

**CCGtown: Alat Anotasi Combinatory  
Categorial Grammar (CCG) Semi-otomatis  
Berbasis Web**

**Proposal Tugas Akhir**

**Kelas TA NLP**

**Wisnu Adi Nurcahyo**

**NIM: 1301160479**



**Program Studi Sarjana Informatika**

**Fakultas Informatika**

**Universitas Telkom**

**Bandung**

**2020**

## Lembar Persetujuan

**CCGtown: Alat Anotasi Combinatory Categorical Grammar  
(CCG) Semi-otomatis Berbasis Web**

***CCGtown: A Web-based Semi-automatic Combinatory  
Categorical Grammar (CCG) Annotation Tool***

**Wisnu Adi Nurcahyo  
NIM: 1301160479**

Proposal ini diajukan sebagai usulan pembuatan tugas akhir pada  
Program Studi Sarjana Informatika  
Fakultas Informatika Universitas Telkom

Bandung, 13 November 2020  
Menyetujui

Calon Pembimbing 1

Calon Pembimbing 2

Dr. Ade Romadhony, S.T., M.T.  
NIP: 06840042

Said Al Faraby, S.T., M.Sc.  
NIP: 15890019

# Abstrak

TBA.

**Kata Kunci:** natural language processing, combinatory categorial grammar, annotation tool

# Daftar Isi

|   |           |
|---|-----------|
| <b>Abstrak</b>                                | <b>i</b>  |
| <b>Daftar Isi</b>                             | <b>ii</b> |
| <b>I Pendahuluan</b>                          | <b>1</b>  |
| 1.1 Latar Belakang . . . . .                  | 1         |
| 1.2 Perumusan Masalah . . . . .               | 2         |
| 1.3 Tujuan . . . . .                          | 2         |
| 1.4 Rencana Kegiatan . . . . .                | 2         |
| 1.5 Jadwal Kegiatan . . . . .                 | 2         |
| <b>II Kajian Pustaka</b>                      | <b>4</b>  |
| 2.1 Categorical Grammar . . . . .             | 4         |
| 2.2 Combinatory Categorical Grammar . . . . . | 5         |
| 2.3 Lambda Calculus . . . . .                 | 6         |
| 2.4 CCGweb . . . . .                          | 7         |
| <b>III Perancangan Sistem</b>                 | <b>8</b>  |
| 3.1 Desain Database . . . . .                 | 8         |
| 3.2 Desain Sistem . . . . .                   | 10        |
| 3.3 Pertimbangan UI/UX . . . . .              | 15        |
| <b>Daftar Pustaka</b>                         | <b>16</b> |
| <b>Lampiran</b>                               | <b>17</b> |

# Bab I

## Pendahuluan

### 1.1 Latar Belakang

Riset pemrosesan bahasa alami untuk bahasa Indonesia saat ini masih terbilang sedikit. Bahkan, masih banyak area riset yang belum tersentuh seperti contohnya *combinatory categorial grammar* (CCG). Sementara itu, riset mengenai CCG untuk bahasa Inggris sudah cukup matang. Adapun untuk bahasa lainnya (seperti bahasa Vietnam) sudah mulai menggunakan CCG di dalam penelitiannya [5]. Agar dapat menerapkan CCG di dalam aplikasi yang dibangun, *tools* seperti CCG *parser* dan CCG *supertagger* harus tersedia terlebih dahulu. Masing-masing dari *tools* tersebut memerlukan *dataset* agar dapat memberikan hasil yang akurat.

Umumnya terdapat dua cara yang paling sering digunakan untuk mengembangkan CCG *supertagger* maupun CCG *parser* bahasa lokal yaitu (1) membangun *dataset* CCG *supertag* secara manual maupun semi-otomatis atau (2) melakukan transfer *dataset* dari CCGbank (atau dari sumber lainnya) ke dalam bahasa lokal dengan cara melakukan alih bahasa dan bila perlu melakukan penyesuaian untuk *supertag*-nya [2]. Proses pembangunan *dataset* umumnya menggunakan bantuan *annotation tool* agar proses anotasinya menjadi lebih mudah. Salah satu *annotation tool* yang dapat digunakan adalah CCGweb [1].

Tugas akhir dengan judul “CCGtown: Alat Anotasi Combinatory Categorical Grammar (CCG) Semi-otomatis Berbasis Web” berusaha untuk membangun alat anotasi CCG baru dengan UI/UX yang lebih baik dari CCGweb. Selain itu, dengan bantuan NLTK alat anotasi ini dapat melakukan *generate* untuk CCG *derivation*-nya kemudian pengguna juga dapat mengubah *derivation*-nya apabila diperlukan. Tujuan dari dibangunnya alat anotasi CCG ini adalah untuk mempermudah proses anotasi yang repetitif. Selanjutnya, *dataset* CCG pertama untuk bahasa Indonesia diharapkan dapat dipublikasikan.

## 1.2 Perumusan Masalah

Rumusan masalah yang akan diangkat yaitu:

1. Bagaimana proses pembangunan alat anotasi untuk CCG?
2. Apakah CCGtown dapat digunakan dengan mudah?

## 1.3 Tujuan

Tujuan yang diharapkan dapat tercapai oleh tugas akhir ini yaitu:

1. Membangun dan merilis alat anotasi CCG semi-otomatis.
2. Dapat digunakan untuk membangun *dataset* CCG.

## 1.4 Rencana Kegiatan

Rencana kegiatan yang akan dilakukan adalah sebagai berikut:

- Studi literatur
- Studi *tools* yang tersedia
- Studi bahasa pemrograman yang akan digunakan
- Perancangan sistem *annotation tool* CCGtown
- Membangun *annotation tool* CCGtown
- Menguji kemudahan dalam menggunakan CCGtown

## 1.5 Jadwal Kegiatan

Laporan proposal ini akan dijadwalkan sesuai dengan tabel 1.1.

