Cromosom: *Aplikasi Crowdsourced Software Engineering* Menggunakan Komponen Rekomendasi *Task* Berbasis Media Sosial

Kurnia Ramadhan Putra [1], Muhammad Zuhri Catur Candra [2]
[1] Program Studi Sistem Informasi, Fakultas Teknik Industri, ITENAS, Bandung
[2] Program Studi Magister Informatika, STEI, ITB, Bandung
Email: kurniaramadhan@itenas.ac.id [1], catur@informatika.org [2]

ABSTRAK

Konsep crowdsourcing dapat dimanfaatkan pada bidang rekayasa perangkat lunak yang dikenal dengan Crowdsourced Software Engineering (CSE). CSE digunakan untuk menyelesaikan task yang berkaitan dengan perangkat lunak, seperti desain, implementasi, dan pengujian perangkat luna serta perbaikan bug. Saat ini, permasalahan umum yang terjadi aplikasi CSE adalah worker menghabiskan waktu untuk menemukan task yang relevan sesuai dengan keahliannya dan requester sulit untuk memilih worker yang dapat dipercaya untuk mengerjakan task. Komponen sistem rekomendasi dapat diintegrasikan pada aplikasi CSE yang dipercaya mampu untuk mengatasi permasalahan tersebut. Beberapa penelitian yang ada tentang integrasi komponen sistem rekomendasi pada aplikasi CSE hanya untuk menangani rekomendasi task dan tidak memepertimbangkan trustworthiness dari worker yang akan mengerjakan task. Sedangkan pada penelitian ini, integrasi komponen sistem rekomendasi selain dapat membantu merekomendasikan task kepada worker juga melakukan perangkingan terhadap worker berdasarkan trustworthiness dari worker tersebut sehingga dapat menjadi pertimbangan untuk requester dalam memilih worker yang akan mengerjakan task. Pendekatan yang diusulkan pada penelitian ini adalah kombinasi antara pendekatan content based dengan individual based. Pendekatan content based untuk menangani proses pencocokan antara kebutuhan keahlian yang diperlukan untuk mengerjakan task dengan kualifikasi worker yang akan mengerjakan task. Sedangkan pendekatan individual based untuk menangani proses perhitungan nilai social profile dalam menghasilkan trustworthiness dari worker. Implementasi dilakukan dengan mengembangkan aplikasi CSE yang dikenal dengan Cromosom, yang diintegrasikan dengan komponen sistem rekomendasi untuk membantu merekomendasikan task kepada worker dan melakukan perangkingan terhadap worker berdasarkan trustworthiness dari worker tersebut. Dari hasil pengujian fungsionalitas yang dilakukan, aplikasi Cromosom dapat membantu worker untuk menemukan task yang lebih relevan sesuai dengan keahliannya, dan membantu requester dalam memilih worker yang memiliki trustworthiness untuk mengerjakan task.

Kata kunci: *crowdsourcing, crowdsourced software engineering, task recommendation.*

ABSTRACT

The concept of crowdsourcing can be utilized in the field of software engineering known as Crowdsourced Software Engineering (CSE). CSE is used to address tasks related to software, such as design, implementation, and testing of software and bug fixes. Currently, a common problem that occurs with CSE applications is that workers spend time finding relevant tasks according to their expertise and requester is difficult to choose workers who can be trusted to do the task. The recommendation system component can be integrated into CSE applications that are believed to be able to overcome these problems. Some existing research on the integration of recommendation system components in CSE applications is only to handle task recommendations and not consider the

trustworthiness of workers who will be working on tasks. Whereas in this study, the integration of recommendation system components in addition to being able to help recommend tasks to workers also rank workers based on the trustworthiness of the workers so that they can be considered for the requester in choosing workers who will do the task. The approach proposed in this study is a combination of content-based and individual-based approaches. Content-based approach to handle the matching process between the skill requirements needed to do the task and the qualifications of the worker who will be working on the task. While the individual-based approach to handle the process of calculating the value of social profiles in generating trustworthiness from workers. Implementation is done by developing a CSE application known as a Cromosom, which is integrated with the recommendation system component to help recommend tasks to workers and rank workers based on the trustworthiness of the worker. From the results of the functionality testing, the Cromosom application can help workers find more relevant tasks according to their expertise, and help requester in choosing workers who have trustworthiness to do the task.
Keywords: *crowdsourcing, crowdsourced software engineering, task recommendation.*

1. PENDAHULUAN

Perusahaan dapat mengalihdayakan pekerjaaan kepada tenaga kerja yang ada di luar perusahaan yang dikenal dengan istilah outsourcing. Permasalahan yang biasa muncul dalam kegiatan outsourcing adalah sulitnya menemukan tenaga kerja yang memiliki keahlian terkait dengan pekerjaan yang diberikan dan mahalnya upah yang harus dibayarkan. Crowdsourcing diperkenalkan oleh Jeff Howie pada tahun 2006 yang mendefinisikan crowdsourcing sebagai aktivitas untuk mengalihdayakan pekerjaan ke sekelompok orang yang terhubung ke jaringan internet secara open call [1]. Crowdsourcing sebagai solusi alternatif untuk mengurangi biaya yang harus dikeluarkan perusahaan dan memanfaatkan tenaga kerja lebih efisien [2], selain itu juga meningkatkan produktivitas dari perusahaan [3].

Crowdsourcing dapat dimanfaatkan dalam bidang Rekayasa Perangkat Lunak yang dikenal dengan istilah Crowdsourced Software Engineering (CSE). Beberapa pekerjaan pada CSE diantaranya desain arsitektur, pengembangan komponen, dan perbaikan bug [4]. Kegiatan dalam memilih task pada platform CSE saat ini masih dilakukan secara manual sehingga menghabiskan waktu dan sulitnya menemukan worker yang dapat dipercaya untuk mengerjakan task [5]. Salah satu platform CSE yang terkenal adalah Top Coder, dimana ada sekitar 60 task aktif setiap harinya yang dipilih secara manual oleh worker. Sistem rekomendasi dapat diintegrasikan pada platform CSE untuk membantu worker dalam menemukan task yang relevan dan menganilisi preferensi worker terhadap task yang akan dikerjakan [6].

