UNIVERSITAS INDONESIA

REALTIME AUDIO TRANSCRIPTION

LAPORAN KERJA PRAKTIK

RAKINA ZATA AMNI

1306398951

FAKULTAS ILMU KOMPUTER

PROGRAM STUDI ILMU KOMPUTER

DEPOK

OKTOBER 2016

HALAMAN PERSETUJUAN DOSEN MATA KULIAH KERJA PRAKTIK

Laporan ini diajukan oleh	:	
Nama	:	Rakina Zata Amni
NPM	:	1306398951
Program Studi	:	Ilmu Komputer
Judul Kerja Praktik	:	*Realtime Audio Transcription*

Telah berhasil diselesaikan laporan kerja praktik untuk fakultas dan dipresentasikan hasil kerja praktiknya dalam forum seminar kerja praktik sebagai persyaratan yang harus dipenuhi dalam mata kuliah Kerja Praktik.

DOSEN MATA KULIAH KERJA PRAKTIK	**PESERTA KERJA PRAKTIK**
(Meganingrum Arista Jiwanggi, S.Kom., M.Kom.)	**(Rakina Zata Amni)**

Ditetapkan di : Depok

Tanggal : [TANGGAL]

ABSTRAK

Nama: Rakina Zata Amni
Program Studi: Ilmu Komputer
Judul: *Realtime Audio Transcription*

Laporan kerja praktik ini berisi hasil dari kerja praktik yang telah penulis lakukan bersama penyelia di Google Inc selama dua belas minggu. Penulis berperan sebagai *software engineering intern* dengan penyelia Amit Dubey yang bekerja sebagai *software engineer* di Google. Dalam proyek kerja praktik yang diberikan oleh pihak penyelia, penulis membuat sistem *audio transcription* yang dapat menerima *audio stream* dan mengembalikan kumpulan teks hasil transkripsi dari *stream* tersebut. Proyek kerja praktik ini dibawahi oleh tim yang berada di dalam departemen *Research and Machine Intelligence* di Google dan nantinya akan digunakan dalam upaya meningkatkan performa pengolahan bahasa manusia di Google. Selama kerja praktik, penulis memperoleh banyak pengalaman menjadi seorang *software engineer*, khususnya di perusahaan besar seperti Google dan di daerah yang menjunjung tinggi bidang teknologi seperti San Francisco Bay Area (Silicon Valley).

Kata kunci: *audio transcription,* Google, kerja praktik

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR LAMPIRAN

BAB 1
PENDAHULUAN

1.1. Proses Pencarian Kerja Praktik

Penulis mulai mencari lowongan kerja praktik di akhir bulan Agustus 2015. Karena sebelumnya penulis sudah pernah melakukan *internship* di sebuah perusahaan di San Francisco, penulis memulai dengan melamar pekerjaan di beberapa perusahaan teknologi yang ada di daerah San Francisco Bay Area (Silicon Valley) lagi dengan cara mengirimkan *resume* penulis di halaman karir beberapa perusahaan tersebut.

Setelah beberapa minggu, penulis mendapat respon dari beberapa perusahaan untuk menjadwalkan wawancara. Salah satu perusahaan yang merespon lamaran kerja penulis adalah Google. Penulis melakukan wawancara dengan dua orang *software engineer* Google di pertengahan bulan Oktober 2015. Format kedua wawancara ini sama, yaitu 45 menit yang berisi sedikit perkenalan, dua buah pertanyaan pemrograman, dan sesi tanya jawab tentang pengalaman pewawancara di Google. Dua minggu setelah wawancara dilaksanakan, penulis mendapatkan kabar dari perekrut dari Google bahwa penulis diterima sebagai *software engineering intern* di kantor pusat Google di Mountain View, California, Amerika Serikat.