Tabel 1.1: Jadwal kegiatan proposal tugas akhir.

| No | Kegiatan                                   | Bulan ke- |  |  |  |   |  |  |  |   |  |  |  |   |  |  |  |   |  |  |  |   |  |  |  |
|----|--|-----------|--|--|--|---|--|--|--|---|--|--|--|---|--|--|--|---|--|--|--|---|--|--|--|
|    |  | 1         |  |  |  | 2 |  |  |  | 3 |  |  |  | 4 |  |  |  | 5 |  |  |  | 6 |  |  |  |
| 1  | Studi Literatur                            |           |  |  |  |   |  |  |  |   |  |  |  |   |  |  |  |   |  |  |  |   |  |  |  |
| 2  | Studi <i>Tools</i> yang Tersedia           |           |  |  |  |   |  |  |  |   |  |  |  |   |  |  |  |   |  |  |  |   |  |  |  |
| 3  | Studi Bahasa Pemrograman dan Framework-nya |           |  |  |  |   |  |  |  |   |  |  |  |   |  |  |  |   |  |  |  |   |  |  |  |
| 5  | Analisis dan Perancangan Sistem            |           |  |  |  |   |  |  |  |   |  |  |  |   |  |  |  |   |  |  |  |   |  |  |  |
| 6  | Implementasi Sistem                        |           |  |  |  |   |  |  |  |   |  |  |  |   |  |  |  |   |  |  |  |   |  |  |  |
| 7  | Analisa Hasil Implementasi                 |           |  |  |  |   |  |  |  |   |  |  |  |   |  |  |  |   |  |  |  |   |  |  |  |
| 8  | Penulisan Laporan                          |           |  |  |  |   |  |  |  |   |  |  |  |   |  |  |  |   |  |  |  |   |  |  |  |

## Bab II

### Kajian Pustaka

#### 2.1 Categorical Grammar

Categorical Grammar (CG) merupakan sebuah istilah yang mencakup beberapa formalisme terkait yang diajukan untuk sintaks dan semantik dari bahasa alami serta untuk bahasa logis dan matematis [7]. Karakteristik yang paling terlihat dari CG adalah bentuk ekstrim dari leksikalismenya di mana beban utama (atau bahkan seluruh beban) sintaksisnya ditanggung oleh leksikon. Konstituen tata bahasa dalam *categorical grammar* dan khususnya semua leksikal diasosiasikan dengan suatu *type* atau “*category*” (dalam *category theory*) yang mendefinisikan potensi mereka untuk dikombinasikan dengan konstituen lain untuk menghasilkan konstituen majemuk. *Category* tersebut adalah salah satu dari sejumlah kecil *category* dasar (seperti NP) atau *functor* (dalam *category theory*). Dalam hal ini, *category* dapat diartikan sebagai *syntactic type* dari suatu kata.

Secara formal, *syntactic type* didefinisikan sebagai himpunan bagian dari suatu *semigroup*  $M$  yang tunduk pada tiga operasi yaitu 2.1, 2.2, dan 2.3 dimana  $A$ ,  $B$ , dan  $C$  merupakan himpunan bagian dari  $M$  [3]. Adapun  $A \cdot B$  dibaca  $A$  times  $B$ ,  $C/B$  dibaca  $C$  over  $B$ , dan  $A \setminus C$  dibaca  $A$  under  $C$ . Selanjutnya, dapat dilihat bahwasannya untuk semua  $A, B, C \subseteq M$  sehingga kita dapatkan 2.4 dan 2.5. Terakhir, persamaan 2.6 dapat diabaikan apabila dihadapkan dengan *multiplicative system* yang tidak asosiatif. Sementara itu, apabila *semigroup*-nya merupakan sebuah *monoid* dengan identitas 1 maka kita dapatkan 2.7 dimana  $I = \{1\}$ .

$$A \cdot B = \{x \cdot y \in M \mid x \in A \wedge y \in B\} \quad (2.1)$$

$$C/B = \{x \in M \mid \forall_{y \in B} x \cdot y \in C\} \quad (2.2)$$

$$A \setminus C = \{y \in M \mid \forall_{x \in A} x \cdot y \in C\} \quad (2.3)$$

$$A \cdot B \subseteq C \quad \text{jika dan hanya jika} \quad A \subseteq C/B \quad (2.4)$$

$$A \cdot B \subseteq C \quad \text{jika dan hanya jika} \quad B \subseteq A \setminus C \quad (2.5)$$



Pamungkas  $\vdash$  NP : *pamungkas'*  
 Setyo  $\vdash$  NP : *setyo'*  
 dan  $\vdash$  CONJ :  $\lambda x.\lambda y.\lambda f. (f\ x) \wedge (f\ y)$   
 menyukai  $\vdash$  (S\NP)/NP :  $\lambda x.\lambda y. suka(y, x)$   
 rendang  $\vdash$  NP : *rendang'*

Gambar 2.1: Kamus yang memetakan token kata ke bentuk CCG *lexicon*-nya.

