

**Analisis Desain Berbasis Arsitektur Microservices dengan
Representational State Transfer (REST)
(Studi Kasus: Heartenly.Com)**

Tugas Akhir

diajukan untuk memenuhi salah satu syarat

memperoleh gelar sarjana

dari Program Studi Teknik Informatika

Fakultas Informatika

Universitas Telkom

1301178327

Kautsar Tisamawi



Program Studi Sarjana Teknik Informatika

Fakultas Informatika

Universitas Telkom

Bandung

2019

LEMBAR PENGESAHAN

ANALISIS DESAIN BERBASIS ARSITEKTUR MICROSERVICE DENGAN REPRESENTATIONAL STATE TRANSFER (REST) (STUDI KASUS: HEARTENLY.COM)

Design Analysis Based on Microservice Architecture with Representational State
Transfer (REST)

(Case Study: Heartenly.com)

NIM : 1301178327

KAUTSAR TISAMAWI

Tugas akhir ini telah diterima dan disahkan untuk memenuhi sebagian syarat memperoleh
gelar pada Program Studi Sarjana Teknik Informatika
Fakultas Informatika
Universitas Telkom

Bandung, 23 Juli 2019

Menyetujui

Pembimbing I,



Dana S. Kusumo, S.T., M.T., Ph.D.

02780291

Pembimbing II,



Shinta Yulia Puspitasari S.T., M.T.

18800124

Ketua Program Studi
Sarjana Teknik Informatika,



Niken Dwi Wahyu Cahyani, S.T., M.Kom., Ph.D.

NIP: 00750052

LEMBAR PERNYATAAN

Dengan ini saya, Kautsar Tisamawi, menyatakan sesungguhnya bahwa Tugas Akhir saya dengan judul *Analisis Desain berbasis Arsitektur Microservice dengan Representational State Transfer (REST) (Studi Kasus: Heartenly.Com)* beserta dengan seluruh isinya adalah merupakan hasil karya sendiri, dan saya tidak melakukan penjiplakan yang tidak sesuai dengan etika keilmuan yang berlaku dalam masyarakat keilmuan. Saya siap menanggung resiko/sanksi yang diberikan jika di kemudian hari ditemukan pelanggaran terhadap etika keilmuan dalam buku TA atau jika ada klaim dari pihak lain terhadap keaslian karya,

Bandung, 23 Juli 2019

Yang Menyatakan



Kautsar Tisamawi

Analisis Desain Berbasis Arsitektur Microservice dengan Representational State Transfer (Rest) (Studi Kasus: Heartenly.Com)

Kautsar Tisamawi¹, Dana Sulistiyo Kusumo S.T., M.T., Ph.D², Shinta Yulia Puspitasari S.T., M.T.³

^{1,2,3}Fakultas Informatika, Universitas Telkom, Bandung

¹ktisamawi@students.telkomuniversity.ac.id, ²pembimbing danakusumo@telkomuniversity.ac.id, ³pembimbing shintayulia@telkomuniversity.ac.id

Abstrak

Arsitektur monolitik mendesain sistem aplikasi hanya terdiri dari satu bagian besar dan berjalan pada satu *computational instance* [1]. Hal tersebut menyebabkan, proses meningkatkan atau memperbaiki sistem aplikasi harus secara menyeluruh sekecil apapun peningkatan yang dilakukan. Heartenly.com sebagai aplikasi pencari jodoh asal Indonesia, masih menggunakan arsitektur monolitik. Heartenly.com, membutuhkan arsitektur yang dapat melakukan peningkatan atau perbaikan sistem dapat dilakukan secara independen tanpa harus mematikan seluruh layanan. Untuk itu Heartenly.com perlu melakukan perubahan pada arsitekturnya, dari monolitik menjadi *microservice*. Proses perubahan tersebut melibatkan proses dekomposisi untuk memecah layanan-layanan yang ada pada Heartenly.com. Metode dekomposisi menggunakan prinsip *Domain-driven Design* (DDD) dipadukan dengan arsitektur *Representational State Transfer* (REST) untuk mengoptimalkan jalur komunikasi karena Heartenly.com berbasis *web*. Hasil dekomposisi diuji dengan metode *Single Responsibility Principle* (SRP) dan *Common Closure Principle* (CCP) adaptasi dari prinsip *Object Oriented Programming* (OOP). Dalam penelitian ini, hasil analisis membuktikan bahwa, *microservice* dapat melakukan peningkatan layanan dan *deployment* secara independen

Kata kunci : *Microservice*, dekomposisi, domain-driven design, REST

Abstract

Monolithic architecture designing application systems consists of only one large part and runs on one computational instance [1]. This causes, the process of improving or repairing the application system involving the whole system. Heartenly.com as a matchmaking application from Indonesia, still uses monolithic architecture. Heartenly.com, requires an architecture that can improve or repair the system independently without having to turn off all services. For this reason Heartenly.com changes its architecture to Microservice. The process of change involves a decomposition process to break down the services that are on Heartenly.com. The decomposition method uses the principle of Domain-driven Design (DDD) combined with the Representational State Transfer (REST) architecture to optimize communication lines because Heartenly.com is web-based. The results of decomposition are tested by the method of the Single Responsibility Principle (SRP) and the Common Closure Principle (SRP) adaptation of the principle of Object Oriented Programming (OOP). In this research, the results of the analysis prove that microservice can do improvement and deployment independently

Keywords: *software architecture, microservice, domain-driven design, REST*

1. Pendahuluan

Arsitektur monolitik mendesain perangkat lunak hanya terdiri dari satu bagian besar dan berjalan pada satu *computational instance* [1]. Meskipun begitu, arsitektur monolitik memungkinkan untuk menjalankan lebih dari satu proses pada satu CPU namun semuanya berbagi satu sistem operasi dan perangkat keras yang sama. Hal ini dapat menyebabkan diantaranya kurangnya fleksibilitas dan skalabilitas [1]. Ketika akan melakukan peningkatan atau memperbaiki kerusakan suatu layanan, dalam arsitektur monolitik harus mematikan seluruh layanannya. Tidak hanya itu, *update* sekecil apapun sistem harus di *re-compile* dan *deploy* secara keseluruhan, agar *update* bisa sampai kepengguna. Permasalahan tersebut yang dicoba untuk diatasi oleh Heartenly.com.

Heartenly.com merupakan salah satu aplikasi pencari jodoh asal Indonesia, yang saat ini mempunyai jumlah pengguna lebih dari 300.000 dan masih terus bertambah. Heartenly.com berencana untuk menambahkah layanan-layanan baru dan memperbaiki yang sudah ada agar lebih memanjakan penggunanya. Namun saat ini Heartenly.com masih menggunakan arsitektur monolitik. Seperti yang sudah dijelaskan sebelumnya, dalam monolitik harus mematikan seluruh layanan ketika akan menambah atau memperbaiki suatu layanan tertentu. Untuk itu, sebelum melakukan penambahan atau peningkatan layanan, Heartenly.com harus mempersiapkan arsitekturnya terlebih dahulu agar mudah untuk melakukan peningkatan atau perbaikan layanan. Arsitektur tersebut adalah arsitektur *Microservice* [1].

Arsitektur *microservice* mendekomposisi satu entitas besar atau layanan besar menjadi layanan – layanan yang kecil sehingga layanan tersebut independen namun secara *autonomous* masih saling bekerja sama [2]. Tidak ada ukuran yang pasti dalam menentukan seberapa kecil suatu layanan didekomposisi, tetapi faktor penetapan ukuran dapat dilihat dari keselarasan tim untuk menangani kompleksitas sistem dari hasil dekomposisi layanan [2]. Hasil dekomposisi layanan tersebut kemudian dihubungkan menggunakan arsitektur REST (Representational State Transfer), sebuah gaya pendekatan dalam membangun arsitektur perangkat lunak yang memisahkan antara *Client* dengan Server atau sebagai medium antara *Client* dengan *Service* [3]. Client dan Server berkomunikasi mengikuti standar protokol *Hypertext Transfer Protocol* (HTTP) [4]. REST dipilih selain karena Heartenly.com sudah menggunakannya, tetapi juga karena penyedia *Web* saat ini dianggap dapat menyediakan skalabilitas, keamanan dan *reliability* [4]. HTTP dapat diimplementasikan untuk berbagai macam kebutuhan seperti *transfer file*, *streaming*, dan *shared database* [5]. Kombinasi keunggulan *microservice* dan REST ini sejalan dengan kebutuhan Heartenly.com, yaitu dapat menambah dan/atau memperbaiki layanannya, secara independen, maka Heartenly.com perlu merubah arsitekturnya dari monolitik ke *microservice*.

Berdasarkan permasalahan di atas dibutuhkan analisis dan desain arsitektur untuk mendekomposisi layanan-layanan Heartenly.com, sehingga dapat menangani permasalahan di atas. Dekomposisi yang dilakukan menggunakan pendekatan pendekatan *Domain-Driven Design* (DDD). DDD merupakan suatu metode untuk memodelkan kegiatan domain organisasi yang ada di dunia nyata [2]. Dalam metode DDD pengembang dan analis duduk bersama untuk memahami konsep domain, *policies*, dan logika [6]. Analisis desain ini dapat digunakan sebagai referensi untuk pengembangan Heartenly.com selanjutnya.

Rumusan Masalah

Berdasarkan pendahuluan yang telah di jelaskan sebelumnya, maka didapat beberapa rumusan masalah, yaitu :

1. Bagaimana mendesain arsitektur *Microservices* dengan REST pada Heartenly.com?
2. Bagaimana perbedaan kemudahan *upgrade* antara arsitektur Monolitik dengan *Microservice*?

Tujuan

Tujuan dari penelitian tugas akhir ini yaitu

1. Mendesain arsitektur *Microservice* dengan REST pada Heartenly.com
2. Membandingkan kemudahan *upgrade* arsitektur Monolitik dan *Microservice*

2. Tinjauan Pustaka

2.1. Arsitektur Perangkat Lunak

Menurut ISO/IEC Standard 42010, arsitektur perangkat lunak adalah hal fundamental dari suatu sistem di lingkungannya yang terkandung didalamnya elemen-elemen, hubungan antar elemen, dan dalam prinsip-prinsip desain dan evolusi [7]. Dalam arsitektur perangkat lunak terdapat beberapa sudut pandang, agar arsitektur dapat difahami oleh para pemangku kepentingan yaitu:

- a. Sudut Pandang Struktur
Menggambarkan blok komponen, konektor, dan konfigurasi pada sistem dengan diagram .
- b. Sudut Pandang Tingkah Laku
Menjelaskan spesifikasi protokol komunikasi antar komponen melalui konektor dengan perhatian/pandangan tingkah laku pada arsitektur

c. Sudut Pandang Eksekusi

Menjelaskan representasi arsitektur ke dalam pandangan kode (*code view*)

2.2. Desain Perangkat Lunak

Desain perangkat lunak adalah proses dimana, suatu agen membuat spesifikasi artefak perangkat lunak, yang dimaksudkan untuk mencapai suatu tujuan, menggunakan serangkaian komponen primitif dan mengikuti suatu aturan [8]. Desain perangkat lunak dapat merujuk pada "semua aktivitas yang terlibat dalam konseptualisasi, pembingkai, implementasi, *commissioning* (proses verifikasi fungsionalitas apakah sesuai tujuan atau spesifikasi yang sudah ditentukan sebelumnya), dan memodifikasi sistem kompleks" atau "aktivitas yang mengikuti spesifikasi sebelum pemrograman, seperti proses rekayasa perangkat lunak bergaya." [9].