Penulis mendapat beberapa tawaran magang juga dari beberapa perusahaan lain seperti Uber dan Square, namun akhirnya penulis memilih untuk menerima tawaran Google. Setelah menerima tawaran tersebut, penulis lalu menunggu kabar tentang tim dimana penulis akan ditempatkan. Penulis juga mengontak beberapa kenalannya yang bekerja di Google untuk mencari tim yang sesuai, dan salah satu kenalan penulis yang bekerja di bawah departemen *Research and Machine Intelligence* di Google memiliki kenalan di departemen yang sama yang punya lowongan untuk seorang *intern* di timnya. Tim tersebut akhirnya menjadi pilihan penulis.

1.2 Tempat Kerja Praktik

Penulis melaksanakan kerja praktik di kantor pusat Google Inc. di Mountain View, California, Amerika Serikat.

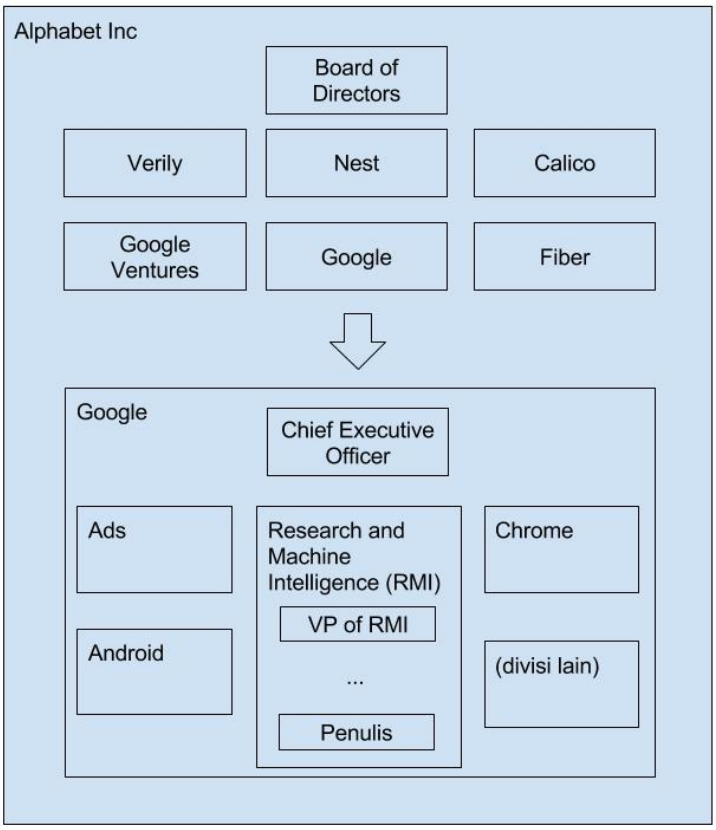
1.2.1. Profil Tempat Kerja Praktik

Google Inc. adalah perusahaan multinasional Amerika Serikat yang berada di bawah Alphabet Inc. dan didirikan pada tahun 1998 oleh Larry Page dan Sergey Brin. Google Inc. bergerak di bidang jasa dan produk internet, dan memiliki berbagai macam produk seperti mesin pencari, sistem operasi, perangkat lunak, serta periklanan.

Secara struktur, Google dibagi menjadi beberapa departemen sesuai pekerjaan dari tiap departemen, seperti departemen *engineering,* departemen legal/hukum, departemen sumber daya manusia, dan sebagainya. Departemen terbesar di Google adalah departemen *engineering,* yang dibagi menjadi beberapa sub departemen lagi di bawahnya. Beberapa sub departemen ini berkaitan deangan produk yang dikembangkan di bawah sub departemen tersebut, seperti *Ads*, *Geo*, *Research and Machine Intelligence*, atau *YouTube*. Di bawah subdepartemen tersebut, terdapat banyak tim besar yang berisi tim-tim kecil yang lebih terfokus. Tiap tim memiliki *engineering manager* serta beberapa *software engineer.* Tiap karwayan di Google memiliki *level* masing-masing, yang menyatakan seberapa berpengalaman orang tersebut di bidang pekerjaannya.

1.2.2. Posisi Penempatan Pelaksana Kerja Praktik dalam Struktur Organisasi

Pada kerja praktik ini, penulis ditempatkan sebagai *software engineering intern* di bawah salah satu tim di bawah departemen *engineering,* di sub departemen *Research and Machine Intelligence.* Tim tempat penulis ditempatkan memiliki tujuan meningkatkan kemampuan Google dalam mengolah bahasa manusia.