$$(A \cdot B) \cdot C = A \cdot (B \cdot C) \quad (2.6)$$

$$I \cdot A = A = A \cdot I \quad (2.7)$$

Ada beberapa notasi berbeda untuk *category* dalam merepresentasikan *directional*-nya. Notasi yang paling umum digunakan adalah “*slash notation*” yang dipelopori oleh Bar-Hilel, Lambek, dan kemudian dimodifikasi dalam kelompok teori yang dibedakan sebagai tata bahasa “*combinatory*” *categorial grammar* (CCG). Sebagai contoh, *category* (S\NP)/NP merupakan suatu *functor* yang memiliki dua buah notasi *slash* yaitu \ dan /. Masing-masing notasi *slash* tersebut merepresentasikan *directionality* yang berbeda. Notasi *forward slash*, /, mengindikasikan bahwa argumen dari suatu *functor* X/Y ada di bagian kanan atau dengan kata lain Y. Adapun *backward slash*, \, mengindikasikan bahwa argumen dari suatu *functor* X\Y ada di bagian kiri atau dengan kata lain X. Demikian itu, penggunaan notasi *slash* yang tepat sangat penting dikarenakan hal ini dapat mempengaruhi konstituen dari hasil “kombinasi” *category*-nya.

## 2.2 Combinatory Categorial Grammar

Combinatory Categorial Grammar (CCG) merupakan salah satu formalisme tata bahasa yang gaya aturannya diturunkan dari *categorial grammar* dengan beberapa penambahan aturan dan istilah baru [8]. Di CCG, *category* dapat dipasangkan dengan *semantic representation*. Dalam hal ini, *semantic representation* yang dimaksud adalah abstraksi fungsi lambda (dalam *lambda calculus*, *lambda function*). Sebagai contoh, *category* (S\NP)/NP dapat dipasangkan dengan fungsi lambda  $\lambda x.f x$  sehingga dapat ditulis menjadi (S\NP)/NP :  $\lambda x.f x$ . Adapun pemetaan dari suatu token kata ke *category*-nya menggunakan notasi  $\vdash$ . Sebagai contoh, anggap saja kita memiliki kamus pemetaan seperti pada Gambar 2.1. Apabila kita memiliki kalimat “Pamungkas dan Setyo menyukai rendang”, maka kita dapatkan:

|                |   |            |                                     |              |
|----------------|---|------------|-------------------------------------|--------------|
| Pamungkas      | dan   | Setyo      | menyukai                            | rendang      |
| NP             | CONJ  | NP         | (S\NP)/NP                           | NP           |
| : $pamungkas'$ | : $\lambda x.\lambda y.\lambda f. (f\ x) \wedge (f\ y)$ | : $setyo'$ | : $\lambda x.\lambda y. suka(y, x)$ | : $rendang'$ |

Ada beberapa operasi yang dapat dilakukan dalam CCG. *Operand* dari operasi yang dimaksud adalah *category*. Berdasarkan contoh di atas, akan ada tiga operasi yang dijalankan yaitu *coordination*, *forward application*, dan *type rising*. Untuk mendapatkan hasil yang diinginkan, kita lakukan *type rising* sebelum *forward application* di akhir. Sehingga, kita dapatkan:

|   |   |            |  |              |
|---|---|------------|--|--------------|
| Pamungkas   | dan   | Setyo      | menyukai   | rendang      |
| NP  | CONJ  | NP         | (S\NP)/NP  | NP           |
| : $pamungkas'$                                    | : $\lambda x.\lambda y.\lambda f. (f\ x) \wedge (f\ y)$ | : $setyo'$ | : $\lambda x.\lambda y. suka(y, x)$                          | : $rendang'$ |
| <hr/>   |   |            | < & >  | >            |
| NP  |   |            | S\NP   |              |
| : $\lambda f. (f\ pamungkas') \wedge (f\ setyo')$ |   |            | : $\lambda y. suka(y, rendang')$                             |              |
| <hr/>   |   |            | > T  |              |
| S/(S\NP)  |   |            |  |              |
| : $\lambda f. (f\ pamungkas') \wedge (f\ setyo')$ |   |            |  |              |
| <hr/>   |   |            | >  |              |
| S   |   |            |  |              |
|   |   |            | : $suka(pamungkas', rendang') \wedge suka(setyo', rendang')$ |              |

Berdasarkan hasil evaluasi tersebut, kita dapatkan *query* 2.8 yang diperoleh dari kalimat “Pamungkas dan Setyo menyukai rendang”. Demikian itu, komputer dapat melakukan komputasi berdasarkan *query* yang telah diperoleh. Kegiatan tersebut merupakan apa yang disebut dengan CCG *parsing*. Untuk dapat melakukan parsing, CCG *lexicon* diperlukan. Untuk mendapatkan CCG *lexicon* kita dapat menggunakan CCG *supertagger* yang akan melakukan pelabelan suatu token kata ke CCG *lexicon* berdasarkan pemetaannya.

$$suka(pamungkas', rendang') \wedge suka(setyo', rendang') \quad (2.8)$$

## 2.3 Lambda Calculus

*Lambda calculus* ( $\lambda$ -*calculus*) merupakan sebuah formalisme yang dikembangkan oleh Alonzo Church sebagai alat yang digunakan untuk memahami konsep komputasi yang efektif [6]. Formalisme  $\lambda$ -*calculus* cukup populer dan bahkan dijadikan sebagai pondasi teori bagi paradigma pemrograman *functional programming*. Konsep utama dari  $\lambda$ -*calculus* adalah apa yang disebut dengan *expression*. Suatu *expression* dalam  $\lambda$ -*calculus* terdiri dari tiga bagian yaitu *lambda notation* ( $\lambda$ ), *argument* (seperti  $a$ ,  $b$ ,  $c$ ,  $x$ , dan lain-lain), dan *body* yang dipisahkan dengan tanda titik. Sebagai contoh, fungsi lambda