Ada beberapa penelitian yang telah dilakukan mengenai integrasi sistem rekomendasi pada platform crowdsourcing. Penelitian yang dilakukan oleh M. C. Yuen dkk (2011) mengusulkan sebuah algoritma untuk melakukan pencocokan antara keahlian worker dengan task yang direkomendasikan [7]. King dkk (2014) memperkenalkan sebuah framework yang diberi nama TaskRec yang mampu merekomendasikan item cold start kepada worker [8]. Yang dkk (2017) mengusulkan metode yang memanfaatkan deskripsi task dan dan task yang populer untuk direkomendasikan kepada cold start developer.

Pada penelitian sebelumnya, platform crowdsourcing hanya untuk menangani rekomendasi task kepada worker tetapi tidak mempertimbangkan trustworthiness dari worker yang akan mengerjakan task. Sedangkan pada penelitian ini, selain merekomendasikan task kepada worker juga

merekomendasikan worker kepada requester berdasarkan trustworthiness, sehingga menjadi dasar pertimbangan requester dalam memilih worker yang akan mengerjakan task.

Media sosial dapat dimanfaatkan untuk menjaring worker dengan keahlian beragam [9]. Github adalah repositori kode program sekaligus media sosial yang digunakan untuk berkolaborasi dalam mengembangkan perangkat lunak. Pada penelitian ini, Github digunakan untuk menjaring worker dengan keahlian beragam dan membangun trustworthiness dari worker berdasarkan reputasi, kredensial, dan keahliannya. Berdasarkan latar belakang tersebut, maka dikembangkan platform CSE yang dikenal dengan Cromosom untuk menangani task kode program yang diintegrasikan dengan sistem rekomendasi agar mampu merekomendasikan task kepada worker dan juga melakukan perangkingan terhadap worker berdasarkan trustworthiness sehingga menjadi dasark pertimbangan untuk requester dalam memilih worker yang akan mengerjakan task.

2. METODOLOGI

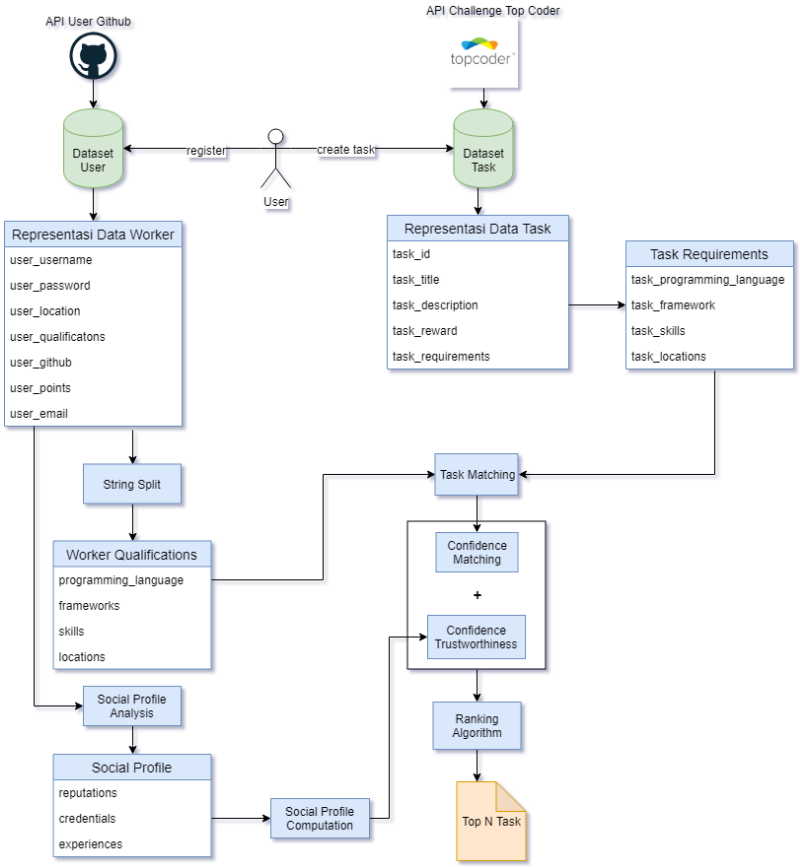
Cromosom adalah singkatan dari *Crowdsourced Software Engineering Application Using Recommendation Component Based on Social Media*. Ada dua aktor pada aplikasi Cromosom, yaitu *requester* adalah *user* yang mempublikasikan *task* dan *worker* adalah *user* yang mengerjakan *task*, sedangkan *task* adalah pekerjaan yang harus diselesaikan oleh *worker* dalam bentuk solusi kode program. Komponen sistem rekomendasi yang diintegrasikan pada aplikasi Cromosom sebagai berikut:

1) Komponen *Task Matching*, proses untuk mencocokkan antara keahlian yang dibutuhkan untuk mengerjakan task dengan kualifikasi *worker*.
2) Komponen *Social Profile Computation*, proses untuk menghitung *trustworthiness* dari *worker*.
3) Komponen *Top N Recommendation*, proses untuk melakukan perangkingan *task* yang akan direkomendasikan kepada *worker* berdasarkan hasil perhitungan dari *Task Matching* dan *Social Profile Computation*.

Untuk memberikan gambaran umum dari keterkaitan antar komponen pada arsitektur aplikasi Cromosom maka dapat dilihat “Gambar 1”.

