# Pendekatan *MapReduce* untuk Implementasi *Suffix Tree* pada *AutoComplete* Produk dengan Metodologi Agile

http://dx.doi.org/10.28932/jutisi.v5i2.1743

Yosef Ariyanto Irawan[#1], Billyanto Hendrik[*2], Antonius[#3], Taufan Abdurrachman[#4], Arie Tunggal ✉ [#5], Bernard R. Suteja[#6]

[#]*Program Studi Magister Ilmu Komputer, Fakultas Teknologi Informasi*
*Universitas Kristen Maranatha*
*Jl. Prof. drg. Surya Sumantri No.65 Bandung*

[1]yosef.xenoner@gmail.com
[2]hbillyanto@gmail.com
[3]stefanusantonius17@gmail.com
[4]taufanabd@gmail.com
[5]arie.tunggal@gmail.com
[6]bernard.rs@it.maranatha.edu

*Abstract* — **The development of technology and communication today encourages everyone to get everything quickly, easily and precisely, especially in the world of e-commerce which is increasingly widespread and needed by almost everyone. The hope is that with just one keyword, customers can find the right product using a search engine. Therefore, in this study, the Agile Scrum, MapReduce, and Suffix Tree methods are used to create an application that can help users to find products quickly and provide suggestions based on keywords. With a target result of an average search time of less than 100 ms, the prototype of the application already has a fast and accurate response time to display suggestions for customers.**

***Keywords*** — **Agile, autocomplete, e-commerce, search engine, suffix tree, MapReduce**

## I. PENDAHULUAN

### *A. Identifikasi Masalah*

Dengan kebutuhan informasi, pertumbuhan tingkat kecerdasan manusia dan perkembangan teknologi informasi khususnya di Indonesia berkembang sangat cepat saat ini. Teknologi yang semakin canggih memudahkan semua orang untuk memperoleh informasi yang mereka inginkan. Saat ini telah banyak sistem informasi yang digunakan untuk menunjang dan menyelesaikan suatu permasalahan yang biasanya timbul dalam suatu organisasi, perusahaan, atau instansi pemerintah, yang kini dikenal dengan *e-commerce* [1].

Dalam *e-commerce* terdapat berbagai jenis teknik penjualan, salah satunya adalah C2C (*Consumer to Consumer*). C2C merupakan teknik menjual produk atau jasa ke orang lain. Dapat juga disebut sebagai pelanggan ke pelanggan yaitu orang yang menjual produk dan jasa ke satu sama lain. Seperti halnya yang sering dijumpai pada beberapa situs jejaring sosial yang menawarkan barang ataupun jasa dari pelanggan ke pelanggan lainnya. Dalam hal ini bisnis ini disebut juga bisnis jual beli *online shopping* [1].

Kebanyakan *e-commerce* yang ada saat ini, seperti: Lazada, MatahariMall, Tokopedia, dan lainnya, masih menggunakan *search engine* yang sedikit saja memiliki kesalahan penulisan dalam pencarian barang akan membuat hasil yang diinginkan tidak keluar sesuai dengan keinginan user [2]. Diharapkan dengan menggunakan fitur *autocomplete* metode *suffix tree* pada *search engine* di web *e-commerce* ini akan memudahkan *user* dalam mencari produk yang diinginkan.

Berdasarkan permasalahan diatas, maka dicoba untuk menyelesaikan permasalahan tersebut dengan membuat prototipe aplikasi *autocomplete e-commerce* berbasis web menggunakan metode *suffix tree* berdasarkan produk paling laku, di mana untuk proses *autocomplete* menggunakan metode *suffix tree*.

*Autocomplete* ini hanya sebagian kecil fitur yang dapat digunakan pada *e-commerce*, tetapi walaupun ini hanya fitur kecil dalam pembuatannya cukup kompleks karena melibatkan data yang besar dan harus dapat dieksekusi dengan sangat cepat.

Permasalahan yang timbul dalam pembuatan *autocomplete* ini adalah dalam pengelolaan data yang besar dan eksekusi yang cepat, sehingga metode atau cara yang digunakan harus tepat. Dengan cepatnya eksekusi akan mendorong untuk memberikan sugesti produk terhadap *customer*.

### *B. Tujuan Penelitian*

Berdasarkan permasalahan yang ada, maka tujuan dari penelitian ini adalah:

1. Membuat pengaturan pembagian kerja dalam pembuatan aplikasi *autocomplete* menggunakan Metode Agile.
2. Mendapatkan pencarian yang cepat melalui *autocomplete* dengan *response time* kurang dari 0,1 detik atau 100 ms.
3. Memberikan sugesti kepada konsumen yang mencari produk dengan menampilkan 5 produk yang diurutkan berdasarkan produk paling laku.

## II. KAJIAN TEORI

### *A. Autocomplete*

*Autocomplete* adalah fitur pencarian yang memberikan saran untuk istilah pencarian saat pengguna mengetik teks di kotak pencarian. *Autocomplete* sebagai *plug-in* telah dilakukan di mana-mana pada pencarian situs besar maupun kecil. Penelitian tentang *autocomplete* mencakup berbagai istilah teknis yang merujuk pada sistem yang menggunakan arsitektur ini. Contohnya termasuk *Real Time Query Expansion* (RTQE), *enteractive query expansion*, *Search-as-you-Type* (SayT), *query completion*, *type-ahead search*, *auto-suggest, and suggestive searching*/ *search suggestions*. Kekhawatiran utama penelitian untuk *autocomplete* mencakup masalah yang terkait dengan arsitektur *backend* dan penilaian kepuasan pengguna dan sistem untuk implementasi spesifik [3].

Scrum *Methodology* merupakan salah satu model metode agile. Scrum pertama kali diperkenalkan oleh Jeff Sutherland awal tahun 1990an, dan dikembangkan selanjutnya dilakukan oleh Schwaber dan Beedle. Pada dasarnya Scrum merupakan salah satu komponen dari metodologi pengembangan Agile mengenai pertemuan harian untuk membahas kemajuan [11].

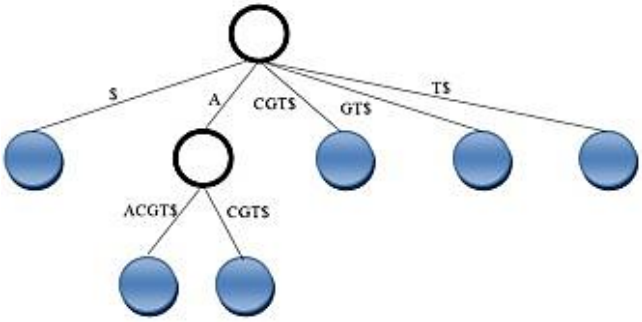
Scrum memiliki prinsip yaitu [11]:

a. Ukuran tim yang kecil melancarkan komunikasi, mengurangi biaya, dan memberdayakan satu sama lain.
b. Proses dapat beradaptasi terhadap perubahan teknik dan bisnis.
c. Proses menghasilkan beberapa *software increment*.
d. Pembangunan dan orang yang membangun dibagi dalam tim yang kecil.
e. Dokumentasi dan pengujian terus menerus dilakukan setelah *software* dibangun.
f. Proses scrum mampu menyatakan bahwa produk selesai kapanpun diperlukan.