Gambar 1.1 Penempatan penulis dalam Google Inc

BAB 2
ISI

2.1 Pekerjaan dalam Kerja Praktik

Pegawai magang di Google biasanya diberikan suatu proyek kecil yang berkaitan dengan tim dimana pegawai magang tersebut ditempatkan. Proyek ini diharapkan dapat selesai diimplementasi di akhir masa kerja pegawai magang tersebut. Karena penulis ditempatkan di sub departemen *Research and Machine Intelligence*, proyek yang diberikan kepada penulis adalah proyek yang berkaitan dengan bidang tersebut.

2.1.1. Latar Belakang Pekerjaan

Google menawarkan banyak produk dan jasa bagi para penggunanya, seperti Google Search, Google Maps, Google Mail, YouTube serta beberapa produk dan jasa lainnya. Kebanyakan produk tersebut menerima masukan berupa bahasa manusia, seperti permintaan pencarian di Google Search, YouTube dan Google Maps atau pengategorian topik surat elektronik di Google Mail. Karena itu, pengolahan bahasa manusia atau *Natural Language Processing* adalah topik yang cukup penting bagi Google agar dapat mengolah data dari pengguna dengan lebih baik. Pengolahan bahasa manusia mencakup topik-topik seperti *Natural Language Understanding* yang bertujuan untuk mengerti bahasa manusia dan *Natural Language Generation* yang bertujuan untuk membangkitkan atau membuat kalimat-kalimat dalam bahasa manusia.

Google memiliki beberapa sub departmen di bawah departemen *engineering.* Sub departemen yang membawahi bidang *Natural Language Processing* di Google adalah sub departemen *Research and Machine Intelligence*. Penulis ditempatkan di sebuah tim *Natural Language Processing* di bawah sub departemen tersebut yang bertugas untuk mengembangkan sebuah aplikasi yang digunakan untuk mengumpulkan data dialog atau pembicaraan untuk meningkatkan kemampuan Google dalam bidang *Natural Language Generation.*

Aplikasi yang dikembangkan oleh tim ini berperan seperti asisten pribadi atau *Natural Language Assistant* bagi pengguna. Aplikasi ini mewajibkan pengguna untuk

berinteraksi dengan aplikasinya menggunakan suara. Respon dari aplikasi berupa suara yang dibangkitkan dengan program *text-to-speech*. Dengan ini, Google bisa mendapatkan data berupa dialog antara pengguna dan aplikasi. Menurut penyelia kerja praktik penulis, data percakapan adalah data yang cukup berharga karena jumlahnya saat ini tidak terlalu banyak dan dapat membantu Google dalam membuat produk-produk yang memiliki bahasa yang lebih alami.

Saat penulis memulai kerja praktik, data suara manusia yang diterima disimpan secara utuh dan belum diproses atau ditranskripsikan. Agar data tersebut dan seluruh data yang didapatkan selanjutnya dapat segera digunakan, penulis ditugaskan untuk membuat sistem yang menerima data suara manusia secara *real-time* dan mentranskripsikannya. Data hasil transkripsi ini nantinya akan diolah dan menjadi data latihan untuk algoritma *Natural Language Processing* milik Google.

Proyek yang diberikan kepada penulis tidak mencakup pembuatan algoritma transkripsi, karena Google sudah memiliki jasa transkripsi sendiri yaitu Google Cloud Speech API [6]. Sistem yang dibuat penulis berfungsi sebagai penyambung antara aplikasi utama dengan Google Cloud Speech API dan harus mampu menerima banyak arus transkripsi dari beberapa percakapan berbeda sekaligus serta menangani transkripsi secara *real-time* yang rawan akan *latency*. Proyek yang diberikan penulis juga mencakup integrasi dengan aplikasi utama, seperti membuat arus komunikasi antara aplikasi utama dan sistem yang dibuat dan menampilkan hasil transkripsi di aplikasi utama secara *real-time* juga.