2.3. Monolitik

Arsitektur monolitik merupakan pendekatan alami untuk membangun sebuah perangkat lunak. Monolitik mengembangkan dan menyebarkan perangkat lunak sebagai satu entitas. Perangkat lunak berarsitektur monolitik merupakan pendekatan alami untuk membangun sebuah perangkat lunak [8]. Namun memiliki beberapa masalah seperti di bawah ini [10]:

a. Adaptasi Perubahan

Perubahan pada satu modul dari arsitektur monolitik membutuhkan proses *rebooting* perangkat lunak secara menyeluruh. Untuk proyek yang besar, proses *rebooting* biasanya mempertimbangkan *downtimes* dan memberhentikan proses pengembangan, pengujian dan pengelolaan proyek.

b. Deployment

Menyebarkan perangkat lunak monolitik biasanya kurang optimal karena persyaratan yang saling bertentangan mengenai sumber daya. Ketika menentukan lingkungan untuk penyebaran, pengembang harus berkompromi dengan konfigurasi *one-size-fit-all* yang bisa suboptimal atau mahal tergantung dengan modul individu.

c. Skalabilitas

Monolitik membatasi skalabilitas pada perangkat lunak. Cara umum untuk menangani kenaikan permintaan yaitu dengan membuat *instance* baru dengan perangkat lunak yang sama untuk membagi beban. Tetapi bisa saja kenaikan lalu lintas hanya bagian dari modul. Yang membuat alokasi sumber daya baru untuk komponen lain menjadi sulit.

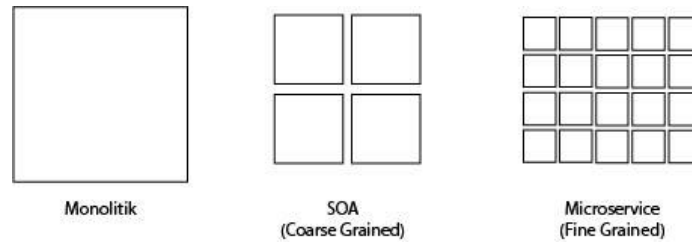
2.4. Service Oriented Architecture

Service Oriented Architecture (SOA) dikembangkan untuk menutupi kelemahan-kelemahan dari arsitektur monolitik. SOA merupakan sebuah pendekatan untuk mendesain bagaimana beberapa layanan berkolaborasi untuk menyediakan suatu kumpulan kapabilitas. Layanan yang dimaksud yaitu layanan yang benar-benar independen tidak memakai sistem operasi yang sama namun tetap berjalan. Komunikasi antar layanan terjadi melalui suatu jaringan tidak di dalam batas proses. SOA mempunyai tujuan untuk meningkatkan reusability suatu perangkat lunak. Sebagai contoh dua atau lebih suatu aplikasi dapat menggunakan satu layanan yang sama. SOA juga bertujuan membuat perangkat lunak mudah untuk dikelola atau ditulis ulang, bahkan secara teori dapat mengganti suatu layanan tanpa diketahui orang lain selama layanan semantic tidak berubah terlalu banyak [2].

2.5. Microservices

Standar definisi dari *microservice* sampai saat ini belum ada, tetapi dapat dijelaskan sebagai teknik untuk membangun atau mengembangkan perangkat lunak yang mensyaratkan suatu aplikasi tunggal dapat disebar (deploy) secara independen [8].

SOA dan *microservice* menghasilkan banyak perdebatan antar pengembang perangkat lunak ada yang menerangkan bahwa *microservice* adalah versi terbaru dari SOA, ada yang meyakini bahwa *microservice* merupakan suatu konsep yang berbeda dengan SOA. Perbedaan utama dari SOA dan *microservice* ialah pada ukuran dan cakupan suatu layanan. Bersandar pada istilah "*micro*" pada *microservice* yang berarti kecil jauh lebih kecil dengan SOA yang berorientasi pada layanan seperti dapat di lihat pada Gambar 2.3 [8]. *Microservice* berorientasi pada "*Small and focused on doing one thing well*" dan berjalan otonom [9].



Gambar 2. 3 Monolitik vs SOA vs Microservice

2.6. Representational State Transfer (REST)

Representational State Transfer (REST) adalah gaya arsitektur untuk mendistribusikan *Hypermedia*. *Hypermedia* sendiri adalah istilah yang digunakan untuk Hypertext yang tidak dibatasi oleh teks, dapat berupa gambar, suara dan video [11]. Layanan web yang sesuai dengan gaya arsitektur REST, atau layanan web RESTful, menyediakan interoperabilitas antara sistem komputer di Internet. Layanan web REST-compliant memungkinkan sistem yang meminta untuk mengakses dan memanipulasi representasi tekstual sumber daya web dengan menggunakan serangkaian *stateless operations* yang seragam dan telah ditentukan sebelumnya [12].

2.7. Domain Driven-Design (DDD)

Domain Driven Design (DDD) adalah koleksi prinsip-prinsip dan pola yang membantu pengembang menghasilkan sistem yang elegan mudah difahami. DDD memungkinkan membuat kode dari hasil komunikasi dengan stakeholder, tanpa dokumentasi yang terlalu banyak [7].

DDD menggabungkan penerapan desain dan *development practice*, dan menunjukkan bagaimana desain dan proses *development* dapat bekerja sama untuk menciptakan solusi yang lebih baik. Desain yang baik akan mengakselerasi pengembangan, sementara umpan balik dari proses *development* akan menyempurnakan desain [12]

2.7.1. Dekomposisi Layanan

Untuk beralih dari arsitektur monolitik ke arsitektur *microservice*, layanan-layanan yang tadinya menjadi satu entitas (arsitektur monolitik) dipecah menjadi layanan kecil yang independen (*microservice*). Terdapat beberapa definisi tentang bagaimana memecah layanan monolitik menjadi *microservice* diantaranya:

a. *Bounded Context*

Merujuk pada konsep Domain-driven design Eric Evans, singkatnya layanan yang sudah ada dianalisis kemudian ditentukan batas-batas konteks sesuai tanggung jawab layanan, lalu irisan dari batas yang sudah ditentukan menjadi satu layanan utuh yang independen. Definisi lainnya ialah tanggung jawab khusus yang dibuat oleh batas-batas konteks secara eksplisit misal jika membutuhkan suatu informasi dari *bounded-context* atau membuat suatu fungsionalitas dalam *bounded-context* maka harus berkomunikasi dengan *bounded-context*-nya secara eksplisit dengan model [2].

b. *Business Capabilities*

Business Capabilities adalah suatu kegiatan yang dilakukan oleh bisnis untuk menghasilkan nilai. Kemampuan bisnis tergantung pada jenis bisnis yang dilakukan, misalnya kemampuan bisnis untuk perusahaan asuransi biasanya penjaminan, manajemen klaim, *billing* dan yang lainnya. Dekomposisi layanan menggunakan cara ini merupakan cara yang cukup stabil, karena dibuat berdasarkan bagaimana organisasi melakukan bisnisnya [13].

Berdasar pada teori dari Sam Newman [2], Chris Richardson [13] menjelaskan lebih rinci dalam menggambarkan bagaimana membuat domain dari suatu studi kasus sehingga penulis membuat langkah-langkah atau metode untuk mendekomposisi layanan dengan prinsip DDD yaitu:

a. Penentuan Domain

Setelah memahami arsitektur *microservice*, dan DDD kemudian melakukan diskusi dengan pengembang untuk menentukan domain apa saja yang ada pada sistem Heartenly.com. Tahap ini berfungsi untuk mendapatkan dasar pemahaman yang sama dengan pengembang, sehingga hasil domain yang sudah ditentukan sesuai dengan kebutuhan.

b. Menentukan Subdomain Layanan

Setelah memiliki cukup data kemudian melakukan dekomposisi layanan, dari domain yang sudah ditentukan sebelumnya, menjadi lebih kecil. Ukuran kecil disini tergantung pada jenis bisnis serta kesiapan tim pengembang. Lalu hasil dekomposisi tersebut disebut sebagai subdomain. Subdomain ini yang nantinya akan menjadi *microservice*. Pada tahap ini, dalam menentukan subdomain memakai prinsip *microservice* “*Small, and Focus on Doing One Thing Well*” [2]. Pada tahap ini juga dilakukan metode *Bounded-Context* untuk memperjelas tanggung jawab masing-masing subdomain.

c. Menentukan Fungsionalitas dari Subdomain

Untuk mempermudah seluruh pihak yang terlibat dalam pembuatan desain *microservice*, setelah mendapatkan domain harus disertakan semua fungsionalitas-fungsionalitas dari domain. Langkah “b” sampai “c” dilakukan berulang-ulang dengan pihak pengembang atau pihak terkait agar hasil desain memenuhi ketentuan yang pengembang inginkan.

2.8. Metode Evaluasi

Keluaran dari penelitian ini berupa sebuah desain arsitektur oleh karena itu dibutuhkan suatu kriteria untuk menentukan suatu desain itu baik atau buruk. Kriteria yang dipakai dalam penelitian ini mengadaptasi dua prinsip *Object Oriented design* [13] yaitu:

1. Single Responsibility Principle (SRP)

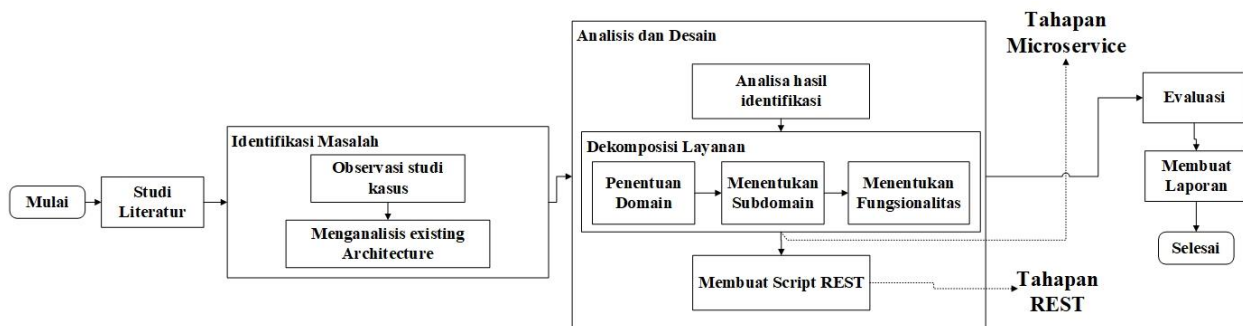
Setiap tanggung jawab yang dimiliki kelas, mempunyai potensi berubah. Jika suatu kelas memiliki banyak tanggung jawab yang berubah secara independen, kelas tidak akan stabil. Dengan mengikuti SRP, masing-masing kelas memiliki satu tanggung jawab sehingga hanya memiliki satu alasan untuk melakukan perubahan. SRP dapat mendefinisikan arsitektur *microservice* dan membuat layanan kecil, kohesif yang masing-masing memiliki tanggung jawab tunggal. Ini akan mengurangi ukuran layanan dan meningkatkan stabilitas mereka.

2. Common Closure Principle (CCP)

Idenya adalah bahwa jika dua kelas berubah karena alasan mendasar yang sama, maka mereka termasuk dalam paket yang sama, misalnya, kelas-kelas tersebut menerapkan aspek berbeda dari aturan bisnis tertentu. Tujuannya adalah ketika aturan bisnis itu berubah, pengembang hanya perlu mengubah *package* kode dalam jumlah kecil (idealnya hanya satu). Mematuhi CCP secara signifikan dapat meningkatkan pemeliharaan aplikasi. Cara yang sama dapat diterapkan dalam membuat *microservice*. Melakukan hal tersebut akan meminimalkan jumlah layanan yang perlu diubah ketika terjadi perubahan. Idealnya, perubahan hanya akan memengaruhi satu tim dan satu layanan. CCP adalah *anti-pattern* dari arsitektur monolitik.

3. Metodologi Penelitian

Dalam melakukan penelitian dan pembangunan arsitektur *microservice* dengan REST pada layanan Heartenly.com di lakukan berdasarkan metode penelitian seperti yang terlihat pada gambar dibawah ini.