### *B. Suffix Tree*

*Suffix tree* adalah struktur data yang merepresentasikan *suffix* (akhiran) dari string sedemikian sehingga memudahkan implementasi tertentu dengan cepat untuk berbagai operasi pada string. *Suffix tree* dari string S adalah pohon yang sisinya (edge) diberikan label string yang merupakan *suffix* dari S dan sesuai dengan tepat satu jalur dari *root* ke *leaf*. *Suffix tree* untuk string S yang panjangnya n didefinisikan sebagai suatu pohon sedemikian sehingga: Lintasan dari *root* ke *leaf* memiliki hubungan satu-satu dengan *suffix* dari *S*. Setiap *edge* (sisi) diisi string tak kosong. Semua simpul dalam (*internal node*) kecuali root memiliki paling sedikit dua anak (*child*).

Di dalam *suffix tree*, string *S* selalu ditambahkan dengan sebuah *terminal symbol* yang tidak termasuk di dalam substring *S*, biasanya dilambangkan dengan $.




Gambar 1. *Suffix tree* untuk string ‘AACGT’ [4]

*Suffix tree* juga menyediakan salah satu solusi waktu linear pertama untuk masalah *longest common substring*. Ini menyebabkan *suffix tree* sebuah string biasanya membutuhkan ruang yang jauh lebih banyak daripada menyimpan string itu sendiri [5]. Gambar 1 merupakan contoh dari bentuk *suffix tree* dengan string “AACGT”. Pada contoh tersebut menggambarkan 1 kata akan dipotong-potong untuk setiap huruf dari belakang dan dimasukan pada *tree*.”AACGT” akan dipotong menjadi “T”, ”GT”, ”CGT”, ”ACGT”, dan ”AACGT”. Pemotongan tersebut dijutukan agar sistem mengetahui seluruh subkata dari “AACGT”

### *C. MapReduce*

*MapReduce* adalah model pemrograman data-paralel yang dipelopori oleh Google untuk *cluster* mesin. Ini diterapkan untuk memproses dan menghasilkan kumpulan data yang besar untuk memecahkan berbagai masalah dan tugas di dunia nyata. Perhitungan ditentukan dalam istilah fungsi peta dan fungsi pengurangan, dan sistem *runtime* yang mendasarinya secara otomatis memparalelkan perhitungan melintasi *cluster* mesin besar, menangani kegagalan alat berat, dan menjadwalkan komunikasi antar-mesin, memanfaatkannya secara efektif jaringan [6].

Pada penelitian ini *MapReduce* akan memiliki peran dalam memproses data penjualan dari supermarket yang menjual berbagai macam produk yang memiliki tingkat

pembelian terbanyak hingga terendah, tentunya dengan metode ini akan sangat membantu mempercepat melakukan pencarian dan pengelompokan yang hasilnya harus secara cepat ditampilkan setelah kata kunci di masukan di *Autocomplete*.

Pada *MapReduce* akan terbagi menjadi 2 bagian yaitu *mapper* dan *reducer*. Pada *mapper*, system akan membagi data yang akan dikelola ke *node* dan akan menjalankan algoritma pada setiap data yang telah dibagikan pada setiap *node*. Sedangkan pada *reducer*, system akan menggabungkan kembali data yang telah diolah pada setiap *node* dan akan menjalankan sebuah algoritma saat data telah dikumpulkan. Algoritma pada *mapper* dan *reducer* dapat diatur oleh pengguna *MapReduce*. Untuk algorima yang digunakan akan dijelaskan pada BAB III.

*D. Hadoop*

Hadoop adalah alat penyimpanan dan pemrosesan data yang besar untuk menganalisis data, meliputi data dengan volume, variasi, dan kecepatan yang sangat besar. Hadoop adalah kerangka kerja yang berhubungan dengan *Big data* dan memiliki cara sendiri yang mendukung pemrosesan berbagai hal yang terikat dalam satu payung yang disebut Ekosistem Hadoop [7]. *MapReduce* dapat dijalankan dengan menggunakan Hadoop.

*E. Web Service:*

Web *Service* adalah suatu sistem perangkat lunak yang dirancang untuk mendukung interoperabilitas dan interaksi antar sistem pada suatu jaringan. Web *service* digunakan sebagai suatu fasilitas yang disediakan oleh suatu website untuk menyediakan layanan (dalam bentuk informasi) kepada sistem lain, sehingga sistem lain dapat berinteraksi dengan sistem tersebut melalui layanan-layanan (*service*) yang disediakan oleh suatu sistem yang menyediakan web *service* [8].

*Service Provider* berfungsi untuk menyediakan layanan/ *service* dan mengolah sebuah *registry* agar layanan-layanan tersebut dapat tersedia. *Service Registry* berfungsi sebagai lokasi central yang mendeskripsikan semua layanan/ *service* yang telah di-*register*. *Service Requestor* merupakan peminta layanan yang mencari dan menemukan layanan yang dibutuhkan serta menggunakan layanan tersebut.

*F. Java Script Object Notation (JSON)*

JSON adalah format pertukaran data yang ditemukan oleh Douglas Crockford pada tahun 2006 yang memiliki ukuran data yang lebih kecil serta waktu proses yang lebih cepat dibandingkan dengan XML yang sudah terlebih dulu ada. Sebelum ditemukannya JSON, web *service* yang ada menggunakan XML sebagai media pertukaran data, yang sudah menjadi standar dan umum digunakan oleh para programmer, namun sekarang JSON bisa juga digunakan sebagai media alternatif pertukaran data di dalam web *service* [9].

*G. .NET Core*

.NET Core adalah platform lintas, implementasi ulang sumber terbuka dari .NET *Framework*. Ini secara aktif dikelola oleh Microsoft dan komunitas pengembang yang besar di GitHub. Pada tahun 2016, Microsoft membeli Xamarin dan mulai mengintegrasikannya ke dalam set alat pengembang Microsoft [10].

*H. Metode Scrum - Agile Software Development*

Metode Agile adalah kumpulan dari metode-metode pengembangan perangkat lunak yang berbasis pada *Iterative* dan *Incremental* Model. Metode Agile memungkinkan untuk mengembangkan perangkat lunak yang memiliki *requirement* yang mudah berubah dengan cepat.

Scrum *Methodology* merupakan salah satu model metode agile. Scrum pertama kali diperkenalkan oleh Jeff Sutherland awal tahun 1990an, dan dikembangkan selanjutnya dilakukan oleh Schwaber dan Beedle. Pada dasarnya Scrum merupakan salah satu komponen dari metodologi pengembangan Agile mengenai pertemuan harian untuk membahas kemajuan [11].

Scrum memiliki prinsip yaitu [11]:

a. Ukuran tim yang kecil melancarkan komunikasi, mengurangi biaya, dan memberdayakan satu sama lain.
b. Proses dapat beradaptasi terhadap perubahan teknik dan bisnis.
c. Proses menghasilkan beberapa *software increment*.
d. Pembangunan dan orang yang membangun dibagi dalam tim yang kecil.
e. Dokumentasi dan pengujian terus menerus dilakukan setelah *software* dibangun.
f. Proses scrum mampu menyatakan bahwa produk selesai kapanpun diperlukan.

*I. Web service synonym*

Web *service synonym* merupakan web *service* yang menyediakan pencocokan sinonim kata melalui API. Webs service yang digunakan diambil dari https://similiarwords.p.rapidapi.com [12].