2.1.2 Metodologi

Proyek kerja praktik yang dibuat oleh penulis adalah proyek baru dan belum pernah diimplementasikan sebelumnya. Terdapat beberapa alternatif cara untuk mengimplementasikan proyek tersebut, namun belum ada cara yang dicoba. Saat memulai, penyelia kerja praktik memberitahu bahwa beberapa alternatif cara yang lebih gampang mungkin saja tidak dapat diimplementasikan karena batasan dari teknologi yang digunakan oleh *server* aplikasi utama. Karena itu, penulis ditugaskan terlebih dahulu untuk membuat versi kecil dari sistem yang ingin dibangun sebagai *proof-of-*

concept dan mengetahui batasan-batasan apa yang dapat menghalangi pengembangan sistem.

Penulis memulai dengan membangun sistem yang menjadi bagian dari aplikasi utama dan berada sebagian besar di bagian *front-end,* lalu mencoba membangun sistem yang juga menjadi bagian dari aplikasi utama namun berada di bagian *back-end*, dan akhirnya membangun sistem terpisah dari aplikasi utama. Untuk setiap halangan baru yang penulis temukan dalam proses pengembangan sistem-sistem tersebut, penulis mendiskusikannya dengan penyelia kerja praktik untuk mencari cara untuk menyelesaikannya.

Setelah melakukan eksplorasi dan mengetahui sistem seperti apa yang akan dibangun, penulis memulai pengembangan sistem secara bertahap. Setiap tahap berisi satu atau lebih fitur yang ingin dikembangkan dan diselesaikan dalam beberapa hari, yang diakhiri dengan tahap *code review* dari penyelia kerja praktik dan anggota tim lainnya.

Selain mengerjakan proyek utama kerja praktik, penulis juga ditugaskan untuk mengerjakan beberapa fitur kecil dan *bugfix* di beberapa bagian dari aplikasi utama. Penulis mengerjakan pekerjaan-pekerjaan sampingan tersebut ketika penulis menunggu hasil dari *code review* untuk proyek utama milik penulis, agar tetap menghabiskan waktu dengan produktif.

2.1.3 Teknologi yang Digunakan

Dalam pengerjaan proyek kerja praktik, penulis mendapatkan banyak pengetahuan baru tentang teknologi-teknologi yang sebelumnya belum pernah dipakai oleh penulis seperti gRPC [4], Google *App Engine* [2], React [1] dan Google *Cloud Speech* API [6]. Penulis juga mendapat kesempatan untuk mendalami teknologi-teknologi yang sebelumnya tidak banyak digunakan oleh penulis seperti JavaScript dan Java.

Aplikasi utama yang dikembangkan oleh tim yang menampung penulis memiliki tiga bagian utama, yaitu *backend* API*, front end client,* serta aplikasi Android. *Backend* API adalah sekumpulan REST API yang dikembangan dengan bahasa Java dan menggunakan teknologi JavaServer Pages (JSP), aplikasi Android dikembangkan

dengan bahasa Java, dan *front end client* dibuat dengan JavaScript dan *framework* React. *Server* dari aplikasi utama ini di*deploy* di Google App Engine.

Aplikasi Android berperan sebagai asisten pribadi yang dapat mendengarkan suara pengguna. Aplikasi kemudian akan mengirimkan masukan suara pengguna ke *front end client* yang akan memainkan suara pengguna secara *real-time* menggunakan *Web Audio* API. *Front end client* juga mengirimkan berkas suara pengguna tersebut ke *endpoint* penyimpanan berkas di *backend* API menggunakan AJAX untuk disimpan di basis data. Karena keterbatasan basis data Google App Engine, berkas data hanya bisa memiliki panjang sekitar satu detik, sehingga sebuah percakapan dapat terdiri dari banyak potongan suara (*audio chunk)*. Proses pemotongan berkas audio menjadi potongan kecil menggunakan *library* JavaScript bernama Recorderjs[1], dan menghasilkan sebuah potongan audio setiap satu detik dalam format berkas WAV.