Gambar 3. 1 Skema Metodologi Penelitian

3.1. Studi Literatur

Dalam tahap ini penulis melakukan studi literatur baik itu dari perpustakaan Universitas Telkom maupun *online*. Serta menganalisis dokumen-dokumen yang perlu dipahami dari studi kasus yaitu Heartenly.com

3.2. Identifikasi Masalah

Proses menemukan, mengumpulkan, meneliti, mendaftarkan, mencatat data dan informasi dari studi kasus sebagai langkah awal dari penelitian ini. Dalam identifikasi masalah pada penelitian ini melibatkan proses lain yaitu :

3.2.1. Observasi Studi Kasus

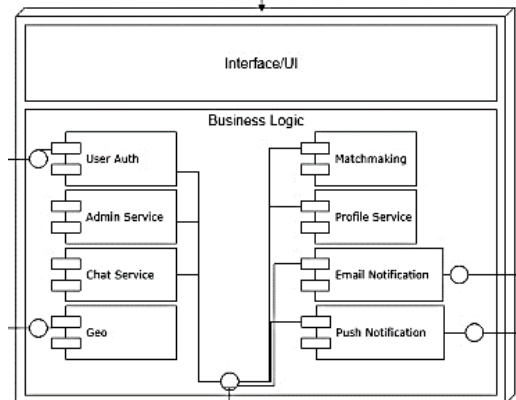
Melakukan observasi langsung dengan berdiskusi bersama pihak pengembang dan tim Heartenly.com untuk mendapatkan data-data yang diperlukan. Serta secara tidak langsung dengan mengobservasi website Heartenly.com

3.2.2. Analisis Arsitektur yang Ada

Merupakan tahap lanjutan dari observasi yang berfokus pada analisis arsitektur yang ada pada Heartenly.com yang masih menggunakan arsitektur monolitik.

3.3. Analisis dan Desain

Dalam tahap ini dilakukan proses identifikasi dari studi kasus kemudian di lakukan proses dekomposisi layanan Heartenly.com berdasarkan Identifikasi Masalah dan data-data yang terkumpul. Proses dilakukan dalam tiga (3) tahap yaitu:



Gambar 3. 2 Arsitektur Monolitik Heartenly.com

a. Identifikasi Studi Kasus (Heartenly.com)

Melakukan identifikasi pada arsitektur monolitik Heartenly.com dengan observasi langsung terhadap halamannya disertai dengan wawancara kepada pengembang. Sehingga didapatkan kerangka layanan yang ada pada Heartenly.com yang digambarkan pada Gambar 3.2. Arsitektur Monolitik Heartenly. Kotak kecil yang ada didalam kotak “Business Logic” merupakan rangkuman fungsionalitas yang ada pada Heartenly.com

b. Tahap Dekomposisi Layanan

Pada tahap ini dilakukan proses pemecahan layanan-layanan dari studi kasus mengikuti langkah-langkah yang ada pada bagian **2.4.1. Dekomposisi Layanan**

1. Penentuan Domain

Tahap ini melakukan diskusi dengan pengembang, menggunakan data dari hasil identifikasi studi kasus seperti pada Gambar 4.1. Pada gambar tersebut sudah cukup menunjukan domain dari Heartenly.com, namun tidak semuanya termasuk domain, karena ada beberapa fungsionalitas bersumber dari pihak ketiga. Secara singkat domain utama dari Heartenly.com yaitu “Sosial media dan *Matchmaking* (Penjodohan)”. Domain tersebut didekomposisi menjadi lebih kecil sehingga didapatkan domain baru seperti yang terlihat pada Gambar 3.3 dibawah ini



Gambar 3. 3 Domain Utama

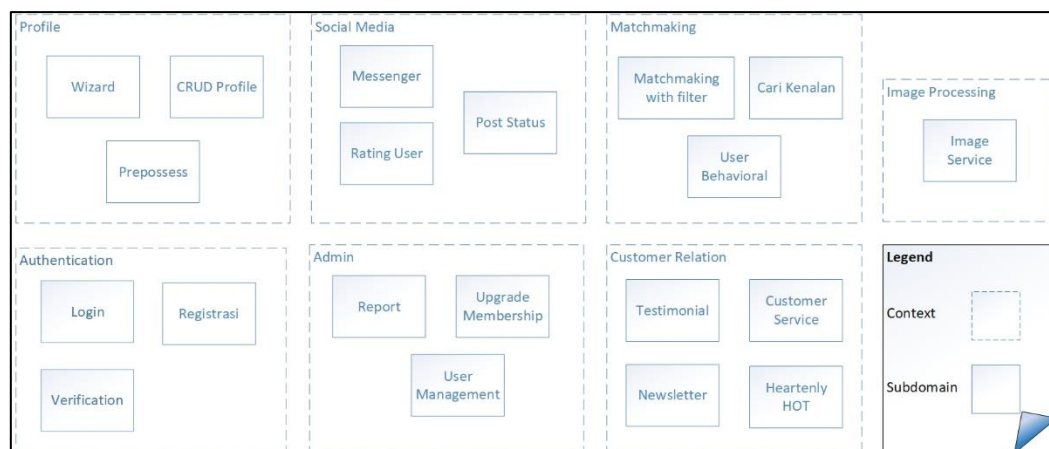
Setelah mendapatkan kapabilitas bisnis dari Heartenly.com yang saat ini sedang berjalan, kemudian melakukan diskusi dengan Pengembang tentang rencana pengembangannya. Lalu Pengembang memberikan gambaran serupa tentang domainnya, namun dengan jumlah yang lebih banyak seperti pada gambar dibawah ini.



Gambar 3. 4 Rencana Pengembangan Layanan Heartenly.com

2. Menentukan Subdomain

Data yang sudah terkumpul sebelumnya termasuk rencana pengembangan, kemudian dianalisis kapabilitas bisnisnya, sehingga layanan menjadi lebih kecil. Tolak ukur kesesuaian layanan kecil yang direkomendasikan, yaitu berdasarkan persetujuan dari Pengembang, karena ukuran kecil layanan tidak ada aturan bakunya. Ukuran layanan hanya disesuaikan dengan kesiapan tim pengembang [2], sehingga tahapan ini melibatkan Pengembang untuk memvalidasi rancangan layanan.



Gambar 3. 5 Rancangan *Bounded Context*

Untuk mendapatkan subdomain dari setiap kapabilitas bisnis yang sudah didekomposisi, caranya dengan melakukan pendekatan *Bounded-Context*. *Bounded-context* ini merupakan metode yang ada pada DDD untuk membatasi tanggung jawab suatu layanan [2]. Setelah rancangan layanan disetujui kemudian dipetakan dalam bentuk diagram agar memudahkan dalam pembuatan *bounded-context*, maka didapatkan rancangan seperti pada Gambar 3.5 Rancangan *Bounded Context*

3. Menentukan Fungsionalitas

Setelah mendapatkan *bounded-context* serta rancangan *microservice* kemudian melakukan pendefinisian fungsionalitas pada setiap rancangan *microservice* guna memudahkan menyamakan persepsi atas rancangan yang dibuat sehingga didapat keluaran seperti di bawah ini:

Tabel 3. 1 Tabel Penentuan Fungsionalitas Berdasarkan subdomain yang dibuat

No	Domain Context	Subdomain	Functionality
1	Profile	CRUD Profile	Manage Personal Data of user
		Wizard	Manage Job Detail
		Authentication	Manage Education Detail
		Public	Manage Public Profile
			Manage Criteria of Person to be sought
2	Authentication	Authentication	Login for registered user
		Registration	Register new user
			Forgot Password
			Forgot Email
		Verification	Phone Verification
			Email Verification
			NIK Verification
			Job Verification
			Salary Verification
3	Social Media	Posts	Post Status
			Like Status
			Comment Status
		Chat	Manage Inbox
			Realtime Chat
		Rating	Store rating of users
			Processing users rating
4	Matchmaking	Find User	Search specific user
		Matchmaking	find user base on gender type
			find user base on religion
			find user base on culture
			find user base on region
			Suggest other users to "user"
		User Behavioral	Record visitor (user and non user) click activity
5	Customer Relation	Customer Service	Manage FAQ
			Handling User Complaint
			Store User Suggestion

No	Domain Context	Subdomain	Functionality
		Testimonial	Handling user testimonial
		Newsletter	Create News
			Broadcast News
		Blog Service (HOT)	Manage Articles blog
6	Admin	User Management	Manage user
			Determine Featured User
		Membership	Membership registration
			Payment Verification
		Report	Warn Problematic user
7	Image Processing	Image	Reduce Size Uploaded Image
			Resize Dimension Image
8	Notification	Notification	Email Notification
			Phone Notification
			Push Notifications

c. Tahap Script REST

Tahapan ini membuat script REST dari Heartenly.com dengan Swagger, sebagai rekomendasi rancangan arsitektur REST untuk *microservice* yang sudah dibuat. Swagger adalah sebuah aplikasi untuk membuat dokumentasi REST menggunakan script YAML (*YAML Ain't Markup Language*), sebuah bahasa serialisasi data yang dapat dibaca manusia yang biasa digunakan untuk mengkonfigurasi file [14]

3.4. Evaluasi

Evaluasi desain dilakukan dengan metode evaluasi *Single Responsibility Principle* (SRP) dan *Common Closure Principle* (CCP) untuk dekomposisi layanan dari monolitik ke *microservice*. Untuk menguji bagaimana perbandingan proses *upgrade* layanan pada monolitik, akan menggunakan skenario pengujian. Proses evaluasi ini melibatkan pihak pengembang karena membutuhkan keselarasan dalam timnya dan validasi nama *microservice*.

3.1.1. Evaluasi SRP dan CCP

Evaluasi dilakukan dengan memeriksa tanggung jawab dari *microservice* yang sudah dirancang. Caranya dengan membuat daftar tanggung jawab yang dimiliki oleh domain, lalu dibuat *microservice* dari tanggung jawab tersebut. Kemudian memeriksa data atau tanggung jawab yang diolah oleh setiap *microservice*, apakah mempunyai kesamaan atau tidak dengan *microservice* lain. Ketika memiliki kesamaan maka dilebur atau disatukan.

Tabel 3. 2 Daftar Tanggung Jawab Subdomain

No	Domain Context	Subdomain	Responsibility
1	Profile	CRUD Profile	Manage Personal Information
		Wizard	Manage Detail Information of User
		Public	Manage Public Profile
		Criteria Management	Manage Criteria of Person to be sought
2	Authentication	Login	Manage Authentication
		Registration	Manage Registration
		Verification Service	Manage all Verifiaction Process
3	Social Media	Post	Manage Status
		Chat	Manage Chat of users
		Rating	Manage rating of users

No	Domain Context	Subdomain	Responsibility
4	Matchmaking	Matchmaking	Manage Matching user between user
		Cari Kenalan	Manage user search
		User Behavioral	Manage Recording visitor (user and non user) click activity
5	Customer Relation	Customer	Manage User feedback
		Testimonial	Manage user testimonial
		Newsletter	Manage News
		Blog Service (HOT)	Manage Articles blog
6	Admin	User Management	Manage user
		Membership Serice	Manage Membership registration
		Report	Manage Report
7	Image Processing	Image	Reduce Size Uploaded Image
8	Notification	Notification	Email Notification

3.1.2. Skenario Evaluasi Upgrade Layanan

Skenario yang digunakan yaitu, melakukan *upgrade* pada tiga *package*(monolitik)/*microservice* yang mempunyai dependensi ke salah satu *package/microservice*. Disebut *package* karna monolitik pada Heartenly.com menggunakan prinsip OOP (*Object Oriented Programming*) pada *source code*-nya. Untuk melakukan perluasan sistem, dalam monolitik sekecil apapun perluasan yang dilakukan sistem secara keseluruhan harus dimatikan kemudian di *deploy* ulang secara keseluruhan sistem juga. Dalam *microservice* ketika perluasan sistem atau sekedar *upgrade* layanan dapat dilakukan secara independen per-*microservice*, tanpa harus men-*deploy* secara keseluruhan.