*J. Postman*

Postman adalah sebuah aplikasi (berupa *plugin*) pengetesan API untuk mengetahui sejauh mana *performance request* atau *response time* yang terjadi pada saat *searching* suatu kata. [13]

## III. ANALISIS DAN RANCANGAN SISTEM

Aplikasi harus dapat memberikan *suggestion* 5 produk yang paling banyak orang beli. Saat *user* mengetikkan kata pada kolom *search* minimal 3 huruf, agen aplikasi akan mencarikan nama produk yang mengandung kata tersebut, dan aplikasi akan memberikan *suggestion* 5 produk terlaris.

Aplikasi *autocomplete* sangat memiliki komponen yang sangat penting, yaitu waktu proses. Semakin lama waktu

proses akan semakin menurunkan nilai guna dari *autocomplete* tersebut. "*0.1 second is about the limit for having the user feel that the system is reacting instantaneously, meaning that no special feedback is necessary except to display the result*" [14]. Maka aplikasi *autocomplete* akan mematok *response time* 0,1 detik untuk memberikan *suggestion* 5 produk terlaris.

Dalam penelitian ini, data yang digunakan dari *website Kaggle*. Data ini adalah data mengenai penjualan pada sebuah supermarket selama 6 bulan penjualan supermarket tersebut, yang memuat 6.758.125 transaksi dan memiliki 452 jenis produk. Sumber data didapatkan dari *website Kaggle* pada link berikut ini: https://www.kaggle.com/codemysteries/salesdb.

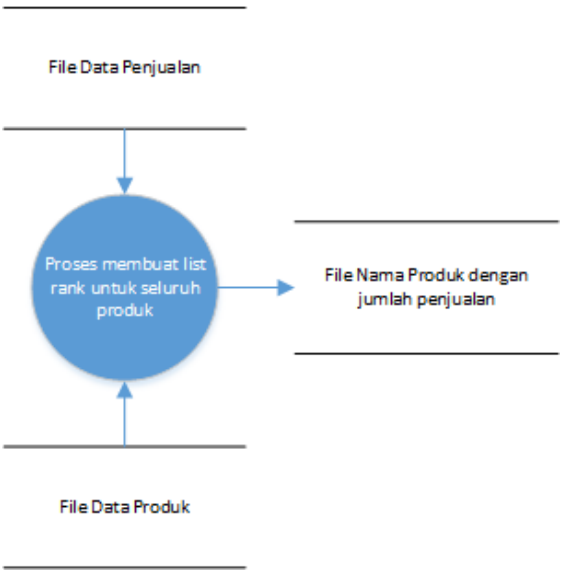
Dengan aplikasi memberikan *suggestion* 5 produk terlaris, maka aplikasi akan memerlukan tabel *ranking* untuk mengetahui produk mana yang memiliki bobot paling tinggi beserta urutannya. Tabel *ranking* dari 6.758.125 transaksi akan diolah dengan *hadoop MapReduce*, proses *MapReduce* tersebut akan menghitung jumlah produk yang sudah laku pada tabel *sales*, sehingga diperoleh tabel ranking dari produk tersebut berdasarkan dari jumlah produk yang terbanyak dibeli.

Aplikasi tidak hanya untuk memberi sugesti yang tepat saja, tetapi aplikasi diharapkan dapat menghasilkan dengan waktu kurang dari 0.1 detik, maka dari itu *string matching* yang digunakan haruslah algoritma yang cepat untuk memenuhi harapan tersebut, aplikasi akan menggunakan *suffix tree* untuk *string matching*. Dari tabel *ranking* tersebut didaftarkan *string* dan *index* pada *suffix tree*. Saat aplikasi di masukan kata kunci, aplikasi akan mencari kata tersebut pada *suffix tree*, setelah dapat indeksnya aplikasi akan melihat *ranking*-nya pada tabel *ranking*.

Tahapan dalam pembuatan aplikasi *autocomplete* ini meliput:

### *A. Data Flow Diagram (DFD)*

Data flow diagram pada Gambar 2 menggambarkan proses membuat list rank dari produk. Proses di atas menggunakan data dari penjualan untuk mengetahui lakunya dari sebuah produk dan menggunakan data dari master produk untuk mengetahui nama dari produk yang laku dari penjualan. Proses pada Gambar 2 akan menghasilkan data nama produk dengan jumlah penjualan. *MapReduce* dengan Hadoop akan digunakan pada proses tersebut.

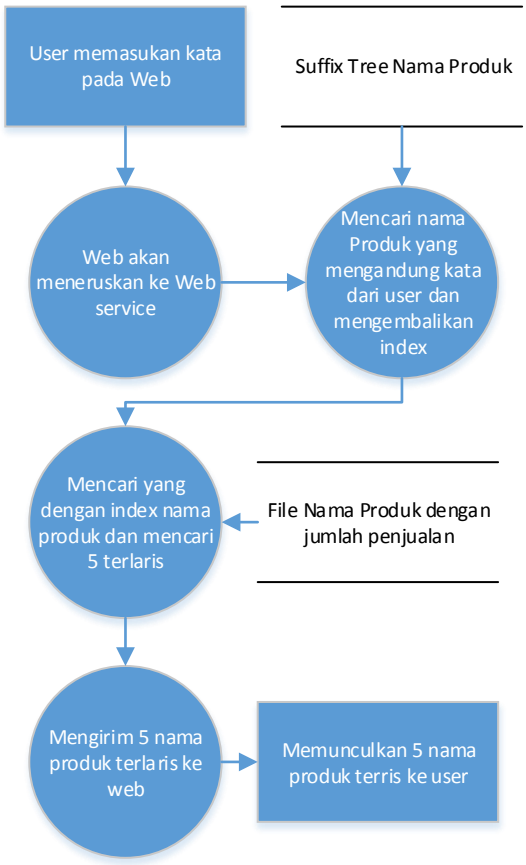



Gambar 2. DFD proses membuat *list rank* produk

Gambar 3 menggambarkan proses membuat *Suffix Tree*. Data dari proses pada Gambar 2 digunakan kembali untuk membuat *Suffix Tree*. *Suffix Tree* yang dihasilkan memiliki indeks yang mengarahkan ke file nama produk dengan jumlah penjualan. Proses pembuatan *Suffix Tree* akan menggunakan *MapReduce* dengan Hadoop.


Gambar 3. DFD proses membuat *suffix tree* untuk nama produk

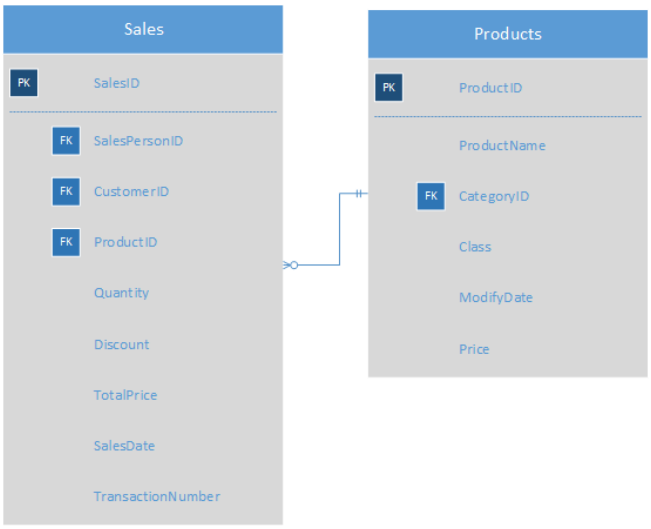
Pada gambar 4, *user* memasukan sepotong kata pada *autocomplete*. Web akan langsung meneruskan kata tersebut ke web *service autocomplete*. Pada web *service* akan menerima kata tersebut dan mencari kata tersebut para *suffix tree*. Setelah mendapat kata yang mengandung kata yang diinput oleh *user* akan dilihat *index* yang dimiliki oleh kata tersebut. *Service* akan mencari *index* tersebut pada *file* nama produk dan jumlah penjualan. *Service* akan mencarikan 5 produk yang memiliki jumlah penjualan paling banyak. Setelah mendapatkan nama produk-produknya, *service* akan mengirim kembali ke web. Setelah web mendapatkan *response* dari *service*, web akan langsung menampilkan 5 nama produk terlaris tersebut ke *user*. Transaksi data antar web *service* dengan web akan menggunakan JSON.