Untuk proyek utama penulis, terdapat tiga alternatif pengerjaan yang semuanya dicoba oleh penulis saat tahap eksplorasi dan pembuatan *proof-of-concept*. Ketiga metode tersebut sama-sama menggunakan Google *Cloud Speech* API, yaitu API yang disediakan oleh Google yang dapat menerima masukan berupa berkas audio dan mengembalikan hasil berupa transkripsi dari berkas audio tersebut. Ketiga metode tersebut juga sama-sama mendapatkan berkas audio yang akan ditranskripsikan dari aplikasi utama.

Metode pertama yang penulis implementasikan adalah membuat sistem *real-time transcription* menjadi bagian dari *front end client* aplikasi utama. Penulis menambahkan *request* AJAX baru ke *front end client,* sehingga setelah menerima berkas audio dari aplikasi Android, *front end client* bisa mengirimkan berkas audio tersebut ke Google Cloud Speech API. Google Cloud Speech API sendiri memiliki dua jenis protokol, yaitu HTTP dan RPC. Karena request AJAX menggunakan protokol HTTP, penulis menggunakan API versi HTTP di implementasi ini.

---

[1] https://github.com/mattdiamond/Recorderjs

Metode kedua yang penulis implementasikan adalah membuat sistem *real-time transcription* menjadi bagian dari *backend* API aplikasi utama. Penulis memodifikasi bagian dari *backend* API yang menerima berkas audio yang dikirimkan oleh *front end client*, agar mengirimkan berkas yang diterima tersebut ke versi RPC dari *Google Cloud Speech* API. Penulis menggunakan gRPC untuk melakukan komunikasi *bidirectional streaming* dengan *endpoint* RPC dari Google *Cloud Speech* API. Tidak seperti versi HTTP, *endpoint* RPC ini bersifat *asynchronous* dan *connection-based*. Ini berarti sistem yang dibuat penulis dapat mengirimkan berkas audio kapan pun ke sebuah koneksi yang sama, dan Google *Cloud Speech* API juga dapat mengirimkan hasil transkripsi kapan pun lewat koneksi tersebut.

Metode ketiga yang penulis implementasikan mirip dengan metode kedua, namun sistem *real-time transcription* yang dibuat tidak menjadi bagian dari aplikasi utama. Aplikasi utama di*deploy* di Google *App Engine*, yang menurut penyelia kerja praktik memiliki banyak batasan yang mungkin membuat metode kedua tidak dapat dilakukan. Sistem yang dibuat dengan metode ketiga berdiri sendiri dan tidak akan di*deploy* di Google App Engine. Sistem terdiri dari sebuah *endpoint* yang menjadi titik komunikasi antara aplikasi utama dengan sistem ini untuk mengirimkan berkas audio dan menerima transkripsi. Sistem menggunakan *framework* milik Google dan dibuat dalam bahasa Java.

Di dalam sistem, terdapat program utama yang mengatur pemanggilan Google *Cloud Speech* API untuk tiap percakapan. Komunikasi dilakukan dengan menggunakan gRPC, dan tiap percakapan memiliki koneksi masing-masing. Tiap percakapan memiliki sebuah *handler* yang berguna untuk mengatur koneksi, mengirim berkas audio, dan menerima hasil transkripsi dari percakapan tersebut. Tiap *handler* memiliki sebuah *BlockingQueue* [3] untuk menampung hasil transkripsi dari Google Cloud Speech API untuk percakapan tersebut.Program utama memiliki sebuah *BlockingQueue* terpusat untuk seluruh berkas audio yang diterima dan belum dikirimkan. Program utama akan menunggu sampai *BlockingQueue* tidak kosong, dan mengirimkan satu persatu berkas audio yang berada di *BlockingQueue* tersebut ke *handler* untuk tiap percakapan. Penulis

menggunakan struktur data *BlockingQueue* karena struktur data yang menampung berkas audio maupun transkripsi mungkin diakses oleh banyak *thread* sekaligus.