Penjelasan dari Gambar 1 sebagai berikut:

1) Pengumpulan dataset user dilakukan dengan mengakses API user dari Github (https://developer.github.com/v3/users/), dimana *dataset* yang dihasilkan dalam format *file* .json kemudian data tersebut disimpan ke *database* Cromosom pada Tabel User. Daftar *task* yang direkomendasikan oleh aplikasi Cromosom secara bebas dapat dipilih oleh *worker* untuk dikerjakan.
2) *Dataset user* yang sudah disimpan pada *database* kemudian direpresentasikan menjadi data *worker* dimana dilakukan pemilihan atribut yang hanya dibutuhkan sebagai kualifikasi dari *worker* untuk digunakan pada proses *task matching*. *Requester* dapat melihat hasil pengerjaan *task*, kemudian dievaluasi dan hasil pengerjaan *task* yang paling baik akan diberikan imbalan sesuai dengan yang telah ditentukan.



Gambar 1. Gambaran Umum Arsitektur Aplikasi Cromosom

3) Pengumpulan *dataset task* dilakukan dengan mengakses API challenge dari Top Coder (www.topcoder.com), dimana dataset yang dihasilkan dalam format file json kemudian data tersebut disimpan ke *database* Cromosom.
4) *Dataset task* yang sudah disimpan pada *database* kemudian direpresentasikan menjadi data *task* dimana dilakukan pemilihan atribut yang hanya dibutuhkan untuk proses *task matching*.
5) Setelah diperoleh hasil representasi *task* dan representasi *user* maka selanjutnya dilakukan proses *task matching* menggunakan teknik *String Distance* untuk menghitung kemiripan antara keahlian yang dibutuhkan untuk mengerjakan *task* dengan kualifikasi *worker*. Nilai dari hasil dari proses *task matching* adalah *confidence matching* dimana nilainya berada pada rentang 0 sampai 1.
6) *Social Profile Analysis* dilakukan untuk mendapatkan parameter yang akan digunakan untuk menghitung nilai *trustworthiness* pada tahap *Social Profile Computation*, diantaranya reputasi, kredensial, dan pengalaman.
7) Proses dari *Social Profile Computation* akan menghasilkan nilai *confidence trustworthiness* dari *worker*.
8) Nilai *confidence matching* dikombinasikan dengan nilai *confidence trustworthiness* dengan bobot masing-masing 50% dan 50%, sehingga didapatkan nilai akhir *confidence* yang dirangking sehingga diperoleh *top-n task*.

2.1 Analisis *Task Matching*

Pada penelitian ini, dilakukan analisis bahwa salah satu teknik yang dapat digunakan untuk menangani proses *task matching* yaitu *String Distance*. *String Distance* adalah teknik untuk menghitung kemiripan antara dua *string*. Pada dataset *user* Github, user mendeskripsikan kualifikasinya dalam bentuk *string* kalimat atau kumpulan dari beberapa kata, sehingga kalimat tersebut perlu diuraikan

menjadi beberapa *string* kata, agar dapat dilakukan perhitungan jarak antara string kata dari keahlian yang dibutuhkan untuk mengerjakan *task* dengan *string kata* dari kualifikasi *worker*. Hasil pemisahan dari *string* kalimat menjadi *string* kata dapat dilihat pada "Tabel 1".

Tabel 1. Hasil Split Dari String Kalimat

String Kalimat	Array	String Kata
Javascript Java Spring AngularJS MySQL	0	Javascripts
	1	Java
	2	Spring
	3	AngularJS
	4	MySQL

Kemudian dilakukan perhitungan kemiripan antara keahlian yang dibutuhkan untuk mengerjakan *task* dengan kualifikasi *worker* menggunakan algoritma *Levenshtein Distance* seperti yang ditunjukkan pada "Tabel 2".

Tabel 2. Hasil Perhitungan Algoritma Levenshtein Distance

Kualifikasi Worker	Kebutuhan Keahlian		Levenshtein Distance
Javascript	Task1	javascript	0.9
	Task2	java	0.571
	Task3	Spring	0.3
	Task4	Angular	0.1
	Task5	MySQL	0.1

Jika nilai hasil perhitungan dari algoritma *Levenshtein Distance* mendekati 1, maka dapat dinyatakan bahwa kebutuhan keahlian terhadap *task* cocok dengan keahlian yang dimiliki oleh *worker*, sebaliknya jika mendekati 0 maka tidak ada kecocokan antara keahlian yang dibutuhkan untuk mengerjakan *task* kualifikasi yang dimiliki oleh *worker*.

Berdasarkan hasil perhitungan algoritma *Levenshtein Distance*, maka dapat dilakukan perangkingan terhadap *task* seperti pada "Tabel 3".

Tabel 3. Hasil Perangkingan Task

Kualifikasi Worker	Kebutuhan Keahlian		Levenshtein Distance (LD)	Task Ranking
Javascript	Task1	javascript	0.9	Task 1
	Task2	java	0.571	Task 2
	Task3	MySQL	0.1	Task 5
	Task4	Angular	0.1	Task 3
	Task5	Spring	0.3	Task 4

Dari "Tabel 3" dapat dilihat dari hasil perangkingan *task* bahwa *task* yang paling cocok direkomendasikan adalah Task1 dengan nilai tertinggi yaitu 0,9.