$\lambda x.x$  merupakan sebuah fungsi identitas yang mengambil argumen  $x$  kemudian mengembalikan nilai  $x$  itu sendiri. Dalam hal ini, terlihat bahwa notasi  $\lambda$  merupakan sebuah penanda bagi suatu fungsi lambda. Kemudian, pengubah  $x$  setelah notasi  $\lambda$  merupakan argumen dari fungsi tersebut. Selanjutnya, tanda titik merupakan pemisah antara *head* dan *body* fungsi lambda. Terakhir, setelah tanda titik adalah *body* dari suatu fungsi lambda yang mana berupa *expression*.

Untuk mempermudah pemahaman,  $\lambda$ -calculus dapat diperlakukan seperti fungsi tanpa nama. Sebagai contoh, fungsi lambda  $(\lambda x.x+5)$  apabila diberikan nilai 2 sehingga menjadi  $(\lambda x.x+5)2$  akan dievaluasi menjadi  $\lambda(2).(2)+5$ . Demikian itu, nilai yang dikembalikan oleh fungsi tersebut adalah 7. Sama seperti fungsi pada umumnya, konsep ini bernama *substitution* (substitusi). Memahami  $\lambda$ -calculus dirasa perlu berhubung dalam tugas akhir ini  $\lambda$ -calculus digunakan sebagai bentuk formal di *category* dalam konteks CCG *lexicon*. Meskipun  $\lambda$ -calculus tidak sesederhana yang dijelaskan sebelumnya, setidaknya memahami  $\lambda$ -calculus seperti ini sudah cukup untuk dapat membangun *supertagger* yang ada di tugas akhir ini.

## 2.4 CCGweb

CCGweb<sup>1</sup> merupakan *open source graphical annotation tool* pertama untuk CCG [1]. Aplikasinya berbasis web dan dibangun dengan menggunakan bahasa pemrograman Python, PHP, dan JavaScript. Fitur yang paling menarik dari *graphical annotation tool* adalah What You See Is What You Get (WYSIWYG) yang mana berupa kemampuan untuk *me-render* CCG *derivation* sesuai dengan apa yang kita lihat. Maksudnya, CCG *derivation* akan ditampilkan horizontal sesuai dengan panjang kalimatnya kemudian hasil *derivation*-nya ditampilkan vertikal seperti contoh pada Bagian 2.2.

Untuk dapat menggunakan CCGweb, kita perlu melakukan instalasi terlebih dahulu. Selanjutnya barulah kita dapat menambahkan kalimat-kalimat yang ingin dianotasi. Satu instalasi CCGweb hanya dapat digunakan untuk satu proyek anotasi sehingga apabila kita memiliki lebih dari satu proyek maka kita perlu melakukan instalasi CCGweb yang baru. Demikian itu, CCGtown<sup>2</sup> hadir dengan fitur *multi-project* dan tanpa perlu melakukan instalasi di komputer lokal karena aplikasinya *hosted* sehingga dapat diakses kapan pun.

---

<sup>1</sup><https://github.com/texttheater/ccgweb>

<sup>2</sup><https://github.com/wisn/ccgtown>

## Bab III

# Perancangan Sistem

CCGtown dibangun dengan menggunakan bahasa pemrograman Python dan JavaScript. Adapun *framework* yang digunakan adalah Django. Versi awal CCGtown merupakan sebuah *proof-of-concept* dari *open source graphical annotation tool* berbasis web yang dilengkapi dengan fitur pengannotasian semi-otomatis. Bahasa pemrograman Python digunakan karena sebagian besar *library* untuk CCG sudah tersedia di PyPi <sup>1</sup>. Salah satu *library* penting yang digunakan sebagai dasar dari fitur pengannotasian semi-otomatis adalah NTLK <sup>2</sup>. Selanjutnya, Django digunakan untuk mempercepat proses pengembangan aplikasi. Adapun JavaScript digunakan untuk menjadikan CCGtown aplikasi berbasis web yang interaktif.

Alur kerja CCGtown pada umumnya adalah (1) pengguna melakukan registrasi, (2) pengguna melakukan *login* ke sistem, (3) pengguna membuat proyek baru, (4) pengguna menambahkan kalimat yang ingin dianotasi, (5) pengguna melakukan anotasi kemudian melakukan *generate CCG derivation* dan/atau melakukan modifikasi *derivation*-nya apabila diperlukan, dan (6) pengguna melakukan *export* setelah selesai melakukan anotasi. Alur kerja tersebut mempengaruhi desain sistem dari CCGtown. Salah satunya adalah desain dari *database* yang akan digunakan.