Setiap *request* yang dikirimkan ke *server* berisi berkas audio dan ID percakapan dari berkas tersebut dalam bentuk Protobuf [5] dan melalui protokol HTTP. Isi dari *request* ini kemudian dimasukkan ke BlockingQueue milik program utama agar dapat diproses. Respons untuk *request* ini adalah isi dari BlockingQueue milik *handler* untuk percakapan yang bersangkutan. Setelah diambil isinya, BlockingQueue yang bersangkutan akan menjadi kosong.

Selain membuat sistem, penulis juga diharuskan mengintegrasikan sistem tersebut dengan aplikasi utama. Penulis membuat integrasi di bagian *backend* dan *frontend*. Di bagian *backend*, penulis menambahkan pemanggilan ke sistem yang dibuat penulis dari API endpoint milik aplikasi utama yang menerima berkas audio untuk disimpan. Setelah berkas audio disimpan, penulis menambahkan konversi berkas audio yang didapat dari format WAV menjadi format RAW dan mengirimkan berkas audio beserta ID dari percakapan yang bersangkutan. Konversi berkas audio diperlukan karena Google *Cloud Speech* API tidak dapat menerima berkas dalam bentuk WAV. Konversi dilakukan secara manual dengan manipulasi representasi biner dari berkas audio. Di bagian *front-end*, penulis menambahkan tampilan yang dapat menunjukkan hasil transkripsi berkas audio secara *real-time* dengan menggunakan JavaScript, React dan JSDoc [7].

2.1.4 Hasil Pekerjaan

Setelah mencoba mengimplementasikan *proof-of-concept* dari metode pertama, penulis menemukan bahwa versi HTTP dari Google Cloud Speech API tidak dapat menerima *request* dalam bentuk *stream.* Google Cloud Speech API versi HTTP memiliki konsep *stateless*, yang berarti bahwa versi ini hanya dapat mentranskripsikan berkas audio dari satu buah *request*, dan tidak bisa menggunakan informasi dari berkas audio yang lain yang dikirim oleh pengguna yang sama. Tentunya ini sangat bermasalah, karena *front end client* mengirimkan berkas audio sepanjang satu detik di tiap *request*.

Solusi yang mungkin dari masalah tersebut adalah menggabungkan berkas audio menjadi satu sampai cukup panjang lalu mentranskripsikannya, tapi ini berarti hasil yang didapatkan tidak akan *real-time*. Solusi lain adalah menggunakan versi RPC dari Google *Cloud Speech* API, yang mendukung *streaming*. Penulis mencari cara untuk menggunakan RPC di JavaScript, namun ternyata teknologi yang ada sekarang belum mendukung hal tersebut. Penulis kemudian berdiskusi dengan penyelia kerja praktik tentang masalah ini. Penulis dan penyelia akhirnya sepakat bahwa tidak ada solusi yang baik untuk masalah ini dan melanjutkan ke metode kedua.

Penulis kemudian mengimplementasikan metode kedua secara lokal dan berhasil membuat program *proof-of-concept.* Program yang dibuat penulis berhasil mengirimkan berkas audio yang disimpan di komputer ke Google *Cloud Speech* API dan mendapatkan hasil transkripsi dari berkas audio tersebut. Namun saat program tersebut di-*deploy* di Google *App Engine*, program tidak berhasil berjalan karena Google *App Engine* tidak memiliki support untuk pemanggilan RPC. Setelah berdiskusi dengan penyelia kerja praktik, penulis dan penyelia memutuskan untuk melanjutkan ke metode ketiga.

Program yang ditulis untuk metode kedua kemudian dipindahkan menjadi bagian dari sebuah sistem baru. Karena proses *deployment* sebuah sistem baru di Google harus melewati beberapa tahap *review* yang akan memakan waktu cukup lama, penulis dan penyelia memutuskan untuk menyelesaikan sistem sampai sempurna terlebih dahulu dan penyelia akan melakukan *deployment* setelah masa kerja praktik selesai. Karena itu, seluruh proses pengembangan dilakukan penulis di *server* lokal.

