, seorang yang lahiran & besar di Siantar Medan - Sumatera Utara, sangat suka belajar. Diberi amanah untuk menjadi *administrator* Laboratorium Pemrograman di tahun 2015, penulis belajar banyak mengenai administrasi *server*, rancang bangun aplikasi terutama di bidang web. Selain itu, beberapa *project* yang diambil penulis mengenai rancang bangun aplikasi yang baik dan buruk yang mengajarkan penulis cara memperbaiki, menangkal dan & mengoptimasinya. Selain itu, penulis juga banyak belajar *softskills* saat menjabat sebagai Sekretaris Departemen HMTC ITS 2015/2016 (Kabinet Optimasi) dan lewat pelatihan-pelatihan yang diberikan donatur beasiswa saat penulis masih diamanahi sebagai beswan Karya Salemba Empat 2014-2016.

Motto penulis yaitu "*Always go for the extra miles*", membawa penulis mengambil topik tugas akhir ini, dimana penulis dapat menerapkan perbaikan, optimasi dan pelajaran yang penulis petik dari *project-project* sebelumnya, dengan bimbingan dosen-dosen pembimbing penulis yang baik hati. Dalam pendalaman topik tugas akhir ini juga, penulis banyak belajar dan menjadi sangat tertarik mendalami *bussiness engineering*, *user experiences and usability*, dan *data scientist & engineering*.

Dengan segala kerendahan hati, ilmu penulis masihlah setitik dibandingkan susu sebelanga. Penulis sangat mengharapkan diskusi, ajaran dan bantuan dalam memperbaiki diri. Apabila pembaca berkenan, penulis dapat dihubungi melalui *email* ke `ronayumik@gmail.com`, atau *Whatsapp* ke nomor +62821-6066-5568.