Gambar 4. DFD proses *autocomplete*

## B. *Perancangan Basis Data*

Sistem *autocomplete* ini menggunakan data dari *Kaggle*. Pada data tersebut terdapat beberapa tabel, tetapi pada prosesnya hanya akan menggunakan tabel produk dan penjualan karena tidak membutuhkan data yang lain.

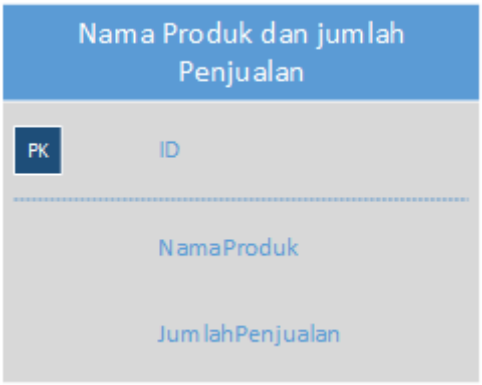



Gambar 5. Struktur data *sales* dan produk

Gambar 5 merupakan ERD data *sales* dan produk. Pada *entity sales* terdapat *foreign key product* ID yang dapat sambungkan dengan *table product* untuk mendapatkan *product name*. Dari tabel *sales* terdapat beberapa *foreign key* selain produk, tetapi tetap menggunakan data tersebut.

Data *sales* dan produk akan diolah menjadi data nama produk dan jumlah transaksi, untuk mendapatkan *ranking* penjualan pada setiap produk. Untuk mendapatkan *ranking* maka menggunakan *hadoop*. Pembobotan *ranking* yang akan digunakan adalah jumlah penjualan pada periode 6 bulan.

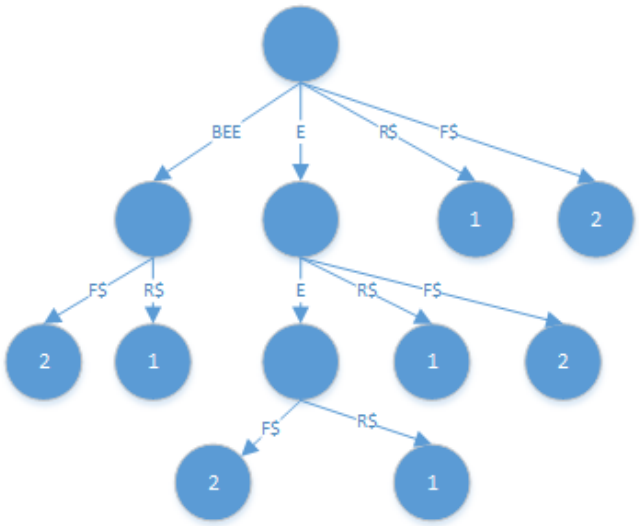
Pada *hadoop* terjadi proses *MapReduce* untuk mendapatkan *ranking* tersebut. Proses *MapReduce* akan terbagi menjadi 2 bagian bagian *mapper* dan bagian *reduce*. Pada algoritma penghitungan *ranking* pejualan ini, proses *mapper* akan mengeluarkan hasil data yang memiliki struktur seperti Gambar 6 dan memiliki jumlah penjualan adalah 1 untuk setiap datanya. Setelah proses *mapper hadoop* akan menjalankan proses *reducer*. Pada proses *reducer*, *hadoop* akan menjumlah seluruh data dari hasil *mapper*. Pada setiap data yang memiliki *ID* yang sama *hadoop* akan menjumlahkan jumlah penjualan dan mengeluarkan datanya unik pada setiap *ID* yang sebelum nya akan terdapat duplikasi pada setiap *ID* pada hasil proses *mapper*.




Gambar 6. Struktur data *ranking*

Gambar 6 merupakan struktur data *ranking* yang merupakan hasil dari *MapReduce* dari data *sales* dan produk. ID akan digunakan *indexing* dari *suffix*.

*p-ISSN : 2443-2210*
*e-ISSN : 2443-2229*




Gambar 7. Struktur *Suffix Tree*

Gambar 7 merupakan struktur *suffix tree* yang akan digunakan pada *suffix tree*, dengan contoh kata "BEER" dan "BEEF". "BEER" memiliki *index* 1 dan "BEEF" memiliki *index* 2. Di setiap *leaf* pada *tree* tersebut memiliki *index* nama produk, lalu dari *index* tersebut aplikasi akan mencari pada data *ranking* yang telah disimpan.

*C. Penerapan Metode Scrum*

Dalam penerapan *metode scrum* dilakukan pembagian tugas yang terdiri dari *product owner*, *scrum master*, *project manager* dan *developer*. Lalu ditentukan estimasi panjang atau lamanya pengerjaan selama 9 minggu, serta membuat *product backlog* yang berisi daftar *user stories* yang diharapkan akan dibuat dan diselesaikan dalam *sprint* pembuatan program *autocomplete* ini.

Sebelum membuat *sprint* backlog terlebih dahulu menentukan *product backlog* yang berisi *backlog item* yang diinginkan atau dicapai dalam pembuatan sistem *autocomplete* ini. Tabel I dibawah ini menerangkan *backlog item* yang ingin dicapai. Dari *backlog item* yang terdapat pada Tabel I, dikembangkan menjadi *task-task* yang diperlukan dalam pembuatan sistem *autocomplete*, seperti pada Table II hingga Tabel X *Sprint Backlog*.

TABEL I
*PRODUCT BACKLOG*

| No. | *Backlog Item* | *Estimate (Days)* |
|---|---|---|
| 1 | *As a user, I want to search a product with autocomplete* | 40 Days |

TABEL II
*SPRINT BACKLOG - WEEK* 1

| **40 *days work in this sprint* (Minggu ke-1)** | | | ***Date Sprint Days*** | | **15/4** | **16/4** | **17/4** | **18/4** | **19/4** |
|---|---|---|---|---|---|---|---|---|---|
| | | | ***Hours Remaining*** | | **512** | **512** | **512** | **512** | **512** |
| **No.** | ***Task*** | **Bobot** | ***Estimate*** | | ***Remaining Hours*** | | | | |
| | | | ***Days*** | ***Hours*** | ***Mon*** | ***Tue*** | ***Wed*** | ***Thu*** | ***Fri*** |
| 1 | Pembuatan *list rank* dengan Hadoop/GCP | 4 | 5 | 80 | 0 | 0 | 0 | 0 | 0 |
| 2 | Pembuatan *suffix tree* dengan Hadoop/GCP | 8 | 6 | 96 | 0 | 0 | 0 | 0 | 0 |
| 3 | Pembuatan web *service* | 4 | 5 | 80 | 0 | 0 | 0 | 0 | 0 |
| 4 | Pembuatan algoritma string *matching* | 8 | 6 | 96 | 0 | 0 | 0 | 0 | 0 |
| 5 | Pembuatan web | 4 | 5 | 80 | 0 | 0 | 0 | 0 | 0 |
| 6 | Penggabungan *backend*, *frontend program* dan pengujian | 4 | 5 | 80 | 0 | 0 | 0 | 0 | 0 |
| | Total Waktu per Hari | | 32 | 512 | 0 | 0 | 0 | 0 | 0 |
| | Sisa Waktu | 32 | 32 | 512 | 512 | 512 | 512 | 512 | 512 |