Penulis melakukan pengembangan sistem secara bertahap. Pada tahap pertama, penulis hanya membuat *proof-of-concept* seperti program *proof-of-concept* untuk metode kedua, namun dalam bentuk sistem yang berdiri sendiri. Setelah itu, penulis melakukan penyempurnaan sistem secara bertahap. Penyempurnaan yang dilakukan antara lain adalah *refactoring*, menambah dukungan untuk mentranskripsikan percakapan yang terdiri dari beberapa potongan atau *chunk*, membuat integrasi sistem dengan aplikasi

utama, menambah dukungan untuk menangani banyak percakapan di suatu waktu, serta *testing*. Setiap tahap penyempurnaan melalui tahap *code review*.

Di minggu terakhir kerja praktik, seluruh fitur yang diharapkan ada sudah berhasil diimplementasikan, *test coverage* untuk sistem yang dibuat sudah lebih dari 80 persen, dan integrasi sistem dengan aplikasi utama sudah berjalan dengan baik. Penulis, penyelia kerja praktik, dan anggota tim yang lain menguji sistem dengan cara bergantian berbicara dengan mikrofon dan sistem berhasil berjalan dengan baik dan menampilkan hasil transkripsinya di aplikasi utama.

2.2 Analisis

Setelah pelaksanaan kerja praktik selesai, penulis menganalisis pelaksanaan kerja praktik yang dijalani penulis serta relevansi mata kuliah di Fasilkom UI dengan pekerjaan penulis di masa kerja praktik.

2.2.1. Pelaksanaan Kerja Praktik

Terdapat beberapa ketidaksesuaian antara pelaksanaan kerja praktik dengan perencanaan yang tertulis di Kerangka Acuan Kerja Praktik. Pada awalnya, penulis dan penyelia kerja praktik mengira sistem yang akan dibuat akan cukup sederhana dan dapat diselesaikan dalam beberapa minggu. Sistem dijadwalkan selesai dalam waktu lima sampai tujuh minggu, yang kemudian akan diikuti dengan peluncuran sistem dan perolehan serta pengolahan data transkripsi yang dihasilkan sistem.

Pada kenyataannya, pengembangan sistem yang menjadi proyek utama kerja praktik penulis memakan waktu yang cukup banyak karena menemui beberapa kendala teknis dalam pengerjaannya. Kendala ini baru diketahui saat penulis mulai mencoba bereksperimen di minggu-minggu awal kerja praktik karena proyek kerja praktik penulis belum pernah dicoba diimplementasikan sebelumnya. Eksperimen dan pengembangan sistem menghabiskan waktu sampai akhir periode kerja praktik.

Selain itu, penulis juga mengerjakan beberapa tugas kecil seperti *bugfix* minor di aplikasi utama yang sebelumnya tidak ada di KAKP. Penulis merasa puas bisa melakukan beberapa pekerjaan kecil tersebut karena bisa mengisi waktu penulis dengan

produktif dan memberikan pengalaman di bidang yang sebelumnya jarang ditekuni penulis seperti pengembangan dan pengujian *frontend* serta aplikasi Android.

Pelaksanaan kerja praktik di Google Inc. menambah wawasan penulis di berbagai bidang disiplin di dunia pengembangan perangkat lunak. Banyak sekali pelajaran baru yang penulis dapatkan selama pelaksanaan kerja praktik, baik pelajaran teknis ataupun non teknis yang bersumber dari penyelia kerja praktik penulis, anggota tim dimana penulis ditempatkan, sumber pembelajaran internal dari perusahaan, maupun dari sesama pelaku kerja praktik di perusahaan tempat kerja praktik penulis.

2.2.2 Relevansi dengan Perkuliahan di Fasilkom UI

Pada proyek kerja praktik yang penulis kerjakan, terdapat cukup banyak relevansi dengan perkuliahan di Fasilkom UI yang pernah diikuti penulis.

Dalam mata kuliah DDP, penulis mempelajari pemrograman dalam bahasa Java yang merupakan bahasa yang digunakan di proyek kerja praktik penulis. Penulis juga mempelajari konsep-konsep *Object-oriented Programming* yang merupakan konsep yang sangat penting dan dipakai dalam perancangan sistem yang saya buat.