2.2 Analisis *Social Profile Computation*

Langkah-langkah dari analisis perhitungan *social profile* digunakan untuk mendapatkan nilai *trustworthiness* dari *worker*, yang dapat dijelaskan sebagai berikut:

1) Membangun Jaringan Sosial User
   Pada tahap ini dilakukan pembangunan jaringan sosial dari 1442 *user* berdasarkan relasi yang terjalin antar *user* tersebut. Relasi antar *user* dinotasikan dalam sebuah *array* dimana jika panjang *array* lebih besar dari nol maka menunjukkan adanya relasi antar satu *user* dengan *user* lainnya. Jaringan sosial *user* dibentuk menjadi sebuah graf yang terdiri dari sekumpulan *node* yang menunjukkan *user* dan sekumpulan *edge* yang menunjukkan relasi yang terjalin antar *user* tersebut
2) Merepresentasikan *Trustworthiness*
   Pada tahap ini dilakukan pembangunan jaringan sosial dari 1442 *user* berdasarkan relasi yang terjalin antar *user* tersebut. Relasi antar *user* dinotasikan dalam sebuah *array* dimana jika panjang *array* lebih besar dari nol maka menunjukkan adanya relasi antar satu *user* dengan *user* lainnya.
3) Perhitungan Nilai *Trustworthiness*
   Hasil perhitungan *reputation*, *credentials*, dan *experiences* digabungkan sehingga didapatkan nilai *trustworthiness*, dimana:

$$ConfidenceTrustworthiness = \left(\left(\frac{reputation}{3} + \frac{credentials}{3} + \frac{experiences}{3}\right) \times 100\%\right)$$

   Dimana,

   a. Skor $reputation$ adalah nilai kombinasi antara skor *success rate* dengan skor *relationship*.

$$reputation = \frac{success\ rate + relationship}{2}$$

   Dimana,

   - $success\ rate$ adalah jumlah *task* yang diterima oleh *requester* dibandingkan dengan jumlah *task* yang dikerjakan oleh *worker*.
     Contoh skenario: Jumlah *task* yang dikerjakan oleh worker adalah 5 *task* sedangkan jumlah *task* yang diterima oleh *requester* adalah 3, sehingga skor *success rate* dari worker adalah $3/5$ sama dengan $0{,}6$.
   - $relationship$ adalah adanya relasi antara *requester* dengan *worker* maka skor *relationship* sama dengan 1 dan jika tidak ada relasi maka skor *relationship* sama dengan 0.
     Contoh skenario: *Requester* memiliki relasi dengan *worker* maka skor *relationship* sama dengan 1.

$$reputation = \frac{success\ rate + relationship}{2} = \frac{0{,}6 + 1}{2} = \frac{1{,}6}{2} = 0{,}8$$

   b. Setelah diperoleh skor *reputation*, *credentials*, dan experiences, maka nilai dapat dihitung nilai *ConfidenceTrustworthiness* sebagai berikut:

$$ConfidenceTrustworthiness = \left(\left(\frac{0.8}{3} + \frac{1}{3} + \frac{0.8}{3}\right) \times 100\%\right)$$
$$= (0{,}27 + 0{,}33 + 0{,}27) = 0{,}87 \times 100\%$$
$$= 87\%$$

2.3 Analisis *Top N Recommendation*

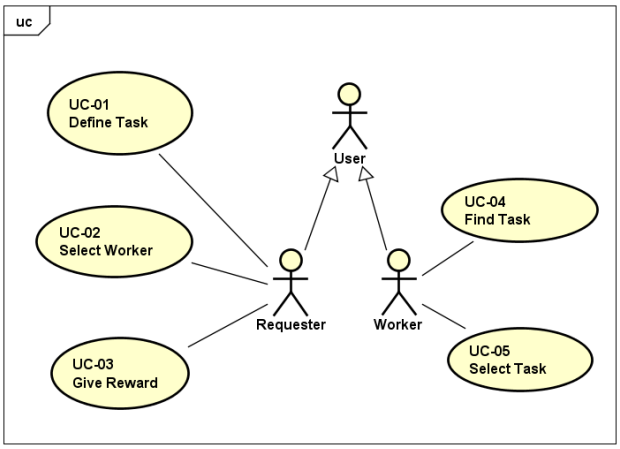
Langkah-langkah dari analisis *top-n task recommendation* dijelaskan sebagai berikut:

1) Skor yang sudah diperoleh dari hasil proses *task matching* kemudian dikombinasikan dengan skor yang diperoleh dari perhitungan *social profile computation* dengan bobot masing-masing sebesar 50% dan 50%.
2) Hasil skor dari kombinasi antara proses *task matching* dengan perhitungan *social profile computation*, dilakukan perangkingan yang diurutkan berdasarkan 10 *task* dengan perolehan skor *confidence matching* ditambah *confidence trustworthiness* tertinggi.

3. PERANCANGAN APLIKASI

3.1 Rancangan Kebutuhan Fungsional

Kebutuhan fungsional dari aplikasi Cromosom dijelaskan pada "Gambar 2".



Gambar 2. Diagram Use Case Aplikasi Cromosom

Penjelasan dari masing-masing *use case* dijabarkan pada "Tabel 4".