TABEL III
*SPRINT BACKLOG - WEEK* 2

| **40 *days work in this sprint* (Minggu ke-2)** | | | ***Date Sprint Days*** | | **22/4** | **23/4** | **24/4** | **25/4** | **26/4** |
|---|---|---|---|---|---|---|---|---|---|
| | | | ***Hours Remaining*** | | **512** | **512** | **512** | **512** | **496** |
| **No.** | ***Task*** | **Bobot** | ***Estimate*** | | ***Remaining Hours*** | | | | |
| | | | ***Days*** | ***Hours*** | ***Mon*** | ***Tue*** | ***Wed*** | ***Thu*** | ***Fri*** |
| 1 | Pembuatan *list rank* dengan Hadoop/GCP | 4 | 5 | 80 | 0 | 0 | 0 | 8 | 8 |
| 2 | Pembuatan *suffix tree* dengan Hadoop/GCP | 8 | 6 | 96 | 0 | 0 | 0 | 0 | 0 |
| 3 | Pembuatan web *service* | 4 | 5 | 80 | 0 | 0 | 0 | 8 | 8 |
| 4 | Pembuatan algoritma string *matching* | 8 | 6 | 96 | 0 | 0 | 0 | 0 | 0 |
| 5 | Pembuatan web | 4 | 5 | 80 | 0 | 0 | 0 | 0 | 0 |
| 6 | Penggabungan *backend*, *frontend program* dan pengujian | 4 | 5 | 80 | 0 | 0 | 0 | 0 | 0 |
| | Total Waktu per Hari | | 32 | 512 | 0 | 0 | 0 | 16 | 16 |
| | Sisa Waktu | 32 | 32 | 512 | 512 | 512 | 512 | 496 | 480 |

TABEL IV
*SPRINT BACKLOG - WEEK* 3

| **40 *days work in this sprint* (Minggu ke-3)** | | | ***Date Sprint Days*** | | **29/4** | **30/4** | **01/5** | **02/5** | **03/5** |
|---|---|---|---|---|---|---|---|---|---|
| | | | ***Hours Remaining*** | | **480** | **464** | **448** | **448** | **432** |
| **No.** | ***Task*** | **Bobot** | ***Estimate*** | | ***Remaining Hours*** | | | | |
| | | | ***Days*** | ***Hours*** | ***Mon*** | ***Tue*** | ***Wed*** | ***Thu*** | ***Fri*** |
| 1 | Pembuatan *list rank* dengan Hadoop/GCP | 4 | 5 | 80 | 8 | 8 | 0 | 0 | 8 |
| 2 | Pembuatan *suffix tree* dengan Hadoop/GCP | 8 | 6 | 96 | 0 | 0 | 0 | 8 | 4 |
| 3 | Pembuatan web *service* | 4 | 5 | 80 | 0 | 0 | 0 | 0 | 0 |
| 4 | Pembuatan algoritma string *matching* | 8 | 6 | 96 | 0 | 8 | 0 | 8 | 0 |
| 5 | Pembuatan web | 4 | 5 | 80 | 8 | 0 | 0 | 0 | 4 |
| 6 | Penggabungan *backend*, *frontend program* dan pengujian | 4 | 5 | 80 | 0 | 0 | 0 | 0 | 0 |
| | Total Waktu per Hari | | 32 | 512 | 16 | 16 | 0 | 16 | 16 |
| | Sisa Waktu | 32 | 32 | 512 | 464 | 448 | 448 | 432 | 416 |

TABEL V
*SPRINT BACKLOG - WEEK* 4

| **40 *days work in this sprint* (Minggu ke-4)** | | | ***Date Sprint Days*** | | **06/5** | **07/5** | **08/5** | **09/5** | **10/5** |
|---|---|---|---|---|---|---|---|---|---|
| | | | ***Hours Remaining*** | | **416** | **400** | **384** | **368** | **352** |
| **No.** | ***Task*** | **Bobot** | ***Estimate*** | | ***Remaining Hours*** | | | | |
| | | | ***Days*** | ***Hours*** | ***Mon*** | ***Tue*** | ***Wed*** | ***Thu*** | ***Fri*** |
| 1 | Pembuatan *list rank* dengan Hadoop/GCP | 4 | 5 | 80 | 8 | 8 | 8 | 0 | 0 |
| 2 | Pembuatan *suffix tree* dengan Hadoop/GCP | 8 | 6 | 96 | 0 | 4 | 0 | 8 | 4 |
| 3 | Pembuatan web *service* | 4 | 5 | 80 | 4 | 4 | 0 | 0 | 4 |
| 4 | Pembuatan algoritma string *matching* | 8 | 6 | 96 | 0 | 0 | 4 | 4 | 4 |
| 5 | Pembuatan web | 4 | 5 | 80 | 4 | 0 | 4 | 4 | 4 |
| 6 | Penggabungan *backend*, *frontend program* dan pengujian | 4 | 5 | 80 | 0 | 0 | 0 | 0 | 0 |
| Total Waktu per Hari | | | 32 | 512 | 16 | 16 | 16 | 16 | 16 |
| Sisa Waktu | | 32 | 32 | 512 | 400 | 384 | 368 | 352 | 336 |

TABEL VI
*SPRINT BACKLOG - WEEK* 5

| **40 *days work in this sprint* (Minggu ke-5)** | | | ***Date Sprint Days*** | | **13/5** | **14/5** | **15/5** | **16/5** | **17/5** |
|---|---|---|---|---|---|---|---|---|---|
| | | | ***Hours Remaining*** | | **336** | **320** | **304** | **288** | **272** |
| **No.** | ***Task*** | **Bobot** | ***Estimate*** | | ***Remaining Hours*** | | | | |
| | | | ***Days*** | ***Hours*** | ***Mon*** | ***Tue*** | ***Wed*** | ***Thu*** | ***Fri*** |
| 1 | Pembuatan *list rank* dengan Hadoop/GCP | 4 | 5 | 80 | 8 | 0 | 8 | 0 | 0 |
| 2 | Pembuatan *suffix tree* dengan Hadoop/GCP | 8 | 6 | 96 | 4 | 4 | 0 | 8 | 0 |
| 3 | Pembuatan web *service* | 4 | 5 | 80 | 2 | 4 | 0 | 6 | 8 |
| 4 | Pembuatan algoritma string *matching* | 8 | 6 | 96 | 2 | 6 | 4 | 2 | 4 |
| 5 | Pembuatan web | 4 | 5 | 80 | 0 | 2 | 4 | 0 | 4 |
| 6 | Penggabungan *backend*, *frontend program* dan pengujian | 4 | 5 | 80 | 0 | 0 | 0 | 0 | 0 |
| Total Waktu per Hari | | | 32 | 512 | 16 | 16 | 16 | 16 | 16 |
| Sisa Waktu | | 32 | 32 | 512 | 320 | 304 | 288 | 272 | 256 |