Dalam mata kuliah PPW, penulis mempelajari dasar-dasar pemrograman berbasis *web.* Karena sistem yang dibuat penulis berbentuk *web service* dan penulis juga mengerjakan beberapa tugas di bagian *frontend* aplikasi utama yang berbasis web*,* ilmu yang didapatkan di perkuliahan tersebut banyak digunakan, terutama penggunaan JavaScript dan AJAX.

Dalam mata kuliah RPL, penulis mempelajari beberapa *design pattern* dan konsep-konsep penting dalam perancangan sebuah perangkat lunak. Pengetahuan penulis akan hal-hal ini sangat membantu penulis dalam merancang sistem yang dibuat penulis di kerja praktik ini. Penulis juga membuat beberapa laporan sebagai syarat peluncuran sebuah sistem di Google Inc. dengan format yang mirip dengan beberapa tugas di perkuliahan RPL.

Dalam mata kuliah Sistem Operasi (OS) dan Pemrograman Sistem, penulis mempelajari dasar-dasar penggunaan sistem operasi berbasis UNIX dan cara melakukan hal-hal yang berkaitan dengan pemrograman di sistem tersebut. Penulis memanfaatkan pengetahuan dan pengalaman dalam menggunakan sistem operasi berbasis UNIX untuk melakukan pekerjaan sehari-hari. Penulis juga menggunakan sedikit pengetahuan *scripting* saat melakukan beberapa eksperimen dengan berkas audio.

BAB 3
PENUTUP

3.1 Kesimpulan

Google Inc. merupakan tempat yang sangat baik untuk menjadi tujuan melaksanakan kerja praktik dan melanjutkan karir setelah lulus. Google Inc., memberikan sarana yang kondusif agar karyawannya mampu mengerjakan pekerjaan dengan baik. Selain itu, seluruh karyawan Google yang berinteraksi dengan penulis sangat terbuka untuk membantu dan mengajarkan penulis hal-hal baru dan kiat-kiat untuk memiliki karir yang suksed di bidang teknologi. Di Google Inc., penulis mendapat banyak pengalaman dan pengetahuan dari lingkungan kerja dan pekerjaan yang penulis lakukan.

3.2 Saran

Penulis menyarankan untuk mahasiswa atau mahasiswi yang akan mengambil kerja praktik untuk giat mencari lowongan tidak hanya di dalam negeri. Menurut penulis, mendapatkan pekerjaan untuk magang di luar negeri lebih mudah daripada mendapatkan pekerjaan tetap di luar negeri, dan pengalaman yang didapatkan baik secara profesional maupun secara pribadi akan jauh lebih menarik daripada di dalam negeri.

DAFTAR REFERENSI

(1) *A JavaScript library for building user interfaces - React.* (2016). Facebook.github.io. Diakses 27 November 2016, from https://facebook.github.io/react/

(2) *App Engine - Platform as a Service | Google Cloud Platform*. (2016). Google Cloud Platform. Retrieved 27 November 2016, from https://cloud.google.com/appengine/

(3) *BlockingQueue (Java Platform SE 7)*. (2016). Docs.oracle.com. Retrieved 27 November 2016, from https://docs.oracle.com/javase/7/docs/api/java/util/concurrent/BlockingQueue.html

(4) *grpc / About*. (2016). Grpc.io. Retrieved 27 November 2016, from http://www.grpc.io/about/

(5) *Protocol Buffers | Google Developers.* (2016). Google Developers. Retrieved 27 November 2016, from https://developers.google.com/protocol-buffers/

(6) *Speech API - Speech Recognition | Google Cloud Platform*. (2016). Google Cloud Platform. Retrieved 27 November 2016, from https://cloud.google.com/speech/

(7) *Use JSDoc: Index*. (2016). Usejsdoc.org. Retrieved 27 November 2016, from http://usejsdoc.org/

LAMPIRAN 1
KERANGKA ACUAN KERJA PRAKTIK

LAMPIRAN 2
LOG KERJA PRAKTIK