Tabel 4. Deskripsi Diagram Use Case

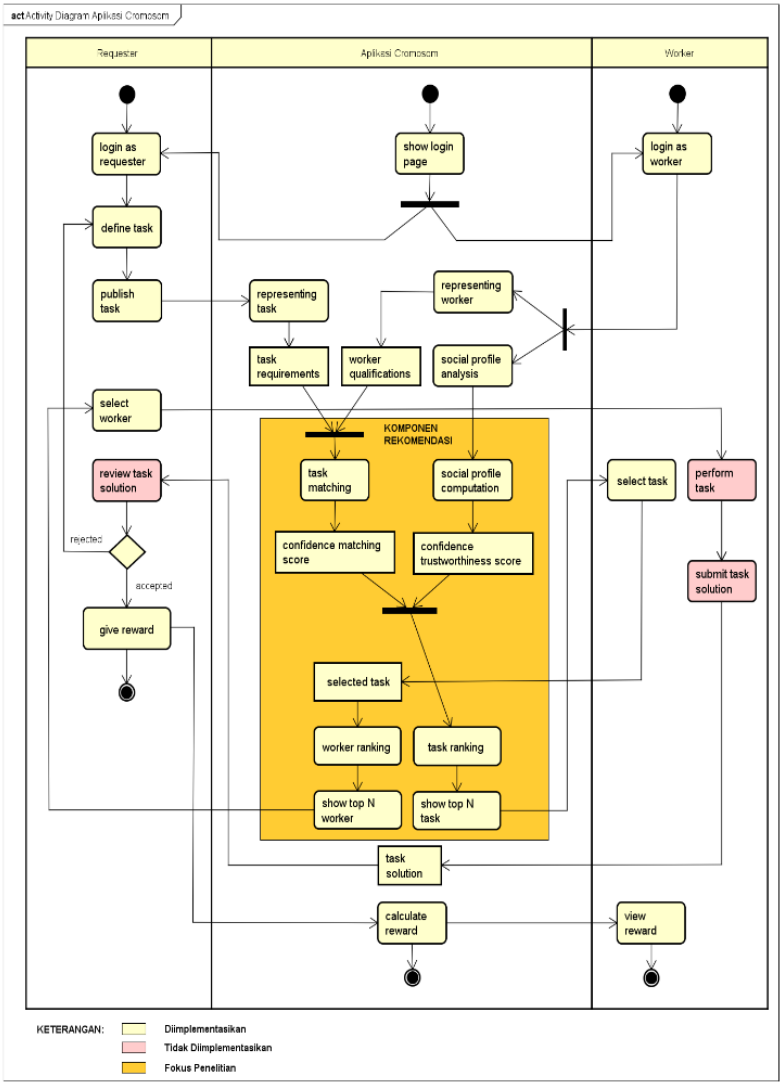
No. Use Case	Nama Use Case	Deskripsi
UC-01	*Define Task*	Requester mampu mendefinisikan task yang akan di-outsource dengan mendeskripsikan informasi dari task tersebut, seperti title, description, duration, programming language, frameworks, skills, location, dan rewards. Kemudian requester mempublikasikan task yang sudah didefinisikan tersebut ke aplikasi Cromosom
UC-02	*Select Worker*	Requester mampu memilih worker yang akan mengerjakan task sesuai dengan hasil perangkingan aplikasi Cromosom.
UC-03	*Give Reward*	Requester mampu memberikan reward kepada worker jika hasil pengerjaan task diterima.
UC-04	*Find Task*	Worker mampu mencari *task* yang sesuai

No. Use Case	Nama Use Case	Deskripsi
		dengan keahliannya
UC-05	*Select Task*	Worker mampu memilih task yang direkomendasikan oleh aplikasi Cromosom.

3.2 Rancangan Alur Kerja Aplikasi Cromosom

Alur kerja pada aplikasi Cromosom diilustrasikan menggunakan *activity diagram* seperti pada "Gambar 3" dengan langkah-langkah sebagai berikut:

1) *Requester* mendefinisikan *task* yang akan dikerjakan oleh *worker*, kemudian *task* tersebut dipublikasikan, dan aplikasi akan menyimpan *task* tersebut pada *database*.
2) *Task* yang sudah disimpan pada *database* direpresentasikan berdasarkan atribut yang hanya dibutuhkan untuk proses *task matching*, yaitu *task requirements*, diantaranya *programming language*, *framework*, *skills*, dan *location*.
3) *Worker* dapat melakukan *login* pada aplikasi Cromosom secara langsung dengan memasukkan *username* dan *password* atau menggunakan otorisasi media sosial Github.
4) Profil dari *worker* direpresentasikan berdasarkan atribut yang hanya dibutuhkan untuk untuk proses *task matching*, yaitu *worker qualifications*.
5) Pada proses *task matching*, dilakukan perhitungan untuk mencari kedekatan nilai atau kemiripan antara *task requirements* dengan *worker qualifications*.
6) Pada proses *social profile computation*, dilakukan perhitungan dengan menjumlahkan nilai *reputation*, *credentials*, dan *experiences*. Hasil perhitungan tersebut akan menghasilkan nilai *confidence trustworthiness*.
7) Nilai *confidence matching* dengan nilai *confidence trustworthiness* digabungkan kemudian dilakukan perangkingan sehingga menghasilkan top 10 *task* dengan nilai tertinggi yang akan direkomendasikan kepada *worker*.
8) Hasil perhitungan dari proses *task matching* akan menghasilkan nilai *condidence matching*.
9) *Worker* memilih salah satu *task* yang direkomendasikan tersebut, kemudian *requester* mendapat notifikasi bahwa *task* sudah dipilih oleh beberapa orang *worker*, kemudian *worker* dirangking berdasarkan nilai *trustworthiness*.
10) *Worker* dengan perolehan skor tertinggi dapat diterima oleh *requester* untuk mengerjakan *task*.
11) Worker yang diterima oleh *requester*, dapat mengerjakan *task* kemudian mengirim hasil pengerjaan *task* yang diunggah pada repositori Github.
12) *Requester* dapat mengunduh hasil pengerjaan *task* kemudian memeriksa hasilnya, jika hasil pengerjaan *task* sesuai dengan kebutuhan *requester*, maka *worker* akan diberikan *reward* dalam bentuk poin



Gambar 3. Diagram Aktivitas Aplikasi Cromosom

3.3 Rancangan Komponen Sistem Rekomendasi

3.3.1 Komponen *Task Matching*

Proses *task matching* dimulai dari pemisahan atribut kebutuhan *task* menjadi atribut bertipe data diskrit dan kontinu. Atribut dengan tipe data diskrit di dalam proses *task matching* akan menghasilkan nilai 0 atau 1 (*true* atau *false*). Sedangkan atribut dengan tipe data continue akan menghasilkan nilai antara 0 dan 1 (*fuzzy*). Atribut dengan tipe data diskrit diantaranya *programming language*, *framework*, dan *skill*, sedangkan atribut dengan tipe data kontinu adalah *location*.