TABEL VII
*SPRINT BACKLOG - WEEK* 6

| **40 *days work in this sprint* (Minggu ke-6)** | | | ***Date Sprint Days*** | | **20/5** | **21/5** | **22/5** | **23/5** | **24/5** |
|---|---|---|---|---|---|---|---|---|---|
| | | | ***Hours Remaining*** | | **256** | **236** | **220** | **200** | **180** |
| **No.** | ***Task*** | **Bobot** | ***Estimate*** | | ***Remaining Hours*** | | | | |
| | | | ***Days*** | ***Hours*** | ***Mon*** | ***Tue*** | ***Wed*** | ***Thu*** | ***Fri*** |
| 1 | Pembuatan *list rank* dengan Hadoop/GCP | 4 | 5 | 80 | 0 | 0 | 0 | 0 | 0 |
| 2 | Pembuatan *suffix tree* dengan Hadoop/GCP | 8 | 6 | 96 | 6 | 6 | 6 | 6 | 6 |
| 3 | Pembuatan web *service* | 4 | 5 | 80 | 4 | 2 | 4 | 4 | 4 |
| 4 | Pembuatan algoritma string *matching* | 8 | 6 | 96 | 6 | 4 | 6 | 4 | 6 |
| 5 | Pembuatan web | 4 | 5 | 80 | 4 | 4 | 4 | 6 | 4 |
| 6 | Penggabungan *backend*, *frontend program* dan pengujian | 4 | 5 | 80 | 0 | 0 | 0 | 0 | 0 |
| Total Waktu per Hari | | | 32 | 512 | 20 | 16 | 20 | 20 | 20 |
| Sisa Waktu | | 32 | 32 | 512 | 236 | 220 | 200 | 180 | 160 |

TABEL VIII
*SPRINT BACKLOG - WEEK* 7

| **40 *days work in this sprint* (Minggu ke-7)** | | | ***Date Sprint Days*** | | **27/5** | **28/5** | **29/5** | **30/5** | **31/5** |
|---|---|---|---|---|---|---|---|---|---|
| | | | ***Hours Remaining*** | | **160** | **144** | **126** | **108** | **108** |
| **No.** | ***Task*** | **Bobot** | ***Estimate*** | | ***Remaining Hours*** | | | | |
| | | | ***Days*** | ***Hours*** | ***Mon*** | ***Tue*** | ***Wed*** | ***Thu*** | ***Fri*** |
| 1 | Pembuatan *list rank* dengan Hadoop/GCP | 4 | 5 | 80 | 0 | 0 | 0 | 0 | 0 |
| 2 | Pembuatan *suffix tree* dengan Hadoop/GCP | 8 | 6 | 96 | 6 | 6 | 6 | 0 | 4 |
| 3 | Pembuatan web *service* | 4 | 5 | 80 | 4 | 4 | 4 | 0 | 2 |
| 4 | Pembuatan algoritma string *matching* | 8 | 6 | 96 | 4 | 6 | 6 | 0 | 8 |
| 5 | Pembuatan web | 4 | 5 | 80 | 2 | 2 | 2 | 0 | 2 |
| 6 | Penggabungan *backend*, *frontend program* dan pengujian | 4 | 5 | 80 | 0 | 0 | 0 | 0 | 0 |
| Total Waktu per Hari | | | 32 | 512 | 16 | 18 | 8 | 0 | 16 |
| Sisa Waktu | | 32 | 32 | 512 | 144 | 126 | 108 | 108 | 92 |

TABEL IX
*SPRINT BACKLOG - WEEK* 8

| **40 *days work in this sprint* (Minggu ke-8)** | | | ***Date Sprint Days*** | | **10/6** | **11/6** | **12/6** | **13/6** | **14/6** |
|---|---|---|---|---|---|---|---|---|---|
| | | | ***Hours Remaining*** | | **92** | **76** | **60** | **44** | **28** |
| **No.** | ***Task*** | **Bobot** | ***Estimate*** | | ***Remaining Hours*** | | | | |
| | | | ***Days*** | ***Hours*** | ***Mon*** | ***Tue*** | ***Wed*** | ***Thu*** | ***Fri*** |
| 1 | Pembuatan *list rank* dengan Hadoop/GCP | 4 | 5 | 80 | 0 | 0 | 0 | 0 | 0 |
| 2 | Pembuatan *suffix tree* dengan Hadoop/GCP | 8 | 6 | 96 | 0 | 0 | 0 | 0 | 0 |
| 3 | Pembuatan web *service* | 4 | 5 | 80 | 0 | 0 | 0 | 0 | 0 |
| 4 | Pembuatan algoritma string *matching* | 8 | 6 | 96 | 0 | 0 | 0 | 0 | 0 |
| 5 | Pembuatan web | 4 | 5 | 80 | 2 | 2 | 4 | 4 | 0 |
| 6 | Penggabungan *backend*, *frontend program* dan pengujian | 4 | 5 | 80 | 14 | 14 | 12 | 12 | 16 |
| Total Waktu per Hari | | | 32 | 512 | 16 | 16 | 16 | 16 | 16 |
| Sisa Waktu | | 32 | 32 | 512 | 76 | 60 | 44 | 28 | 12 |

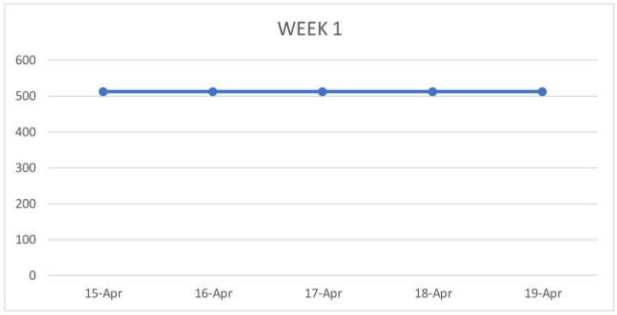
TABEL X
*SPRINT BACKLOG - WEEK* 9

| **40 *days work in this sprint* (Minggu ke-9)** | | | ***Date Sprint Days*** | | **17/6** | **18/6** | **19/6** | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | ***Hours Remaining*** | | **12** | **4** | **0** | | |
| **No.** | ***Task*** | **Bobot** | ***Estimate*** | | ***Remaining Hours*** | | | | |
| | | | ***Days*** | ***Hours*** | ***Mon*** | ***Tue*** | ***Wed*** | ***Thu*** | ***Fri*** |
| 1 | Pembuatan *list rank* dengan Hadoop/GCP | 4 | 5 | 80 | 0 | 0 | 0 | | |
| 2 | Pembuatan *suffix tree* dengan Hadoop/GCP | 8 | 6 | 96 | 0 | 0 | 0 | | |
| 3 | Pembuatan web *service* | 4 | 5 | 80 | 0 | 0 | 0 | | |
| 4 | Pembuatan algoritma string *matching* | 8 | 6 | 96 | 0 | 0 | 0 | | |
| 5 | Pembuatan web | 4 | 5 | 80 | 0 | 0 | 0 | | |
| 6 | Penggabungan *backend*, *frontend program* dan pengujian | 4 | 5 | 80 | 8 | 4 | 0 | | |
| Total Waktu per Hari | | | 32 | 512 | 8 | 4 | 0 | | |
| Sisa Waktu | | 32 | 32 | 512 | 4 | 0 | 0 | | |