### 3.1 Desain Database

CCGtown menggunakan PostgreSQL sebagai DBMS<sup>3</sup>-nya. Hal ini karena PostgreSQL memiliki kemampuan untuk menyimpan struktur data JSON<sup>4</sup> sehingga memudahkan CCGtown untuk menyimpan format JSON dari CCG *derivation* yang telah dimanipulasi oleh pengguna melalui fitur *editable CCG derivation*. PostgreSQL juga memiliki banyak fitur lain termasuk di antaranya dukungan dari *non-relational database model* (seperti *multi-model graph*) sehingga apabila di waktu yang akan datang CCGtown memerlukan perubahan

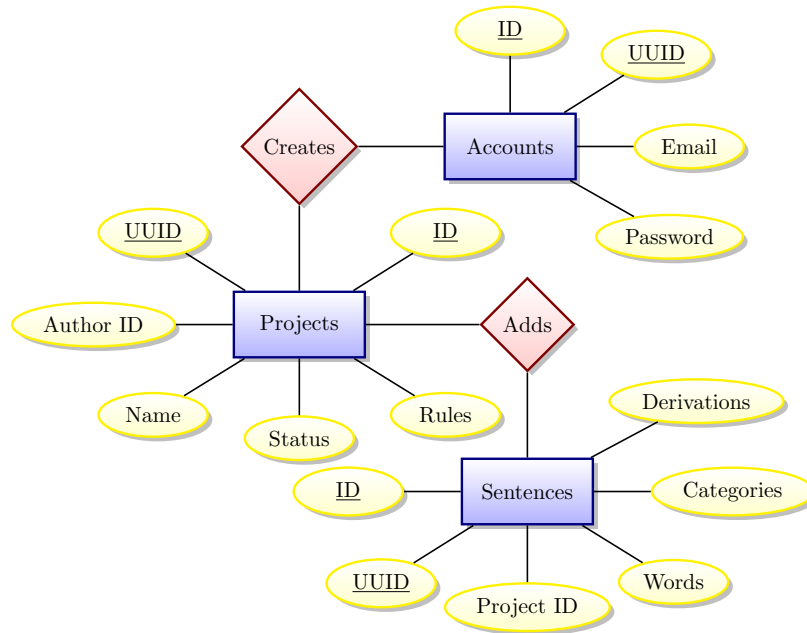
---

<sup>1</sup><https://pypi.org/>

<sup>2</sup><http://www.nltk.org/>

<sup>3</sup>Database Management System

<sup>4</sup>JavaScript Object Notation



Gambar 3.1: Conceptual Entity Relationship Diagram (ERD) CCGtown

an signifikan terhadap desain *database*-nya tidak perlu mengganti DBMS yang digunakan. Fitur lain seperti *function* dan *procedure* juga akan sangat membantu pengembangan CCGtown di waktu yang akan datang.

CCGtown versi awal sejatinya hanya membutuhkan tiga tabel saja yaitu tabel *accounts* untuk menyimpan pengguna yang terdaftar, tabel *projects* untuk menyimpan proyek-proyek yang sudah dibuat, dan tabel *sentences* untuk menyimpan kalimat-kalimat yang akan dianalisis. Tiga tabel tersebut sudah cukup untuk membangun *proof-of-concept* dari alat anotasi CCG yang akan dibangun. Adapun ERD<sup>5</sup>-nya dapat dilihat pada Gambar 3.1.

Masing-masing tabel memiliki dua *key* yaitu *ID* dan *UUID*<sup>6</sup>. *ID* merupakan *primary key integer* dengan *auto increment* yang berfungsi sebagai *identifier* untuk melakukan operasi *update* maupun *delete*. Adapun *UUID* merupakan *indexed column* yang berfungsi sebagai *identifier* publik (dapat dilihat oleh pengguna melalui URL) yang mana digunakan untuk operasi *read*. *ID* tidak digunakan sebagai *identifier* publik karena pengguna dapat melakukan *brute-force* untuk mencari proyek ataupun kalimat berdasarkan *ID* yang bukan miliknya. Demikian itu alasan ditambahkan atribut *UUID*. Alasan kenapa CCGtown tetap menyimpan kolom *ID* adalah karena *ID* nantinya akan digunakan untuk membuat *pagination*.

Pada tabel *accounts*, selain *ID* dan *UUID* juga memiliki atribut *email* dan

<sup>5</sup>Entity Relationship Diagram

<sup>6</sup>Universally Unique Identifier

*password*. Masing-masing atribut tersebut menggunakan tipe data *string* atau VARCHAR di PostgreSQL. Tabel *accounts* memiliki hubungan *one-to-many* terhadap tabel *projects*. Adapun atribut tabel *projects* adalah *author\_id*, *name*, *status*, dan *rules*. Atribut *author\_id* merupakan *foreign key (indexed)* yang mengarah kepada tabel *accounts* dan tipe data yang digunakan sama dengan atribut *ID* yang terdapat di tabel *accounts*. Atribut *name* menggunakan tipe data *string* (VARCHAR). Atribut *status* menggunakan tipe data *integer* yang berperan sebagai *enum* (0 = *just created*, 1 = *in progress*, 2 = *finished*, dan 3 = *dropped*). Tabel *projects* memiliki hubungan *one-to-many* terhadap tabel *sentences*. Adapun atribut tabel *sentences* adalah *project\_id*, *words*, *categories*, dan *derivations*. Atribut *project\_id* merupakan *foreign key (indexed)* yang mengarah kepada tabel *projects* dan tipe data yang digunakan sama dengan atribut *ID* yang terdapat di tabel *projects*. Sisanya, atribut *words*, *categories*, dan *derivations* menggunakan tipe data JSON.