Pada "Tabel 5" dijelaskan contoh proses *task matching* antara keahlian yang dibutuhkan untuk mengerjakan *task* dengan kualifikasi *worker*.

Tabel 5. Perhitungan Proses Task Matching

Parameter	Tipe	Task Requirements	Worker Qualifications	Hasil Matching	Bobot	Perolehan Skor
Programming Language	Diskrit	Java	Java	1	30%	0,3
Frameworks	Diskrit	Spring	Struts	0	40%	0
Skills	Diskrit	Undefined	-	-	-	-

Parameter	Tipe	Task Requirements	Worker Qualifications	Hasil Matching	Bobot	Perolehan Skor
Locations	Kontinu	Radius = 100 KM	Radius = 121 KM	0,826	30%	0,247
Confidence Matching						0,547

Confidence matching adalah nilai yang didapatkan dari penggabungan antara *programming language*, *framework*, *skills*, dan *locations*. Berdasarkan proses dari Task Matching pada "Tabel 5" diperoleh nilai *confidence matching* sebesar 0,547.

3.3.2 Komponen *Social Profile Computation*

Berdasarkan hasil dari analisis *social profile worker* pada media sosial, nilai *trustworthiness* dari seorang *user* terhadap *user* lainnya dapat dipertimbangkan berdasarkan skor *reputation*, skor *credentials*, dan skor *experiences*. Skor *reputation* dihitung dari atribut *success rate* dan *relationship* yang terjalin antar *user*. Skor *credentials* diperoleh dengan melihat sertifikat yang dimiliki oleh *worker* berkaitan dengan bahasa pemograman tertentu, namun pada repositori Github memiliki keterbatasan bahwa tidak adanya sertifikat keahlian *worker* berkaitan dengan bahasa pemograman. Akan tetapi, setiap repositori proyek yang pernah dikerjakan oleh *worker* menyimpan informasi bahasa pemograman, sehingga dapat dimanfaatkan untuk perhitungan skor *credentials*. Pada penelitian ini, skor *credentials* diperoleh dengan melakukan pengecekan terhadap repositori proyek berkaitan dengan bahasa pemograman apakah memiliki kesamaan kualifikasi yang dideskripsikan oleh *worker* pada aplikasi Cromosom. Nilai *experiences* dihitung berdasarkan jumlah *commit* yang dilakukan oleh *worker* pada repositori proyek.

Pada "Tabel 6" dijelaskan contoh hasil perhitungan dari *confidence trustworthiness*.

Tabel 6. Hasil Perhitungan Trustworthiness

Parameter		Kondisional	Nilai Aktual	Perhitungan	Perolehan Skor
Reputation	Relationship (R)	*if arrayRelationship* > 0 ? *return* 1 *else* 0	1	R + SR = $= \frac{1 + 0,8}{2} = 0,9$	0,3
	Success Rate (SR)	$\frac{jumlah\ task\ yang\ diterima}{jumlah\ task\ yang\ dikerjakan}$	$\frac{4}{5} = 0,8$	$= \frac{0,9}{3} = 0,3$	
Credentials		*if taskRequirement.language == repoDetail.language ?* return 1 else 0	1	$\frac{1}{3} = 0,33$	0,33
Experiences		$\frac{worker\ commit}{\max commit}$	max commit = 100 worker commit = 75	$\frac{75}{100} = 0,75$ $\frac{0,75}{3} = 0,25$	0,25
Confidence Trustworthiness					0,88

Confidence trustworthiness adalah skor yang didapatkan dari penggabungan antara skor *reputations*, skor *credentials*, dan skor *experiences*. Berdasarkan proses dari *Social Profile Computation* pada "Tabel 6" diperoleh nilai *confidence trustwortiness* sebesar 0,88. Berdasarkan contoh hasil perolehan

skor *confidence matching* pada "Tabel 5" dan hasil perolehan skor *confidence trustworthiness* pada Tabel 6, maka dapat dilakukan penggabungan skor dengan skenario pembobotan masing-masing sebesar 50% dan 50%, dijelaskan pada "Tabel 7".

Tabel 7. Kombinasi Skor Confidence Matching dengan Confidence Trustworthiness

Parameter	Skor	Bobot	Perolehan Skor	Skor Akhir Confidence
Confidence Matching	0,547	50%	0,2735	0,2735 + 0,44 = 0,677
Confidence Trustworthiness	0,88	50%	0,44	

4. HASIL DAN PEMBAHASAN

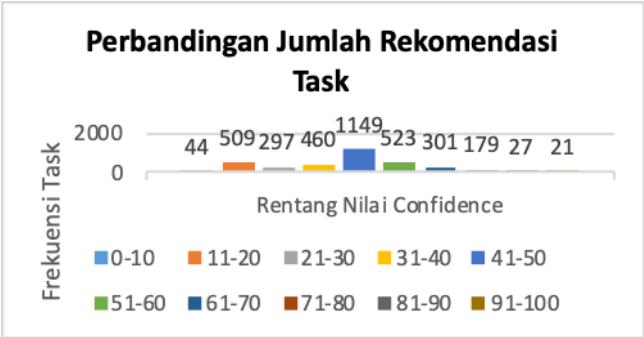
4.1 Hasil Rekomendasi Task Menggunakan Distribusi Normal

Representasi numerik dari nilai *confidence* yang dibagi menjadi 10 skala dengan rentang nilai antara 0 sampai 100%. Untuk setiap rentang nilai dilakukan perhitungan nilai frekuensinya, dimana yang menjadi nilai frekuensi adalah jumlah *task* yang direkomendasikan. Ada 351 *worker* yang memperoleh rekomendasi *task* masing-masing 10 rekomendasi, sehingga total *task* yang direkomendasikan kepada 351 *worker* adalah 3.510 *task*. Pada "Tabel 8" diuraikan distribusi data dari rekomendasi *task* sesuai dengan rentang nilai *confidence*.