3.5. Membuat Laporan

Membuat laporan berdasarkan penelitian yang sudah dilakukan terhadap Heartenly.com sehingga mendapatkan suatu kesimpulan dari penilitian itu.

4. Evaluasi dan Analisis

4.1. Analisis Hasil Evaluasi

4.1.1. Hasil Dekomposisi

Merujuk pada hasil *bounded context* seperti yang terlihat pada Gambar 3.5 maka dilakukan evaluasi, sehingga mendapatkan beberapa subdomain yang dilebur, karena memiliki tanggung jawab yang sama. Subdomain terletak pada konteks *Profile*, *Authentication*, dan *Matchmaking*



Gambar 4. 1. Peleburan dari *Bounded-Context*

a. Wizard dan *CRUD Profile*

Wizard yang dimiliki Heartenly merupakan layanan untuk melengkapi detail profil pengguna, seperti pekerjaan, pendidikan dan yang lainnya. Wizard merupakan layanan yang sudah berjalan sebelumnya bukan hasil dekomposisi. *CRUD Profile* merupakan hasil identifikasi penulis yang fungsinya mengolah profil pengguna. Pada dasarnya kedua

subdomain tersebut, memiliki tanggung jawab yang sama, yaitu “Mengolah Profil”. Sehingga subdomain tersebut dapat disatukan dan diberi nama yang lebih menjelaskan fungsinya yaitu “Bio” kependekan dari Biodata atau Biography.

b. Login dan Registrasi

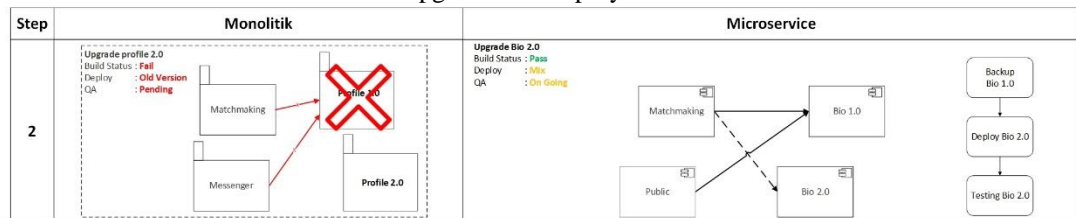
Fungsi inti dari konteks *Authentication*, yaitu menyaring pengunjung yang masuk ke dalam aplikasi sebagai pengguna atau member. Pengunjung tidak dapat mengakses seluruh layanan yang tersedia, pengunjung harus mendaftar terlebih dahulu agar dapat melalui proses penyaringan. Kemudian dapat disimpulkan bahwa Login dan Registrasi memiliki tanggung jawab yang sama yaitu, menyaring pengunjung atau autentikasi. Lalu kedua subdomain tersebut dilebur dan nama subdomainnya menjadi “*Authentication*”.

c. *Matchmaking with Filter* dan Cari Kenalan

Matchmaking dan Cari Kenalan mempunyai fungsi dasar yang sama, yaitu pencarian. Keduanya hanya dibedakan oleh parameter yang digunakan. Oleh karena itu, kedua subdomain tersebut dilebur, sebab tanggung jawabnya tetap pada melakukan pencarian. Kemudian subdomain tersebut dinamakan “*Matchmaking*”

4.1.2. Hasil Evaluasi Skenario

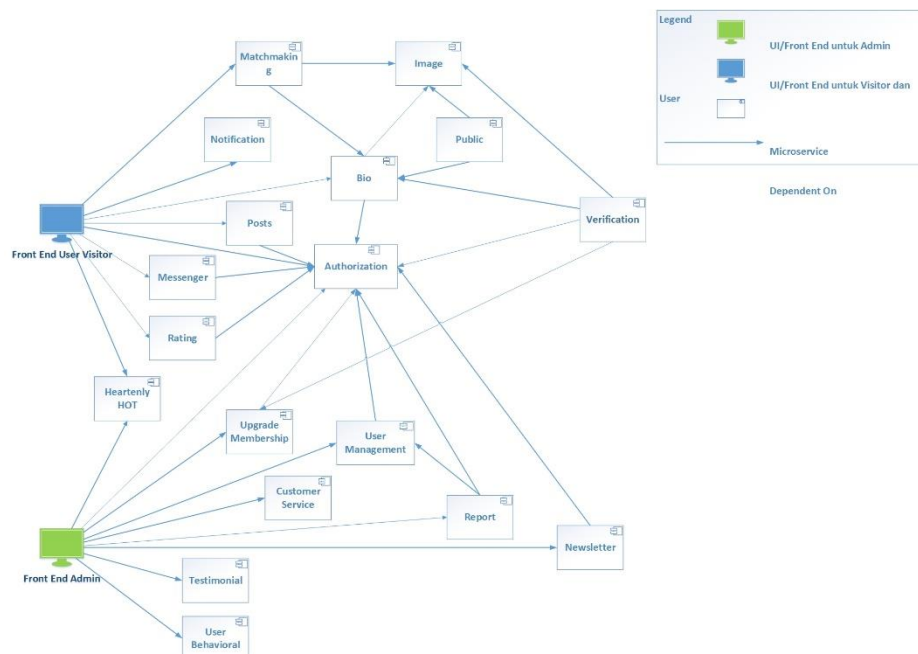
Gambar 4. 2 Proses Upgrade dan Deployment Monolitik dan *Microservice*



Pada Gambar 4.2, pada bagian monolitik terdapat *package Matchmaking, Messenger, Profile 1.0* dan *Profile 2.0*. Gambar tersebut menggambarkan, ketika *Matchmaking* dan *Messenger* akan menggunakan *Profile 2.0* yang merupakan versi terbaru dari *Profile 1.0*, aplikasi tidak dapat di-deploy, seluruh sistem harus dimatikan, dan dipastikan terlebih dahulu ter-compile untuk kemudian maju ke tahap QA, deploy dan testing. Berbeda dengan *microservice* tahap QA, deploy dan testing dapat dilakukan secara independen, pada *microservice* tertentu tanpa mempengaruhi sistem yang lain. Bukti proses upgrade pada arsitektur *microservice* terdapat pada **Lampiran 8**.

4.1.3. Rancangan Desain Akhir Hasil Dekomposisi

Setelah melakukan dekomposisi, kemudian dibuat desain dari keseluruhan *microservice*-nya, sehingga mendapatkan rancangan desain seperti gambar dibawah ini:



Gambar 4. 3 Desain Hasil Dekomposisi

Mengacu pada penjelasan di bagian **2.2 Desain Perangkat Lunak**, Gambar 4.3 merupakan desain perangkat lunak, karena subdomain atau *microservice* pada gambar tersebut merepresentasikan artefak kode yang mempunyai tujuannya masing-masing. Pada gambar tersebut juga mengkonseptualisasikan hubungan antar komponen. Mengacu pada penjelasan di bagian **2.2 Arsitektur Perangkat Lunak**, Gambar 4.3 juga dapat dianggap sebagai arsitektur pada sudut pandang tingkah laku. Sudut pandang tersebut menjelaskan spesifikasi dan komunikasi antar komponen atau *microservice*

5. Kesimpulan

5.1. Kesimpulan

Tujuan dari penelitian ini adalah melakukan desain konsep arsitektur *microservice* dengan REST pada sistem informasi Heartenly.com, yang didekomposisi dengan pendekatan DDD. Hasil evaluasi menggunakan metode SRP menunjukkan, desain konsep dapat memenuhi tanggung jawab layanan yang ada pada Heartenly.com dengan ukuran yang lebih kecil. Desain *microservice* yang sudah divalidasi oleh pengembang Heartenly.com dapat dijadikan rujukan atau referensi jika Heartenly.com akan memakai arsitektur *microservice*. *Microservice* dapat menjadi alternative solusi untuk sistem informasi Heartenly.com, yang dapat meningkatkan dan mendeploy sistem, secara independen tanpa mempengaruhi sistem yang tidak sedang ditingkatkan. Berbanding terbalik dengan arsitektur monolitik yang mengharuskan *upgrade* dan *deploy* secara menyeluruh, tidak bisa secara spesifik *upgrade* dan *deploy* pada sistem tertentu.

5.2. Saran

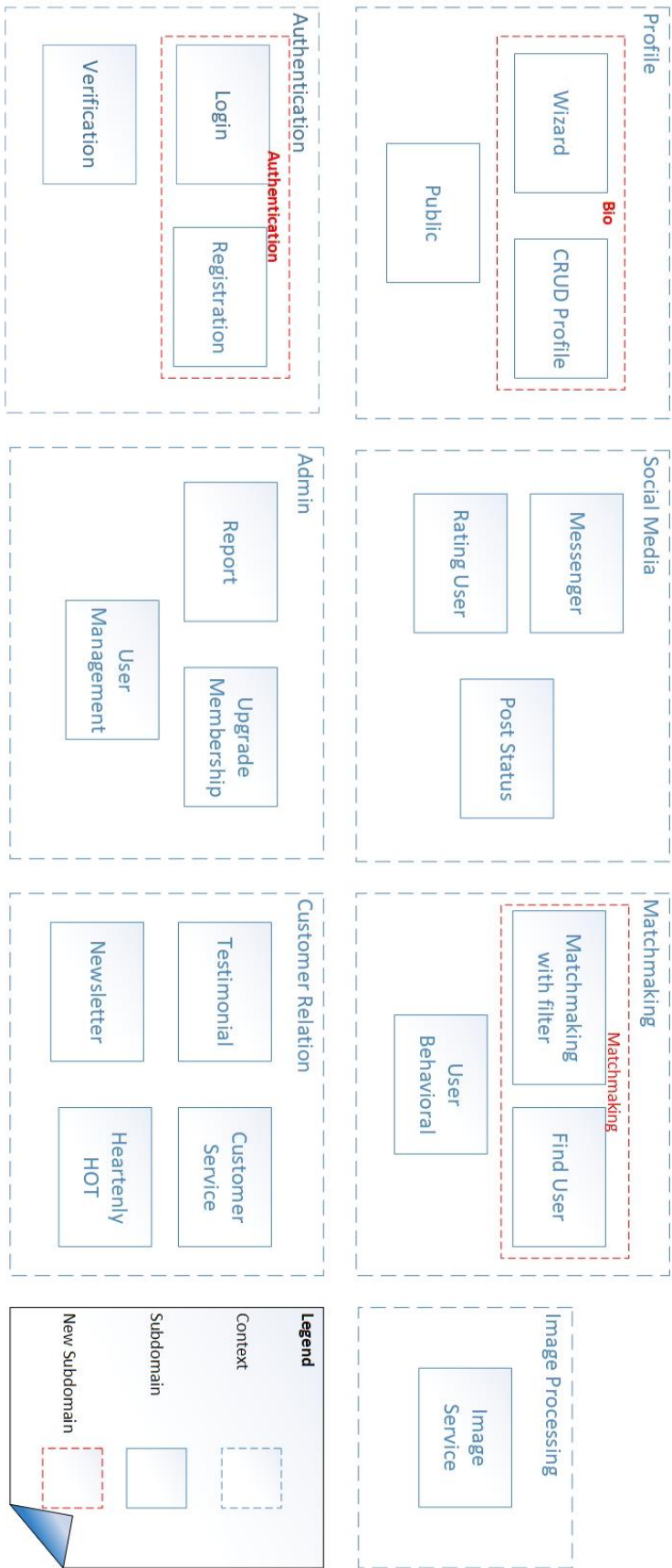
Analisis pada sistem Heartenly.com disarankan menggunakan lebih banyak sudut pandang yang menjadikan *microservice* unggul dari monolitik salah satunya *culture of automation*. Penelitian arsitektur *microservice* dapat mencapai tahap implementasi, agar bisa lebih menggambarkan poin plus *microservice*.