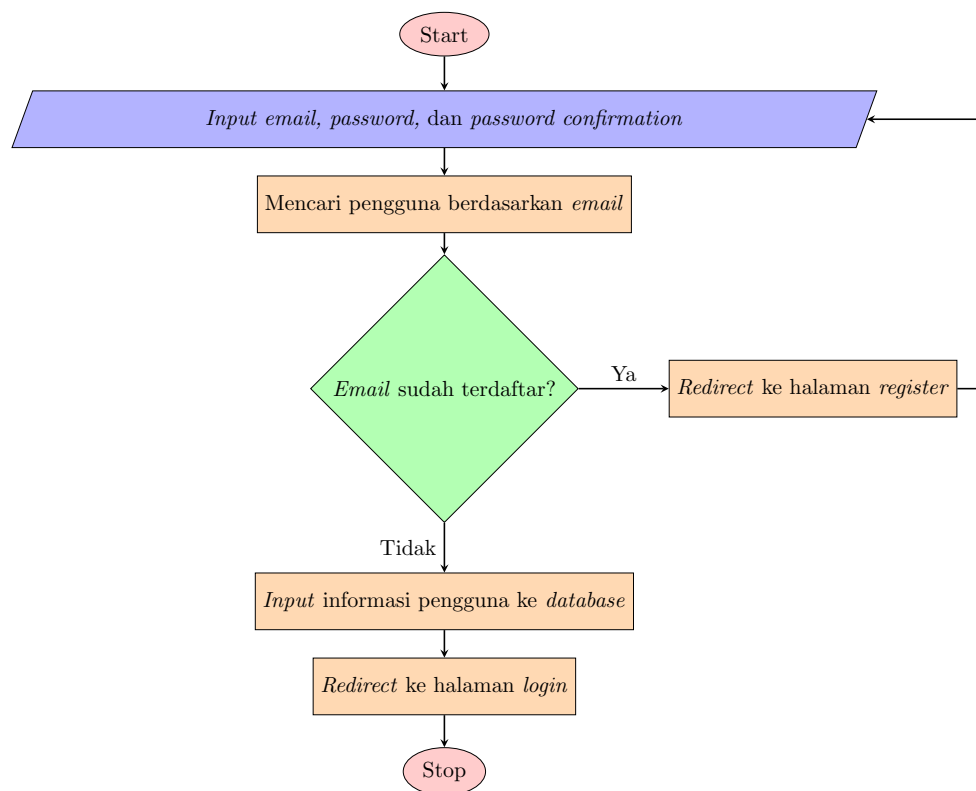
### 3.2 Desain Sistem

CCGtown sejatinya memiliki desain sistem yang cukup sederhana. Fungsionalitas yang akan didukung untuk versi awal adalah (1) *register* dan *login*, (2) manajemen proyek (CRUD<sup>7</sup>), (3) dan manajemen kalimat (CRUD). Pada manajemen kalimat, CCGtown menggunakan JavaScript untuk membuat pembuatan maupun perubahan CCG *derivation* menjadi lebih interaktif. Selain tiga fungsionalitas tersebut, CCGtown juga menambahkan fungsionalitas tambahan seperti *auto-assign category* yang dilakukan di sisi *frontend*. Kemudian, CCGtown juga menambahkan fungsionalitas tambahan di sisi *backend* yaitu CCG *derivation generator* dengan memanfaatkan *library* NLTK dan kemampuan untuk melakukan *export* CCG *derivation* yang disimpan di *database*.

Pengguna harus terdaftar terlebih dahulu sebelum dapat melakukan anotasi sehingga langkah awal yang harus dibangun adalah fungsionalitas *register*. Alur proses pendaftaran pengguna dapat dilihat pada Gambar 3.2. Berhubung fokus saat ini adalah *proof-of-concept*, informasi yang dibutuhkan untuk mendaftar hanyalah *email* dan *password*. Adapun *password confirmation* digunakan untuk memvalidasi *password* sehingga dapat mengurangi risiko pengguna melupakan *password*-nya yang baru saja di-*input*. Saat pengguna melakukan pendaftaran, sistem akan memeriksa apakah *email* yang didaftar sudah terdapat di *database*. Apabila sudah terdaftar, pengguna akan dialihkan ke halaman *register* kembali dan mendapatkan *flash message* dengan keterangan "email sudah terdaftar". Sebaliknya, sistem akan melakukan *input* data tersebut ke dalam *database* lalu mengalihkan pengguna ke halaman *login*. Ketika dialihkan ke halaman *login*, pengguna akan melihat *flash message* dengan keterangan "pengguna berhasil didaftarkan". Pada tahap ini pengguna sudah dapat melakukan *login* ke dalam sistem CCGtown.

---

<sup>7</sup>Create, Read, Update, Delete



Gambar 3.2: Alur proses pendaftaran pengguna.