Tabel 8. Frekuensi Pembagian Nilai Akhir Confidence

Rentang Nilai Confidence (x)	Frekuensi Task Yang Direkomendasikan (f)	Frekuensi Komulatif (fk)
0-10	44	44
11-20	509	553
21-30	297	850
31-40	460	1.310
41-50	1149	2.459
51-60	523	2.982
61-70	301	3.283
71-80	179	3.462
81-90	27	3.489
91-100	21	3.510

Untuk memberikan gambaran apakah data dari *task* yang direkomendasikan berdistribusi normal, maka dapat diilustrasikan menggunakan grafik seperti pada "Gambar 4".



Gambar 4. Grafik Distribusi Normal Nilai AKhir Confidence

Dari “Gambar 4” dapat dilihat bahwa rentang persentil nilai akhir *confidence* dengan rentang nilai antara 41-50 memperoleh frekuensi tertinggi yaitu sebesar 1.149 *task* yang direkomendasikan. Hal tersebut karena dari hasil perhitungan *task matching* dan *social profile computation* menghasilkan rekomendasi *task* yang lebih banyak dengan rentang nilai *confidence* antara 41%-50%.

Dapat diamati bahwa “Gambar” sudah mendekati distribusi normal yang membentuk “bell curve”, namun ada keganjilan yang terjadi pada rentang nilai *confidence* antara 11-20, dimana seharusnya jumlah *task* yang direkomendasikan adalah lebih tinggi dari rentang nilai *confidence* antara 0-10 dan lebih rendah dari rentang nilai *confidence* antara 21-30.

Hal tersebut terjadi karena perhitungan *task matching* lebih dominan dalam memberikan nilai untuk menghasilkan rekomendasi *task*. Perhitungan *task matching* diperoleh dengan mencocokkan antara *task requirements* dengan *worker qualifications* dimana *worker* pada *platform* Cromosom ketika melakukan registrasi dapat mendeskripsikan keahliannya terlepas dari data keahlian tersebut benar atau tidak, sehingga perhitungan *task matching* lebih dominan untuk mendapatkan nilai *confidence*.

Sedangkan perhitungan *social profile computation* yang menghasilkan nilai *trustworthiness* dari *worker* belum memberikan dampak dari hasil perhitungan dari seluruh *task* yang direkomendasikan karena (1) perhitungan nilai *reputation* yaitu belum adanya *relationship* antara *requester* dan *worker*, setiap *task* yang dipublikasikan oleh *requester* belum dikerjakan oleh semua *worker* sehingga *worker* tidak mendapatkan *success rate*, (2) perhitungan nilai *credentials* yaitu banyak *worker* yang mendeskripsikan kehalian palsu pada *platform* Cromosom, dan (3) perhitungan nilai *experiences* yaitu dilihat dari jumlah *commit* dari *worker* terhadap proyek masih sedikit sehingga tidak mendapatkan nilai *experiences* yang maksimal.

4.2 Hasil Kualitas Rekomendasi Task Menggunakan Persentil

Untuk menghitung persentil, secara matematis dapat ditulis menggunakan Persamaan.

$$P_i = T_b + \left\{ \frac{\frac{i}{100}n - f_k}{f_i} \right\} p$$

Dimana,

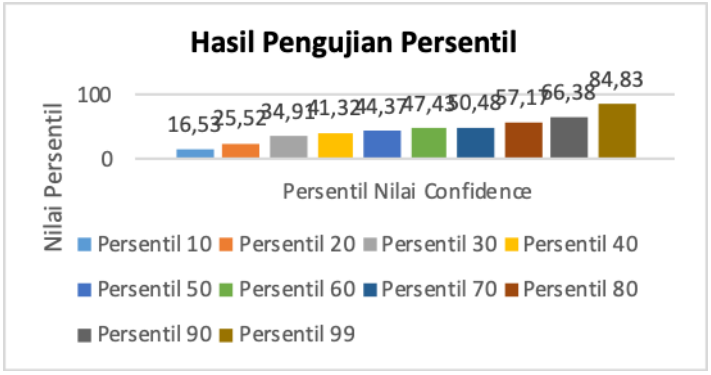
P_i adalah persentil ke-i, dimana i adalah bilangan bulat yang kurang dari 100 ($1..99$)
T_b adalah tepi bawah kelas persentil
n adalah jumlah seluruh frekuensi
fk adalah jumlah frekuensi sebelum kelas persentil
f_i adalah frekuensi kelas persentil
p adalah panjang kelas interval

Dilakukan pengujian untuk mencari nilai *confidence* pada persentil ke-10, persentil ke-20, persentil ke-30, persentil ke-40, persentil, ke-50, persentil ke-60, persentil ke-70, persentil ke-80, dan persentil ke-90. Hasil perhitungan persentil nilai *confidence* diuraikan pada “Tabel 9”

Tabel 9. Persentil Nilai Akhir Confidence

Rangking Persentil	Nilai Persentil
Persentil 10	16,53
Persentil 20	25,52
Persentil 30	34,91
Persentil 40	41,32
Persentil 50	44,37
Persentil 60	47,43
Persentil 70	50,48
Persentil 80	57,17
Persentil 90	66,38
Persentil 99	84,83

Untuk memberikan gambaran dari hasil perhitungan persentil, maka maka dapat diilustrasikan menggunakan grafik seperti pada "Gambar 5".