6. Daftar Pustaka

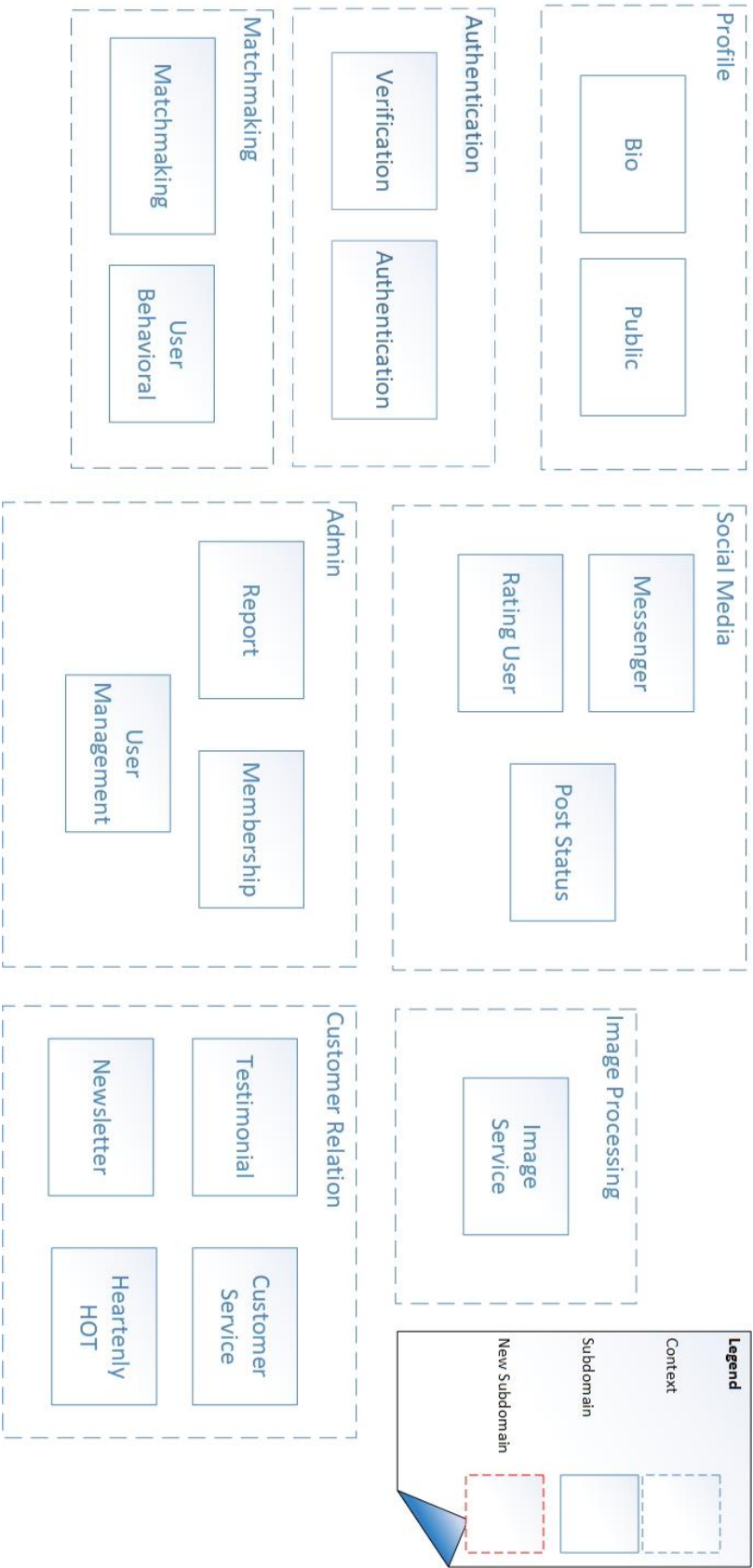
- [1] D. S. D. B. C. H. P. E. T. B. Benjamin Götz, "Challenges of production microservices," dalam *11th CIRP Conference on Intelligent Computation in Manufacturing Engineering - CIRP ICME '17*, Stuttgart, 2018.
- [2] S. Newman, *Building Microservice*, Sebastopol: O'Reilly Media, Inc., 2015.
- [3] B. C. C. P. R. B. Thomas Erl, *SOA with REST : Principles, Patterns & Constraints for Building Enterprise Solutions with REST*, Munich: Prentice Hall, 2013.
- [4] S. P. I. R. Jim Webber, *REST in Practice*, California: O'Reilly Media, 2010.
- [5] O. Z. Cesare Pautasso, "The Web as a Software Connector," *Integration Resting on Linked Resources*, Vol. %1 dari %20740-7459, no. 18, pp. 93-98, 2018.
- [6] N. T. Scott Millett, *Patterns, Principles and Practices of Domain-Driven Design*, Indianapolis: John Wiley & Sons, 2015.
- [7] J. L. T. B. Flavio Oquendo, *Software Architecture in Action*, Switzerland: Springer International, 2016.
- [8] Y. W. Paul Ralph, "A Proposal for a Formal Definition of the Design Concept," dalam *Lecture Notes in Business Information Processing*, Berlin, Springer, Berlin, Heidelberg, 2009, pp. 103-136.
- [9] D. H. Peter Freeman, "A science of design for software-intensive systems," *Communications of the ACM - Interactive immersion in 3D graphics*, vol. XLVII, no. 8, pp. 19-21, 2004.
- [10] S. Fleming, *Continuous Delivery Handbook Non Programmer's Guide To DevOps, Microservices and Kubernetes*, Middletown: Stephen Fleming, 2018.
- [11] W3.org, "What Is Hypertext," W3, 1965. [Online]. Available: <https://www.w3.org/WhatIs.html>. [Diakses 26 July 2019].
- [12] H. H. F. M. E. N. M. C. C. F. D. O. David Booth, "Web Service Architecture," W3C Working Group Note 11 February 2004, 11 February 2004. [Online]. Available: <https://www.w3.org/TR/2004/NOTE-ws-arch-20040211/#relwwwrest>. [Diakses 22 10 2018].
- [13] C. Richardson, *Microservices Patterns*, New York: Manning Publications Co., 2019.

- [14] Yaml.org, “Yet Another Markup Language (YAML) 1.0,” YAML Organization, 10 December 2001. [Online]. Available: <https://yaml.org/spec/history/2001-12-10.html>. [Diakses 22 July 2019].
- [15] A. H. P. J. Armin Balalaie, “Microservices Architecture Enables DevOps,” *Migration to a Cloud-Native Architecture*, Vol. 1 dari 20740-7459, no. 16, pp. 42-52, 2010.
- [16] P. C. R. K. Len Bas, *Software Arcitecture in Practice*, New Jersey: Addison-Wesley, 2013.

Lampiran 1 Hasil Pengujian *Single Responsibility Principle*, dan *Common Closure Principle*



Lampiran 2 Hasil Akhir Dekomposisi



Lampiran 3 Perbandingan Proses Upgrade Monolitik dengan Microservice

Step	Monolitik	Microservice
1	<p>Initial Build Status : Pass Deploy : Old Version QA : Pending</p>	<p>Initial Build Status : Pass Deploy : Old Version QA : Pending</p>
2	<p>Upgrade profile 2.0 Build Status : Fail Deploy : Old Version QA : Pending</p>	<p>Upgrade Bio 2.0 Build Status : Pass Deploy : Mix QA : On Going</p>
3	<p>Half Upgraded profile 2.0 Build Status : Fail Deploy : Old Version QA : Pending</p>	<p>Half Upgrade Bio 2.0 Build Status : Pass Deploy : Mix QA : On Going</p>
4	<p>Fully Upgraded profile 2.0 Build Status : Pass Deploy : Old Version QA : Pending</p>	<p>Fully Upgraded Build Status : Pass Deploy : Upgraded Version QA : Done</p>
5	<p>Backup Build Status : Pass Deploy : Old Version QA : Pending</p>	
6	<p>Implement/Deploy Build Status : Pass Deploy : Upgrade Version QA : On Going</p>	
7	<p>Testing Build Status : Pass Deploy : Upgrade Version QA : Done</p>	<p>Legend</p> <p>Dependensi : —————→</p> <p>Proses Peralihan : - - - - -></p> <p>Tidak terhubung : ————x—————></p> <p>Build Status : Fail : Tidak bisa di compile : Success : Compile sukses</p> <p>QA : Pending : Belum melakukan QA : On Going : Sedang melakukan QA : Done : Sudah melakukan QA</p> <p>Deploy : Old Version : Microservice versi lama : Upgraded Version : Microservice baru : Mix : Campuran Microservice baru dan lama</p> <p>● Pass ● Fail/Pending ● On Going</p>

Lampiran 4 Wawancara Narasumber 1

Analisis Desain Berbasis Arsitektur Microservice dengan Representational State Transfer (REST) (Studi Kasus: Heartenly.com)

Identitas Narasumber

Nama : Hendy Irawan, S.T., M.T.

NIK : 3273071412830007

Jenis Kelamin : Laki-laki

Jabatan : Pemilik Heartenly.com

Pertanyaan :

1. Apa tujuan Heartenly.com menggunakan arsitektur *microservice*?
2. Bagaimana arsitektur aplikasi yang saat ini sedang berjalan?
3. Apa saja domain yang ada pada sistem informasi Heartenly.com?
4. Bagaimana proses validasi hasil dekomposisi yang sudah dilakukan?
5. Apakah hasil evaluasi hasil dekomposisi menggunakan prinsip SRP dan CCP sudah sesuai?

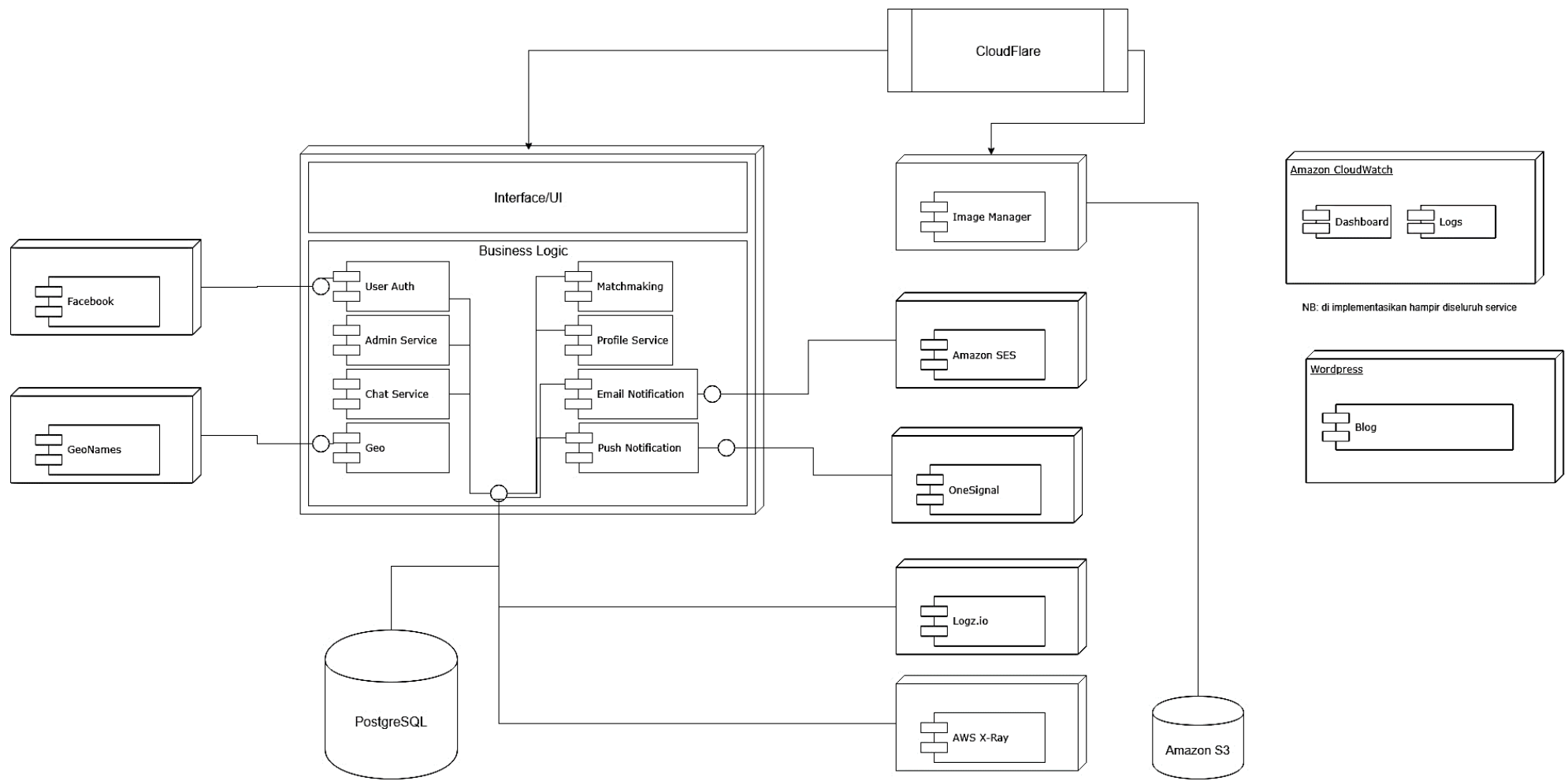
Jawaban :