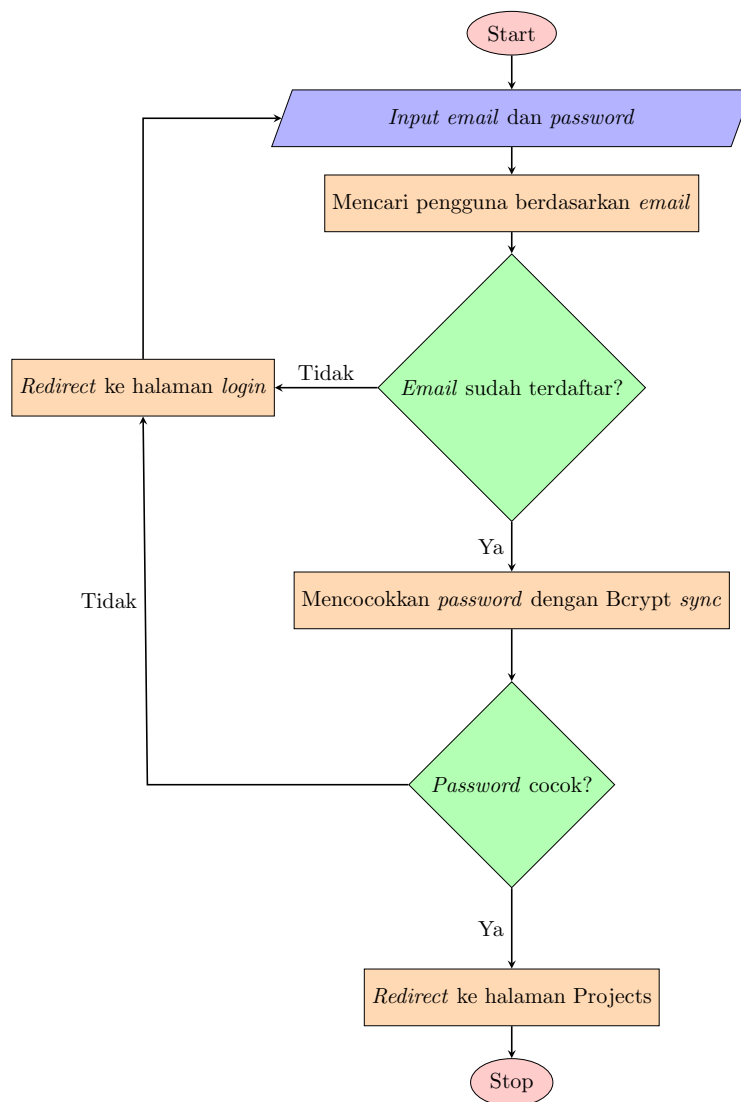
Pada proses "*input* informasi pengguna ke *database*" CCGtown melakukan *password hashing* dengan menggunakan Bcrypt. Informasi sensitif seperti *password* sebaiknya tidak disimpan sebagai *plain text*. Demikian itu CCGtown menggunakan *password hashing*. Apabila hal buruk terjadi seperti misalnya *data breach* (kebocoran data), *password* pengguna tidak dapat langsung digunakan. Peretas perlu mencari cara untuk memecahkan *password* tersebut. Bcrypt merupakan skema *password hashing* berbasis Blowfish *block cipher* yang didesain untuk lebih *resistant* terhadap serangan *brute-force* [4]. Serangan *brute-force* merupakan upaya peretas untuk menebak *password* dengan cara membuat *wordlist* yang kemudian dicocokkan dengan *hash* yang terbentuk satu-demi-satu. Meskipun terjadi *data breach*, peretas perlu usaha ekstra untuk dapat menebak *password* dari satu pengguna. Hal ini mengurangi kerugian yang akan dialami oleh CCGtown apabila *data breach* benar-benar terjadi.

Selanjutnya, setelah melakukan registrasi, pengguna dapat melakukan *login* ke sistem CCGtown. Proses yang dilakukan pada umumnya sama dengan aplikasi web yang memiliki kemampuan *register* dan *login*. Alur proses *login* dapat dilihat pada Gambar 3.3. Setelah pengguna melakukan *input email* dan *password*-nya, CCGtown akan melakukan pencarian di *database* apakah *email* yang diberikan terdaftar. Apabila tidak terdaftar, pengguna akan dialihkan ke halaman *login* dan diberikan *flash message* "*Email dan/atau password tidak cocok*". Pesan ini diberikan agar peretas tidak dapat mencari tahu *email* mana saja yang sudah terdaftar. Selanjutnya, apabila akun dengan *email* tersebut ada, maka langkah selanjutnya adalah mencocokkan *password* yang diberikan oleh pengguna dan *password* yang telah disimpan di *database*. Kemudian, sistem melakukan Bcrypt *sync*. Apabila tidak berhasil, pengguna akan dialihkan ke halaman *login* dan diberikan *flash message* "*Email dan/atau password tidak cocok*". Sebaliknya, pengguna akan dialihkan ke halaman Projects yang berisi daftar proyek yang telah dibuat sebelumnya.

Pada halaman Projects, pengguna dapat membuat proyek atau menghapus proyek. Tidak ada fungsionalitas spesial di halaman Projects selain CRUD pada umumnya. Satu pengguna dapat membuat banyak proyek. Tidak ada larangan tertentu terhadap penamaan proyek. Namun, sangat disarankan memberikan nama proyek yang deskriptif seperti misalnya "*Wide-range Indonesian Dataset*". Setiap proyek memiliki status yang berbeda-beda. Proyek yang baru saja dibuat akan memiliki status *just created*. Hal ini untuk memudahkan *annotator* mencari proyek mana yang baru akan dikerjakan, proyek mana yang sedang dikerjakan, proyek mana yang sudah selesai dikerjakan, atau proyek mana yang tidak jadi dikerjakan. Proyek yang telah dibuat dapat disunting maupun dihapus. Proyek yang dihapus tidak dapat dikembalikan (*undo*). Adapun penyuntingan proyek terjadi di halaman Editor.

Pada halaman Editor, pengguna dapat menyunting informasi proyek seperti nama proyek, status proyek, dan *rules* yang akan digunakan untuk me-





Gambar 3.3: Alur proses *login* ke sistem CCG.

lakukan *generate CCG derivation* via NTLK. Selain itu, pengguna juga dapat menambahkan kalimat baru yang akan dianotasi. Pengguna dapat menambahkan lebih dari satu kalimat sekaligus. Kalimat-kalimat tersebut akan di-*tokenize* menggunakan *library* NLTK. Ekstensi yang digunakan untuk proses *tokenize* ini adalah *punkt*. Setelah itu, barulah pengguna dapat melakukan pengamatan terhadap kalimat-kalimat yang telah ditambahkan. Terdapat dua cara untuk memberikan anotasi yaitu secara langsung di halaman Editor atau dapat juga dilakukan di Editable CCG Modal. Saat ini CCGtown belum mendukung pengamatan terhadap *compound words*. CCGtown saat ini juga belum mendukung pengamatan CCG dengan semantik. Versi awal CCGtown hanya mendukung pengamatan CCG secara sintaksis saja.