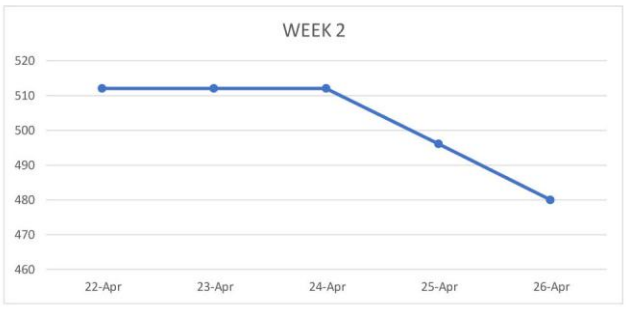
Setiap harinya anggota tim secara rutin melakukan *update progress* harian (*daily scrum*) masing-masing. Setiap 1 minggu diadakan *sprint review meeting* untuk menunjukkan *progress* kerja berupa demonstrasi kepada *product owner*, meliputi pembahasan 3 hal, yaitu:

a. Apa yang sudah dikerjakan selama periode *sprint* kemarin?
b. Apa yang akan dikerjakan selama periode *sprint* berikutnya?
c. Kendala yang dialami saat pengerjaan.

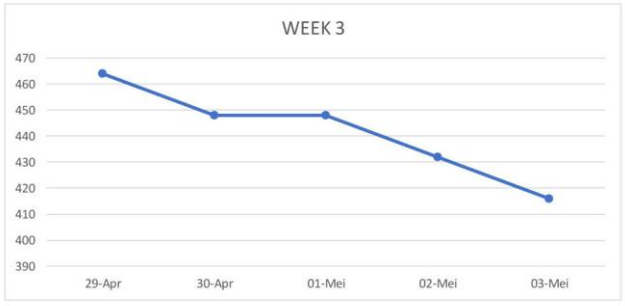
Hasil *meeting* tersebut dimasukkan ke dalam *burndown chart*. *Progress* yang telah dilakukan dapat dilihat pada *burndown chart* pada Gambar 8 hingga Gambar 16.




Gambar 8. *Burndown Chart - week 1*




Gambar 9. *Burndown Chart - week 2*




Gambar 10. *Burndown Chart - week 3*


Gambar 11. *Burndown Chart - week 4*

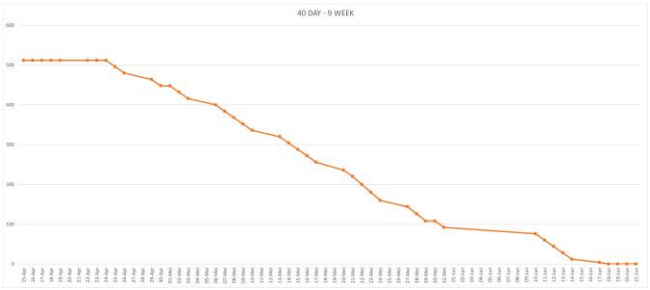

Gambar 12. *Burndown Chart - week 5*


Gambar 13. *Burndown Chart - week 6*


Gambar 14. *Burndown Chart - week 7*


Gambar 15. *Burndown Chart - week 8*


Gambar 16. *Burndown Chart – week 9*




Gambar 17. *Burndown Chart – All Weeks*

Pada *burndown chart* tersebut banyaknya *task* yang telah dikerjakan dapat diukur dan dibandingkan dengan sisa waktu yang diestimasikan. Terlihat belum ada pengerjaan pada minggu pertama. Disinilah peran *scrum master* diperlukan untuk memperbaiki estimasi dan memastikan *sprint* tetap berjalan sesuai dengan yang telah ditentukan.

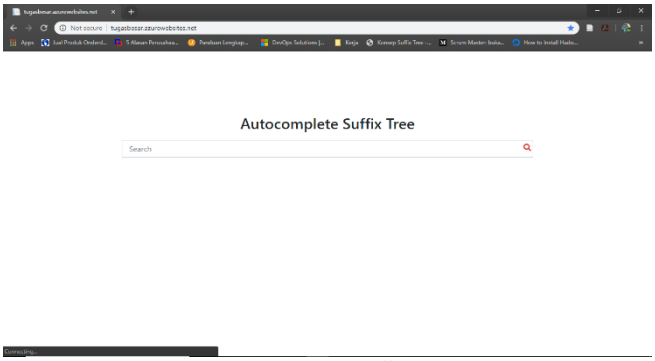
Pada minggu kedua *sprint* mulai berjalan dan pengerjaan *task* mulai berjalan secara konsisten pada minggu ketiga hingga minggu ke delapan. Pada minggu terakhir, pengerjaan selesai tepat waktu dan memenuhi tujuan *sprint*.

Pada setiap *sprint* yang berakhir dilakukan *sprint retrospective*, setelah itu didiskusikan alasan *progress* yang tersendat pada minggu pertama dan paruh kedua, serta menyelesaikan masalah sehingga akan lebih baik di *sprint* berikutnya.

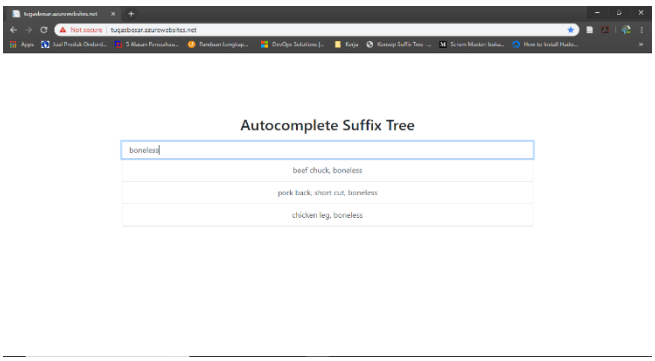
Gambar 17 merupakan sebuah grafik yang menggambarkan *progress* perjalanan *sprint backlog* dari awal proyek dimulai hingga akhir proyek selesai. Grafik yang menurun mengartikan semakin sedikitnya estimasi sisa waktu dalam pembuatan sistem *autocomplete* ini.

### *D. Halaman Pencarian*

Gambar 18 adalah halaman sistem *autocomplete suffix tree*, yang dapat diakses melalui http://tugasbesar.azurewebsites.net/. Gambar 19 merupakan halaman sistem *autocomplete suffix tree*, yang memunculkan *suggestion autocomplete* hasil dari sistem.