1. Tujuan Heartenly.com menggunakan arsitektur *microservice* diantaranya:
 - a. Untuk meningkatkan performa aplikasi (Skalabilitas, Independensi, dan Ease of Deployment)
 - b. Mempersiapkan lingkungan agar dapat selalu mengikuti perkembangan teknologi
 - c. Untuk ikut bersaing dengan kompetitor dalam hal penerapan teknologi
 - d. Mempermudah pembagian tugas developer
2. Arsitektur saat ini masih menggunakan monolitik atau bisa dibilang semi-monolitik karena ada beberapa modul yang benar-benar terpisah, jika digambarkan kurang lebih seperti yang bisa dilihat pada **Gambar 6.2 Arsitektur Monolitik Heartenly.com yang sedang Berjalan**.
3. Penentuan domain ini ditentukan dari hasil identifikasi anda, kemudian saya koreksi. Domain yang anda



tentukan ini sudah benar.

Gambar 6. 1 Penentuan Domain



Gambar 6. 2 Arsitektur Monolitik Heartenly.com yang sedang Berjalan

4. Anda buat hasil dekomposisi kemudian saya koreksi, langkah ini diulang sampai saya merasa cukup dengan hasil dekomposisinya.
- a. Iterasi Pertama

N o	Domain Context	Microservices	Description
1	Profile Service	Registration	Basic Profile just to get labeled as "Registered User"
		Bio Service	Detail profile describing user including body type
		Public Service	User interests such type of kind woman etc
		Topic Services	User interests about hobby view of life etc
2	Authentication Service	Login Service	Handling Login and O-auth using social media
		Verification Service	Handling all verification user such phone number, email, (To be Confirmed Job, Social Security Number (NIK), address, salary)
3	Social Media Service	Posts Service	Handling user posts(Only Word), like and comments
		Chat Service	Handling chat between user
		Rating Service	Handling rate of user like "like or emoticon"
4	Matchmaking Service	Matchmaking Service	Suggest user to other user and vice versa
		(To be Confirmed) User Behavioral Services	Analyse User behavioral activity within di app
5	Legal Service	Legal Service	Handling Term and Condition for user including Policy
6	Customer Service	Customer Service	Handling FAQ, User complaint etc
		Testimonial Service	Handling success story of user, testimonial about the app etc
		Newsletter Service	Handling News about the app
		Blog Service (HOT)	Handling Blog about the app and share related articles
7	Admin Service	User Management Service	Handling User management such as blocking deleting etc
		Membership Service	Handling membership process for user
		Report Service	Handling User report of other User, analyz any possibility of fake user
8	Image Service	Resizing Image Service	Reduce the size of uploaded image to relevant size according to the needs

Pada tahap ini, coba tambahkan kolom fungsionalitas dari subdomainnya agar lebih jelas.

- b. Iterasi Kedua

N o	Domain Context	Microservices	Functionality	Description
1	Profile	Registration Service	Register New User	Basic Profile just to get labeled as "Registered User"
		Bio Service	Manage Personal Detail of user	Detail profile describing user including body type
			Manage Job Detail	Including salary
			Manage Education Detail	
		Preposess Service	Manage Public Profile	User interests such type of kind woman etc
			Manage Criteria of Person to be sought	

N o	Domain Context	Microservices	Functionality	Description
2	Authentication	Login Service	Login for registered user	Handling Login and O-auth using social media
			Forgot Password	
			Forgot Email	
		Verification Service	Phone Verification	Handling all verification user such phone number, email, (To be Confirmed Job, Social Security Number (NIK), address, salary)
			Email Verification	
			NIK Verification	
			Job Verification	
			Salary Verification	
3	Social Media	Posts Service	Post Status	Handling user posts(Only Word), like and comments
			Like Status	
			Comment Status	
		Chat Service	Manage Inbox	Handling chat between user
			Realtime Chat	
		Rating Service	Store rating of users	Suggest user to other user and vice versa
			Processing users rating	
4	Matchmaking	Matchmaking Service	find user base on gender type	Suggest user to other user and vice versa
			find user base on religion	
			find user base on culture	
			find user base on region	
			Suggest other users to "user"	
		(To be Confirmed) User Behavioral Services	Record visitor (user and non user) click activity	Analyse User behavioral activity within di app
5	Customer Relation	Customer Service	Manage FAQ	Handling FAQ, User complaint etc
			Handling User Complaint	
			Store User Suggestion	
		Testimonial Service	Handling user's testimonial	Handling success story of user, testimonial about the app etc
			Handling user success stories	
		Newsletter Service	Create News	Handling News about the app
			Broadcast News	
		Blog Service (HOT)	Manage Articles blog	Handling Blog about the app and share related articles
6	Admin	User Management Service	Manage user	Handling User management such as blocking deleting etc

N o	Domain Context	Microservices	Functionality	Description
			Determine Featured User	
		Membership Service	Membership registration	Handling membership process for user
			Payment Verification	
		Report Service	Warn Problematic user	Handling User report of other User, analyz any possibility of fake user
			Determine punishment to user	
7	Image Processing	Image Service	Naming uploaded image	Reduce the size of uploaded image to relevant size according to the needs
			Reduce Size Uploaded Image	
			Resize Dimension Image	
8	Notification	Notification Service	Email Notification	Reduce the size of uploaded image to relevant size according to the needs
			Phone Notification	
			Push Notifications	

Coba tambahkan kolom perkiraan nama method yang akan digunakan dalam *source code*

c. Iterasi Ketiga

N o	Domain Context	Microservices	Functionality	Method Name Approximation
1	Profile	CRUD Profile Service	Manage personal account of user like email, phone number etc	Change Firstname
				Change Lastname
				Change Email
				Change Phone Number
		Bio Service	Manage Personal Detail of user	Fill Personal Detail
				Update Personal Detail
				Create Physical Detail
				Update Physical Detail
				Create New Address
				Update Address
			Manage Job Detail	Create New Job Details
				Update Job detail
			Manage Education Detail	Create New Education Detail
				Update Education Detail
				Delete Education Detail
		Public Service	Manage Public Profile	Create New Public Profile
				Update Public Profile
				Delete Public Profile
			Manage Criteria of Person to be sought	Choose Criteria
				Update Criteria

N o	Domain Context	Microservices	Functionality	Method Name Approximation
				Delete Criteria
2	Authentication	Registration Service	Register New User	Create New User
				Create New User from O-Auth
		Login Service	Login for registered user	Login By Email
				Login By Phone Number
				Login By Username
				Login By O-Auth
			Forgot Password	Request Reset Password By Email
				Request Reset Password By Phone Number
			Forgot Email	Request Email Reminder By Phone Number
		Verification Service	Phone Verification	Send OTP Code
				Matching OTP Code
				Update Phone Verification Status
			Email Verification	Send Verification Link
				Update Email Verification Status
			NIK Verification	Upload Self Portrait with KTP Next To It
				Verification Image
			Job Verification	Send Verification Link
				Update Job Verification Status
			Salary Verification	Upload Salary Proof
				Update Salary Verification Status
3	Social Media	Posts Service	Post Status	Create New Status
				Update Status
				Delete Status
			Like Status	Like Status
				Unlike Status
			Comment Status	Create New Comment
				Update Comment
				Delete Comment
		Chat Service	Manage Inbox	Create New Chat
				Create New Message
				Delete Message
				Delete Chat
			Realtime Chat	Run Realtime Chat
		Rating Service	Store rating of users	Post Rating of User
				Delete Rating of User
			Processing users rating	Calculate Rating
4	Matchmaking	Matchmaking Service	find user base on gender type	Matchmaking By Gender

N o	Domain Context	Microservices	Functionality	Method Name Approximation
			find user base on religion	Matchmaking By Religion
			find user base on culture	Matchmaking By Culture
			find user base on region	Matchmaking By Region
			Suggest other users to "user"	Create New Recommendation
				Update Recommendation
				Delete Recommendation
		(To be Confirmed) User Behavioral Services	Record visitor (user and non user) click activity	Count Clicked Menu
				Count Viewed Profile
				Count How Often Open Page
				Count Spend Time
5	Customer Relation	Customer Service	Manage FAQ	Create New List FAQ
				Update List FAQ
				Delete List FAQ
				Delete Item in List FAQ
			Handling User Complaint	Show Complaint Form
				Send Complaint Received Message
				Send Complaint Answer
				Create Complaint Report
				Count Complained Topic
			Store User Suggestion	Show Suggestion Form
				Show Suggestion List
				Mark Suggestion
		Testimonial Service	Handling user testimonial	Create New Testimoni
				Create New Success Story
				Update Testimoni
				Update Success Story
				Delete Testimony
				Delete Success Story
		Newsletter Service	Create News	Create News
			Broadcast News	Broadcast News
		Blog Service (HOT)	Manage Articles blog	Create Article
				Update Article
				Delete Article
6	Admin	User Management Service	Manage user	Show User List
				Suspend User
				Block User
			Determine Featured User	Count User Flag
				Make Featured User
				Update Featured User
		Membership Service	Membership registration	Show Membership Step

N o	Domain Context	Microservices	Functionality	Method Name Approximation
				Help
				Show Membership Form
			Payment Verification	Verificate Payment
				Enable Membership Feature
		Report Service	Warn Problematic user	User Flag Negative
				User Flag Positive
				Send Warning Message
				Count Reported User
7	Image Processing	Image Service	Reduce Size Uploaded Image	Check Image Size
				Reduce Image Size
			Resize Dimension Image	Create Thumbnail Photo Dimension
				Create Profile Photo Dimension
				Create Profile Photo for Chat Dimension
8	Notification	Notification Service	Email Notification	Send Email Notification
			Phone Notification	Send Phone Notification
			Push Notifications	Send Push Notification