Setelah semua kata dalam suatu kalimat diberikan anotasi, pengguna dapat melakukan *generate CCG derivation*. Hal ini dapat dilakukan berkat bantuan *library* NLTK. Kami mengambil sebuah *rules* dari tabel *projects* dan kemudian kami mengambil semua *words* serta *categories* dari tabel *sentences* yang merupakan bagian dari proyek tersebut. Kolom *words* merupakan kumpulan kata dari kalimat yang telah di-*tokenize*. Adapun kolom *categories* merupakan anotasi CCG *category*-nya. *Pseudocode* untuk *generate CCG derivation* dapat dilihat pada Kode 3.1 dengan asumsi anotasi yang diberikan absah (dapat dibuat CCG *derivation*-nya). Kode *next* tersebut akan mengambil satu dari banyak kemungkinan *derivation* yang dapat dibuat. Contoh *object* yang di-*return* dapat dilihat pada Kode 3.2. Untuk kepentingan *rendering* di sisi *frontend*, *key* seperti *from* dan *to* sangat diperlukan. *Key from* dan *key to* merepresentasikan *index* posisi terhadap *array words*. Dengan bantuan kedua *key* tersebut, *frontend* dapat melakukan kalkulasi posisi masing-masing elemen yang terdapat di *object derivations*.

Kode 3.1: Pseudocode untuk melakukan *generate CCG derivation*.

---

```

1 from nltk.ccg import chart, lexicon
2
3 def generateCCGDerivation(rules, words, categories, target_words):
4     lex = rules + '\n\n'
5     for i in range(len(words)):
6         lex += words[i] + ' => ' + categories[i] + '\n'
7
8     lex = lexicon.parseLexicon(lex)
9     parser = chart.CCGChartParser(lex, chart.DefaultRuleSet)
10    result = next(parser.parse(target_words))
11    derivations = makeCCGDeriv(result)
12
13    return derivations

```

---

Kode 3.2 didapatkan dari fungsi *makeCCGDeriv* yang terdapat pada Kode 3.1. Fungsi *makeCCGDeriv* sederhananya mengambil *Tree* yang didapatkan dari *parser.parse* kemudian melakukan *tree traversal*. Semua *leaf*, diambil dari paling "kiri", diletakkan di elemen pertama *derivations*. Selanjutnya, kita berjalan melalui *parent* dari *leaf* tersebut hingga ke *root* mencari bentuk CCG

*derivation*-nya. Banyaknya baris yang dibutuhkan oleh CCG *derivation* dapat dilihat dari *height* yang dimiliki oleh *Tree* tersebut. Kemudian, hasil dari CCG *derivation* (umumnya berupa *S*) merupakan elemen terakhir *derivations*.

Kode 3.2: Contoh *derivations object* yang di-return.

---

```

1  [
2    [
3      { "to": 0, "from": 0, "word": "You" },
4      { "to": 1, "from": 1, "word": "prefer" },
5      { "to": 2, "from": 2, "word": "that" },
6      { "to": 3, "from": 3, "word": "cake" }
7    ],
8    [
9      { "to": 0, "from": 0, "category": "NP" },
10     { "to": 1, "from": 1, "category": "((S\NP)/NP)" },
11     { "to": 2, "from": 2, "category": "(NP/N)" },
12     { "to": 3, "from": 3, "category": "N" }
13   ],
14   [
15     { "to": 3, "from": 2, "category": "NP", "operator": ">" }
16   ],
17   [
18     { "to": 3, "from": 1, "category": "(S\NP)", "operator": ">" }
19   ],
20   [
21     { "to": 3, "from": 0, "category": "S", "operator": "<" }
22   ]
23 ]

```

---

### 3.3 Pertimbangan UI/UX

TBA.

## Daftar Pustaka

- [1] Kilian Evang, Lasha Abzianidze, and Johan Bos. CCGweb: a new annotation tool and a first quadrilingual CCG treebank. In *Proceedings of the 13th Linguistic Annotation Workshop*, pages 37–42, Florence, Italy, August 2019. Association for Computational Linguistics.
- [2] Julia Hockenmaier and Mark Steedman. CCGbank: A corpus of CCG derivations and dependency structures extracted from the Penn treebank. *Computational Linguistics*, 33(3):355–396, 2007.
- [3] J. Lambek. *Categorial and Categorical Grammars*, pages 297–317. Springer Netherlands, Dordrecht, 1988.
- [4] Katja Malvoni, Solar Designer, and Josip Knezovic. Are your passwords safe: Energy-efficient bcrypt cracking with low-cost parallel hardware. 08 2014.
- [5] Kiet Van Nguyen and Ngan Luu-Thuy Nguyen. Vietnamese transition-based dependency parsing with supertag features, 2019.
- [6] Raul Rojas. A tutorial introduction to the lambda calculus. *CoRR*, abs/1503.09060, 2015.
- [7] Mark Steedman. Categorial grammar. Technical report, 1992.
- [8] Mark Steedman. A very short introduction to ccg. Technical report, 1996.

## Lampiran