Gambar 18. Tampilan *Autocomplete Suffix Tree* pada halaman web




Gambar 19. Tampilan hasil pencarian boneless pada halaman web *Autocomplete Suffix Tree*

## IV. HASIL EKSPERIMEN DAN EVALUASI

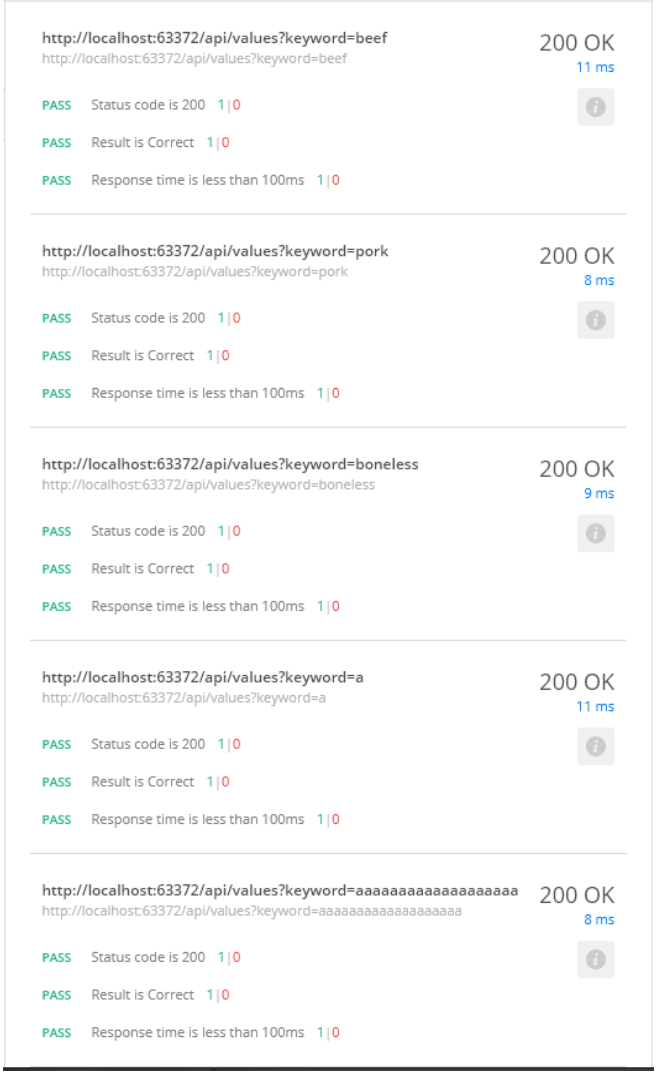
Pada tahap pengujian, pengujian dilakukan secara lokal dan memanfaatkan sebuah aplikasi *plug-in* yaitu *postman*. Tabel XI merupakan daftar *test case* yang akan diuji pada sistem *autocomplete suffix tree*. *Test case* tersebut dimasukan pada *postman*, dan *postman* akan melakukan tes langsung ke *webservice* pada sistem *autocomplete suffix tree*. *Test case* tersebut telah mewakili tes *normal* dan tes yang terberat. Contoh dengan input "a", pada data produk banyak yang memiliki huruf "a" maka sistem akan diuji kecepatan pencarian pada tabel data.

*p-ISSN : 2443-2210*
*e-ISSN : 2443-2229*

TABEL XI
PENGUJIAN DATA DAN RESPONSE TIME

| Uji Kasus | Input pengujian | Hasil Pengujian Data | *Response Time* |
|---|---|---|---|
| 1 | Beef | "beef chuck, boneless"<br>"beef Texas style burger"<br>"beef short loin"<br>"beef montreal smoked brisket"<br>"soup campbells, beef barley" | < 100ms |
| 2 | Pork | "pork hock and feet attached"<br>"pork belly fresh"<br>"pork back, short cut, boneless"<br>"pork inside"<br>"pork kidney" | < 100ms |
| 3 | Boneless | "beef chuck, boneless"<br>"pork back, short cut, boneless"<br>"chicken leg, boneless" | < 100ms |
| 4 | a | "cream of tartar"<br>"wine redchard merritt"<br>"rambutan"<br>"beans kidney white"<br>"table cloth 54x72 white" | < 100ms |
| 5 | aaaaaaaaaaaaaaaa | | < 100ms |

Pada Gambar 20 dibawah ini dapat dilihat bahwa ketika melakukan pengujian secara lokal didapatkan waktu *response* kurang dari 100 ms dan hasil pengujian data (*result correct*) yang sesuai (*pass*).




Gambar 20. Hasil pengujian data dan *response time* secara *collection*

## V. KESIMPULAN

Berdasarkan hasil penelitian dan evaluasi yang telah dilakukan, maka dapat disimpulkan sebagai berikut:

1. Penggunaan metode agile dengan mengaplikasikan *sprint backlog* dapat dilihat pada *burndown chart* pada penelitian ini. *Burndown chart* menggambarkan penurunan yang stabil dan penelitian ini diselesaikan tepat waktu sesuai dengan waktu yang telah diestimasikan sebelum penelitian ini dimulai.
2. Untuk *response time* dari suatu pencarian memerlukan rata-rata waktu kurang dari 100ms (Gambar 20).
3. Prototipe sistem *autocomplete* (Gambar 20) telah dapat menampilkan daftar 5 produk yang diurutkan berdasarkan produk paling laku untuk memberikan sugesti kepada konsumen dalam mencari produk.

## UCAPAN TERIMA KASIH

Terima kasih untuk *website Kaggle* yang telah menyediakan data untuk penelitian ini.

## DAFTAR PUSTAKA

[1] Irmawati, D., Pemanfaatan E-Commerce Dalam Dunia Bisnis. Jurnal Ilmiah Orasi Bisnis - ISSN: 2085-1375 Edisi Ke-IV, 95-112, 2011.

[2] Made Ari Lestari, N., Made Sudarma, Perencanaan Search Engine E-commerce dengan Metode Latent Semantic Indexing Berbasis Multiplatform. Lontar Komputer Vol. 8 No. 1, 2017

[3] David Ward, Jim Hahn, dan Kristen Feist, Autocomplete as a Research Tool: A Study on Providing Search Suggestions. Information Technology and Libraries, 2012.

[4] Rahim Aditrian, Penyusunan Overlap Graph Menggunakan Suffix Tree Pada DNA Sequence, Institut Pertanian Bogor, 2013.

[5] Peta Nurjana, Ernawati, dan Aan Erlansari, Implementasi Algoritma Linear Congruent Method dan Algoritma Suffix Tree Pada Aplikasi Casual Game Tebak Lagu. Jurnal Rekursif Vol. 5 No. 3, 2017.

[6] Anurag Sarkar, Abir Ghosh, Dr. Asoke Nath, MapReduce: A Comprehensive Study on Applications, Scope and Challenges. International Journal of Advance Research in Computer Science and Management Studies Vol. 3, Issue 7, 2015.

[7] Anisha P. Rodrigues, Niranjan N. Chiplunkar, Real-time Twitter Data Analysis Using Hadoop Ecosystem. Cogent Engineering, 2018.

[8] Microsoft (2012), "Web Services". [Online]. Tersedia: http://msdn.microsoft.com/enus/library/ms950421.aspx

[9] Hakim, M. R., Prototipe Sistem Informasi Akademik Berbasis Mobile Menggunakan Script Object Notation (JSON), Skripsi STIKOM Surabaya, 2012.

[10] (2019) What Is Net Core. [Online]. Tersedia: https://cynicaldeveloper.com/blog/what-is-net-core/

[11] Wulan, R. Pengembangan Konfigurasi Model Analisis Arsitektur Agile Pada Perusahaan Bisnis IT Online (Studi Kasus Lazada dan Bhinneka.com), Teknik Informatika Universitas Indraprasta PGRI, 2016.

[12] RapidAPI, Web service synonym. [Online]. Tersedia: https://similiarwords.p.rapidapi.com

[13] M.Miftakul Amin, Interoperabilitas perangkat lunak menggunakan RESTful web service, Teknik Komputer, Politeknik Negeri Sriwijaya, Palembang, 2018.

[14] (2019) Response Times: The 3 Important Limits. [Online]. Tersedia: https://www.nngroup.com/articles/response-times-3-important-limits/