5. Perlihatkan dalam bentuk table hasil evaluasinya.

a. Penentuan Deskripsi Subdomain

Subdomain	Deskripsi
CRUD Profile Service	Bertanggung jawab untuk mengelola data personal seperti email, nomor telepon dan lainnya
Wizard Service	Bertanggung jawab untuk menangani pengolahan data detail pengguna seperti pekerjaan, pendidikan dan data personal lainnya
Public Service	Bertanggung jawab untuk mengelola public profile, halaman profil yang dilihat oleh user lain dan mengelola kriteria orang yang dicari
Authentication Service	Bertanggung jawab untuk mengatur user yang masuk dan autentikasi user seperti Login, lupa password serta lupa email
Verification Service	Bertanggung jawab untuk memverifikasi keaslian data yang di input pada Bio Service seperti nomor ktp, gaji, nomor telepon dan yang lainnya
Posts Service	Bertanggung jawab untuk mengelola status, like dan komentar user terhadap user lain singkatnya seperti fungsi sosial media
Chat Service	Bertanggung jawab untuk mengelola perpesanan atau obrolan antar user meliputi realtime chatting dan kotak pesan
Rating Service	Bertanggung jawab untuk mengelola data rating user yang diberikan user lainnya
Matchmaking Service	Bertanggung jawab untuk melakukan penjemputan atau merekomendasikan satu user ke user lain berdasarkan parameter yang sudah ditentukan, seperti jenis kelamin, agama, budaya, dan yang lainnya
User Behavioral Service	Bertanggung jawab untuk merekam aktivitas klik visitor dan user terhadap profile user yang ada tampilkan di halaman utama atau halaman yang menampilkan daftar profil user yang bisa dilihat oleh user dan visitor

Customer Service	Bertanggung jawab untuk mengelola hubungan user dengan Heartenly sebagai pemilik aplikasi, dengan cara menangani komplain, kritik, saran, dan membuat daftar pertanyaan yang sering di tanyakan oleh pengguna
Testimonial Service	Beratnggun jawab untuk mengelola cerita sukses dari user atau testimoni user terhadap aplikasi
Newsletter Service	Bertanggung jawab untuk mengelola pemberitaan seputar aplikasi seperti fitur baru, perubahan sistem dan yang lainnya, dan dikirim secara broadcast melalui email
Blog Service	Bertanggung jawab untuk mengelola blog yang dimiliki Heartenly yang berisi kumpulan artikel pilihan.
User Management Service	Bertanggung jawab untuk mengelola user. Microservice ini hanya bisa diakses oleh admin
Membership Service	Bertanggung jawab untuk mengelola membership user dari registrasi sampai pembayaran
Report Service	Bertanggung jawab untuk mengelola pelaporan user terhadap user lainnya. Jika terbukti bersalah microservice ini dapat memperingatkan user terlapor sampai mem-banned akun terlapor
Image Processing	Bertanggung jawab untuk mengelola gambar dari upload gambar yang user unggah
Notification Service	Bertanggung jawab untuk mengelola notifikasi untuk user

b. Penentuan Tanggung Jawab Subdomain

<i>No</i>	<i>Domain Context</i>	<i>Microservices</i>	<i>Responsibility</i>
<i>1</i>	<i>Profile</i>	<i>CRUD Profile</i>	<i>Manage Personal Information</i>
		<i>Wizard</i>	<i>Manage Detail Information of User</i>
		<i>Prepossess</i>	<i>Manage Public Profile</i>
		<i>Criteria Management</i>	<i>Manage Criteria of Person to be sought</i>
<i>2</i>	<i>Authentication</i>	<i>Login</i>	<i>Manage Authentication</i>
		<i>Registration</i>	<i>Manage Registration</i>
		<i>Verification Service</i>	<i>Manage all Verifiaction Process</i>
<i>3</i>	<i>Social Media</i>	<i>Post</i>	<i>Manage Status</i>
		<i>Chat</i>	<i>Manage Chat of users</i>
		<i>Rating</i>	<i>Manage rating of users</i>
<i>4</i>	<i>Matchmaking</i>	<i>Matchmaking Service</i>	<i>Manage Matching user between user</i>
		<i>Cari Kenalan</i>	<i>Manage user search</i>
		<i>User Behavioral Services</i>	<i>Manage Recording visitor (user and non user) click activity</i>
<i>5</i>	<i>Customer Relation</i>	<i>Customer Service</i>	<i>Manage User feedback</i>
		<i>Testimonial Service</i>	<i>Manage user testimonial</i>
		<i>Newsletter Service</i>	<i>Manage News</i>
		<i>Blog Service (HOT)</i>	<i>Manage Articles blog</i>
<i>6</i>	<i>Admin</i>	<i>User Management Service</i>	<i>Manage user</i>
		<i>Membership Service</i>	<i>Manage Membership registration</i>
		<i>Report Service</i>	<i>Warn Problematic user</i>
<i>7</i>	<i>Image Processing</i>	<i>Image Service</i>	<i>Reduce Size Uploaded Image</i>

<i>No</i>	<i>Domain Context</i>	<i>Microservices</i>	<i>Responsibility</i>
<i>8</i>	<i>Notification</i>	<i>Notification Service</i>	<i>Email Notification</i>

c. Hasil SRP dan CCP

<i>No</i>	<i>Domain Context</i>	<i>Subdomain</i>	<i>Responsibility</i>
<i>1</i>	<i>Profile</i>	<i>Bio</i>	<i>Manage Personal Detail User Include Criteria</i>
		<i>Prepossess</i>	<i>Manage Public Profile</i>
<i>2</i>	<i>Authentication</i>	<i>Authentication</i>	<i>Manage Authentication include Registration</i>
		<i>Verification</i>	<i>Manage all Verifiaction Process</i>
<i>3</i>	<i>Social Media</i>	<i>Posts</i>	<i>Manage Status</i>
		<i>Chat</i>	<i>Manage Chat of users</i>
		<i>Rating</i>	<i>Manage rating of users</i>
<i>4</i>	<i>Matchmaking</i>	<i>Matchmaking</i>	<i>Manage Matching user between user include searching user</i>
		<i>User Behavioral</i>	<i>Manage Recording visitor (user and non user) click activity</i>
<i>5</i>	<i>Customer Relation</i>	<i>Customer</i>	<i>Manage User feedback</i>
		<i>Testimonial</i>	<i>Manage user testimonial</i>
		<i>Newsletter</i>	<i>Manage News</i>
		<i>Blog Service (HOT)</i>	<i>Manage Articles blog</i>
<i>6</i>	<i>Admin</i>	<i>User Management</i>	<i>Manage user</i>
		<i>Membership</i>	<i>Manage Membership registration</i>
		<i>Report</i>	<i>Warn Problematic user</i>
<i>7</i>	<i>Image Processing</i>	<i>Image</i>	<i>Manage Image</i>
<i>8</i>	<i>Notification</i>	<i>Notification</i>	<i>Manage Notification</i>

Lampiran 5 Wawancara Narasumber 2
Analisis Desain Berbasis Arsitektur Microservice dengan Representational State Transfer (REST) (Studi Kasus: Heartenly.com)

Identitas Narasumber

Nama : Devi Oktaviani, S.Kom.
NIK : 3273071412830007
Jenis Kelamin : Perempuan
Jabatan : Business Development Heartenly.com
Pertanyaan :

1. Domain bisnis Heartenly.com itu terdiri dari domain apa saja?
2. Apa itu matchmaking? dan bagaimana cara kerjanya?
3. Bagaimana Roadmap pengembangan Heartenly.com?

Jawaban :

1. Domain bisnis saat ini yang sedang berjalan adalah
 - a. Registrasi
 - b. Uploadfoto
 - c. Cari-Kenalan
 - d. Messenger
 - e. Crud-Profile
 - f. Social-Media (Profile, Like, Emoji)
 - g. Matchmaking
2. Fitur heartenly yang berfungsi untuk menjodohkan penggunanya, saat ini masih menggunakan pemanggilan query berdasarkan jenis kelamin saja
3. Kedepannya Heartenly akan melakukan pengembangan pada domain yang sudah ada dan menambah domain baru
 - a. Registrasi
 - b. Uploadfoto
 - c. Cari-Kenalan
 - d. Messenger
 - e. Crud-Profile
 - f. Social-Media (Profile, Like Status)
 - g. Wizard
 - h. Newsletter
 - i. Matchmaking Berdasarkan Culture, Religion, Region/Area
 - j. Report
 - k. Rating User
 - l. Upgrade Membership
 - m. Heartenly Hot
 - n. Post Status

Lampiran 6 Surat Pernyataan Narasumber 1

Surat Pernyataan Menyetujui Hasil Dekomposisi Layanan Heartenly.com






Yang bertanda tangan di bawah ini :

Nama : Hendy Irawan, S.T., M.T.

Jenis Kelamin : Laki-laki

Jabatan : Pemilik Heartenly.com

Menyatakan menyetujui hasil penelitian yang Kautsar Tisamawi rekomendasikan, diantaranya:

No	Jenis	Paraf
1	Deskripsi setiap subdomain	
2	Penentuan Tanggung Jawab	
3	Hasil Dekomposisi	
4	Hasil Evaluasi Menggunakan SRP dan CCP	
5	Hasil Desain Microservice	

Rincian penelitian dari tabel diatas dilampirkan pada halaman selanjutnya. Proses validasi desain dilakukan melalui aplikasi WhatsApp yang diiterasi beberapa kali. Sehingga menghasilkan dekomposisi layanan yang dirasa cukup merepresentasikan kebutuhan dari Heartenly.com

Demikian surat pernyataan ini dibuat untuk dipergunakan sebagaimana mestinya.

Bandung 23 Juli 2019


..... Hendy Irawan

Lampiran 7 Surat Pernyataan Narasumber 2

Surat Pernyataan Menyetujui Hasil Dekomposisi Layanan Heartenly.com

Yang bertanda tangan di bawah ini :

Nama : Devi Oktaviani, S.Kom.

Jenis Kelamin : Perempuan

Jabatan : Business Development Heartenly.com

Menyatakan menyetujui hasil penelitian yang Kautsar Tisamawi rekomendasikan, diantaranya:

No	Jenis	Paraf
1	Deskripsi setiap subdomain	Devi
2	Penentuan Tanggung Jawab	Devi
3	Hasil Dekomposisi	Devi
4	Hasil Evaluasi Menggunakan SRP dan CCP	Devi
5	Hasil Desain Microservice	Devi

Rincian penelitian dari tabel diatas dilampirkan pada halaman selanjutnya. Proses validasi desain dilakukan melalui aplikasi WhatsApp yang diiterasi beberapa kali. Sehingga menghasilkan dekomposisi layanan yang dirasa cukup merepresentasikan kebutuhan dari Heartenly.com

Demikian surat pernyataan ini dibuat untuk dipergunakan sebagaimana mestinya.