Gambar 5. Hasil Persentil Nilai Confidence

Berdasarkan "Gambar 5" dapat dilihat nilai persentil untuk setiap nilai *confidence* dimana skala persen nilai *confidence* yaitu $0 < persentil < 100$. Dari hasil pengujian dapat dilihat bahwa nilai persentil semakin naik jika persentil nilai *confidence* mendekati 100. Dari beberapa studi literatur yang dilakukan bahwa seharusnya kurva yang dibentuk dari persentil nilainya akan terus menaik dari kiri ke kanan. Dapat dilihat nilai persentil dari data rekomendasi *task* menaik dari 0 menuju 100, sehingga dapat disimpulkan kualitas *task* yang direkomendasikan cukup baik.

4. KESIMPULAN

4.1 Kesimpulan

Kesimpulan dari penelitian ini adalah sebagai berikut:

1. Berdasarkan fungsionalitasnya, aplikasi Cromosom mampu merekomendasikan *task* kepada *worker* yang sesuai dengan keahliannya.
2. Proses *Social Profile Computation* mampu menghasilkan nilai *trustworthiness* dari worker sehingga dapat dilakukan perangkingan terhadap *worker* berdasarkan nilai *trustworthiness*, dan menjadi dasar untuk *requester* dalam memilih *worker* yang tepat dalam mengerjakan *task*.
3. Berdasarkan hasil pengujian dari distribusi *data* rekomendasi *task* bahwa sudah mendekati grafik distribusi normal namun untuk rentang nilai antara 10-20 memiliki simpangan karena faktor perhitungan *task matching* lebih dominan dalam menghasilkan nilai *confidence*, dan dari hasil pengujian kualitas data rekomendasi task menghasilkan kualitas yang cukup baik.

4.2 Saran

Saran untuk penelitian selanjutnya adalah:

1. Pada penelitian ini memiliki keterbatasan dalam analisis konten, dimana analisis konten nantinya dapat dimanfaatkan untuk menganalisis kredensial *worker* dalam bentuk dokumen sertifikat.
2. Untuk melakukan proses *task matching* nantinya diharapkan dapat dintegrasikan dengan teknik klasifikasi, misalnya naïve bayes dengan tujuan untuk mempercepat waktu proses pencocokan antara *task requirement* dengan *worker qualifications*.
3. Semua pendekatan *task matching* dapat diimplementasikan pada aplikasi Cromosom, sehingga dapat dilakukan perbandingan pendekatan *task matching* mana yang menghasilkan rekomendasi *task* yang paling baik.
4. Saat ini, *relationship* antar user hanya dimanfaatkan untuk menunjukkan adanya relasi antar *user* untuk perhitungan *trustworthiness*. Saran untuk penelitian selanjutnya adalah relationship antar user dapat dimodelkan menjadi sebuah graf, sehingga dapat dimanfaatkan untuk membangun *social trust* antar user.

DAFTAR PUSTAKA

[1] J. Howe, "The Rise of Crowdsourcing," *Wired Mag.*, vol. 14, no. 06, hal. 1–5, 2006.

[2] J. Howe, "Crowdsourcing: Why the Power of The Crowd is Diving the Future Business," *Crown Bus.*, 2008.

[3] M. Allahbakhsh, B. Benatallah, A. Ignjatovic, H. R. Motahari-Nezhad, E. Bertino, dan S. Dustdar, "Quality control in crowdsourcing systems: Issues and directions," *IEEE Internet Comput.*, vol. 17, no. 2, hal. 76–81, 2013.

[4] K. Mao, L. Capra, M. Harman, dan Y. Jia, "A survey of the use of crowdsourcing in software engineering," *J. Syst. Softw.*, vol. 126, hal. 57–84, 2017.

[5] A. Kass, "Trustworthiness in Enterprise Crowdsourcing : a Taxonomy & evidence from data," 2016.

[6] K. Mao, Y. Yang, Q. Wang, Y. Jia, dan M. Harman, "Developer recommendation for crowdsourced software development tasks," *Proc. - 9th IEEE Int. Symp. Serv. Syst. Eng. IEEE SOSE 2015*, vol. 30, hal. 347–356, 2015.

[7] M. C. Yuen, I. King, dan K. S. Leung, "Task matching in crowdsourcing," *Proc. - 2011 IEEE Int. Conf. Internet Things Cyber, Phys. Soc. Comput. iThings/CPSCom 2011*, no. October, hal. 409–412, 2011.

[8] M. Y. I. King, "TaskRec : A Task Recommendation Framework in Crowdsourcing Systems TaskRec : A Task Recommendation Framework in Crowdsourcing Systems," no. June 2015, 2014.

[9] S. Marjanovic, C. Fry, dan J. Chataway, "Crowdsourcing based business models: In search of evidence for innovation 2.0," *Sci. Public Policy*, vol. 39, no. 3, hal. 318–332, 2012.

[10] A. Sari, A. Tosun, dan G. I. Alptekin, "A Systematic Literature Review on Crowdsourcing in Software Engineering." The Journal of Systems & Software, hal. 26, 2018.

[11] T. D. Latoza dan A. Van der Hoek, "Crowdsourcing in Software Engineering," *IEEE Softw.*, 2016.

[12] K. J. Stol, T. D. Latoza, dan C. Bird, "Crowdsourcing for Software Engineering," *IEEE Softw.*, vol. 34, no. 2, hal. 30–36, 2017.

[13] M. Hosseini, A. Shahri, K. Phalp, J. Taylor, R. Ali, dan F. Dalpiaz, "Configuring Crowdsourcing for Requirements Elicitation," 2015.

[14] T. D. Latoza, M. Chen, L. Jiang, M. Zhao, dan A. Van Der Hoek, "Borrowing from the

Crowd : A Study of Recombination in Software Design Competitions," no. c, hal. 551–562, 2015.

[15] A. van der H. Thomas D. LaToza1, W. Ben Towne2, Christian M. Adriano1, "Microtask Programming: Building Software with a Crowd," 2014.