Bandung 23 Juli 2019

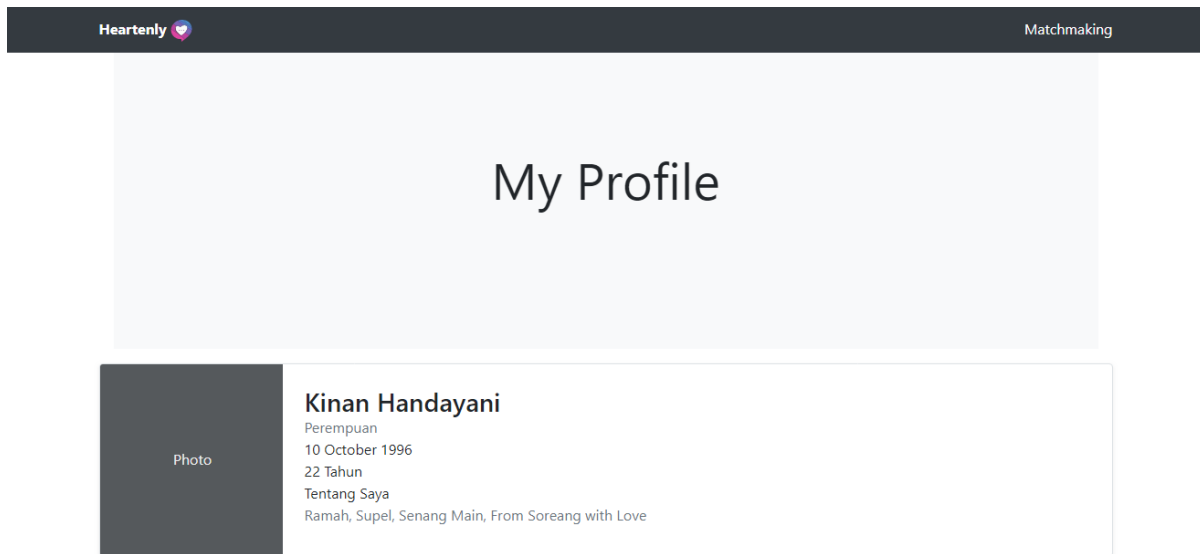
Devi

Lampiran 8 *Prototype Microservice Heartenly.com*

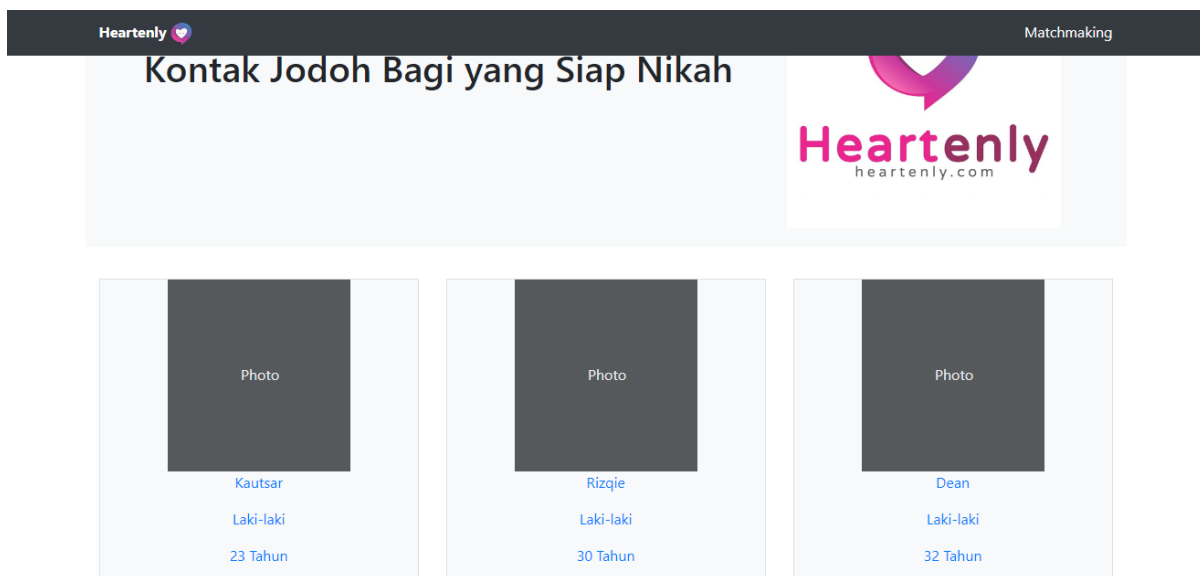
Prototype ini dibangun berdasarkan Skenario Evaluasi, melibatkan lima (5) microservice yaitu:

1. Frontend
2. Bio
3. Bio 2
4. Matchmaking
5. Public

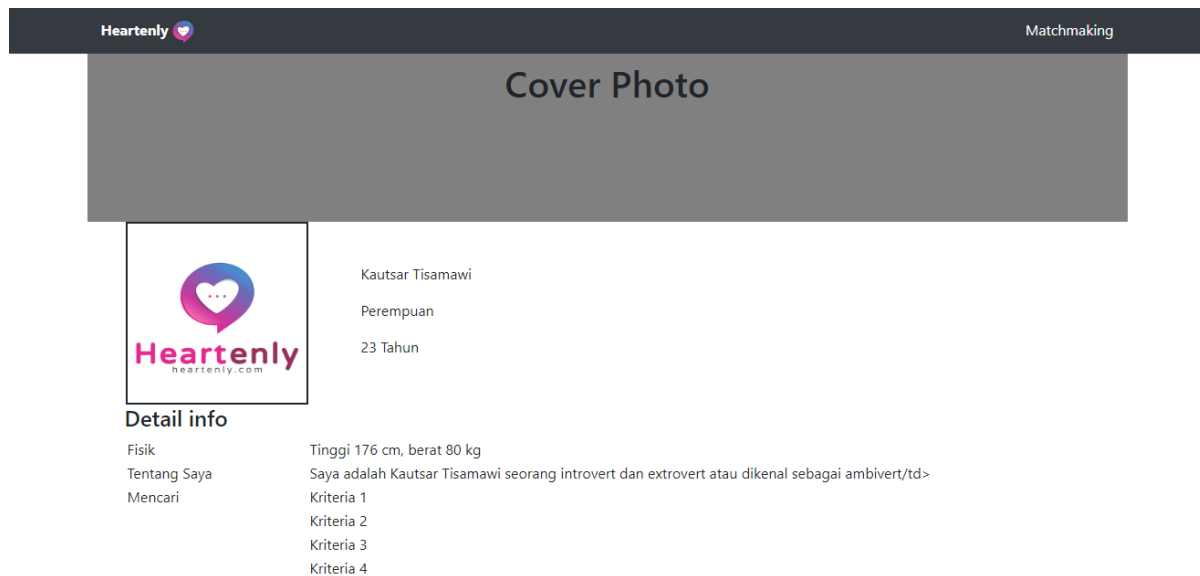
Berikut ini tampilan dari *prototype* terdiri dari empat *microservice* yang terhubung, yaitu Frontend, Matchmaking, Public, dan Bio.



Gambar 8. 1 Halaman Utama yang Datanya diperoleh dari Bio



Gambar 8. 2 Halaman Matchmaking



Gambar 8. 3 Halaman Public Profile

Pada Skenario yang digunakan di asumsikan Matchmaking yang pada awalnya mempunyai dependensi ke Bio akan berpindah dependensinya ke Bio 2

```
class PagesController extends Controller
{
    public function index() { //Halaman My Profile
        $headers = array('Accept' => 'application/json');
        $data = array('name' => 'ahmad', 'company' => 'mashape');

        $body = \Unirest\Request\Body::json($data);

        $response = \Unirest\Request::get('url: 'http://heartenly.bio/api/detail_profile/'.4', $headers, $body);

        return view( view: 'pages.index', ["data" => $response->body]);
    }

    public function dashboard()
    { // Halaman Matchmaking
        $headers = array('Accept' => 'application/json');
        $data = array('name' => 'ahmad', 'company' => 'mashape');

        $body = \Unirest\Request\Body::json($data);

        $response = \Unirest\Request::get('url: 'http://heartenly.matchmaking/index.php/public_profile/public_data', $headers, $body);

        return view( view: 'pages.dashboard', ["data" => $response->body]);
    }

    public function viewprofile($id)
    { //Halaman Public Profile
        $headers = array('Accept' => 'application/json');
        $data = array('name' => 'ahmad', 'company' => 'mashape');

        $body = \Unirest\Request\Body::json($data);

        $response = \Unirest\Request::get('url: 'http://heartenly.pp/index.php/Dp/view_detail/'. $id, $headers, $body);
    }
}
```

\App\Http\Controllers > PagesController > viewprofile()

Gambar 8. 4 Source Code Frontend

Source Code diatas menggambarkan dependensi frontend kepada microservice yang lain, yaitu Matchmaking, Bio, dan Public Profile. Dependensi dapat terlihat pada source code bagian URL

```

public function __construct()
{
    parent::__construct();

    $this->load->library('HttpClient', array(
        'headers' => array(
            'Authorization: SomekeyHere',
            'Content-Type: application/json',
        ),
        'data' => array(
            'id_account' => 'somedata',
        ),
        'url' => 'http://heartenly.bio/api/detail_profile',
    ));
}

public function view_detail($id)
{
    $this->httpclient->setOptions(array(
        'headers' => array(
            'Authorization: SomekeyHere',
            'Content-Type: application/json',
        ),
        'url' => 'http://heartenly.bio/api/detail_profile/' . $id, //Dependensi ke Bio
    ));
    $this->httpclient->get();
    $data = $this->httpclient->getResults();

    echo $data;
}
}

```

Gambar 8. 5 Source Code Public Profile

Microservice Public Profile mendapatkan data dari *microservice* Bio dengan berkomunikasi melalui REST. Data yang diperoleh ditampilkan oleh *frontend*

```

/**
 * PublicProfileController constructor.
 */
public function __construct()
{
    parent::__construct();

    $this->load->library('HttpClient', array(
        'headers' => array(
            'Authorization: SomekeyHere',
            'Content-Type: application/json',
        ),
        'data' => array(
            'somekey' => 'somedata',
            'anotherkey' => 'anotherkeydata'
        ),
        'url' => 'http://heartenly.bio/api/account', //URL ini menandakan dependensi kepada Bio lama
        // 'url' => 'http://v2.heartenly.bio/api/account', //URL ini menandakan dependensi kepada Bio 2
    ));

}

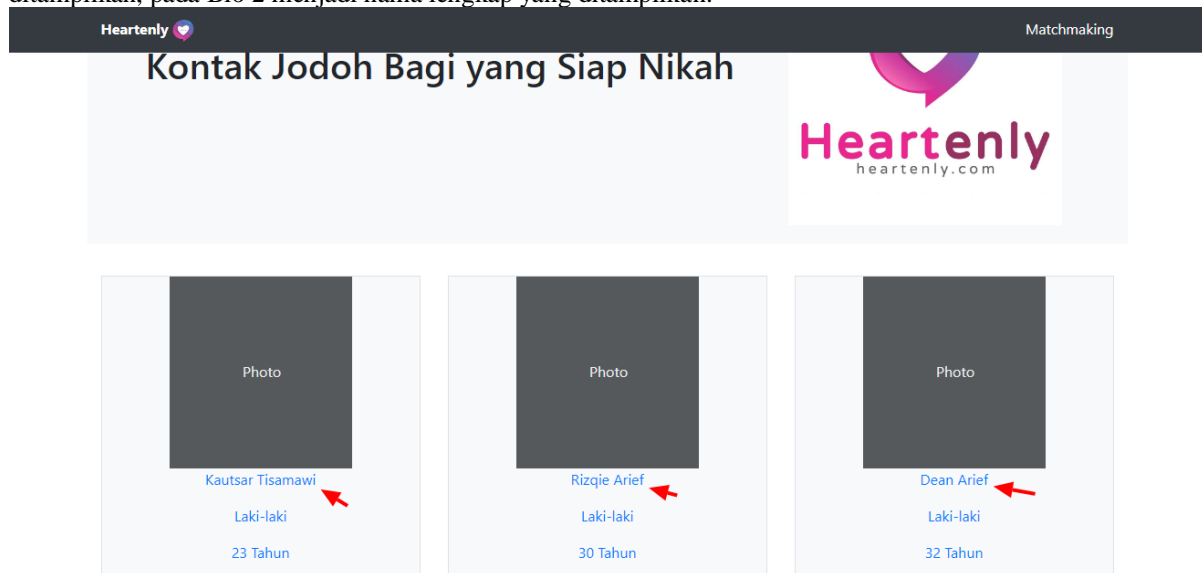
function public_data() {
    $this->httpclient->get();
    $data = $this->httpclient->getResults();

    echo $data;
}
}

```

Gambar 8. 6 Source Code Matchmaking

Mengikuti skenario yang sebelumnya dibahas, diasumsikan bahwa *microservice* Bio akan di upgrade menjadi Bio 2, untuk itu semua *microservice* yang mempunyai dependensi ke Bio harus beralih ke Bio 2. Source code diatas menggambarkan peralihan dependensi *microservice* matchmaking dari Bio ke Bio 2. Untuk menggambarkan perbedaan Bio dan Bio 2, pada halaman matchmaking yang awalnya hanya nama depan yang ditampilkan, pada Bio 2 menjadi nama lengkap yang ditampilkan.



Gambar 8. 7 Halaman Matchmaking Setelah Beralih ke Bio 2