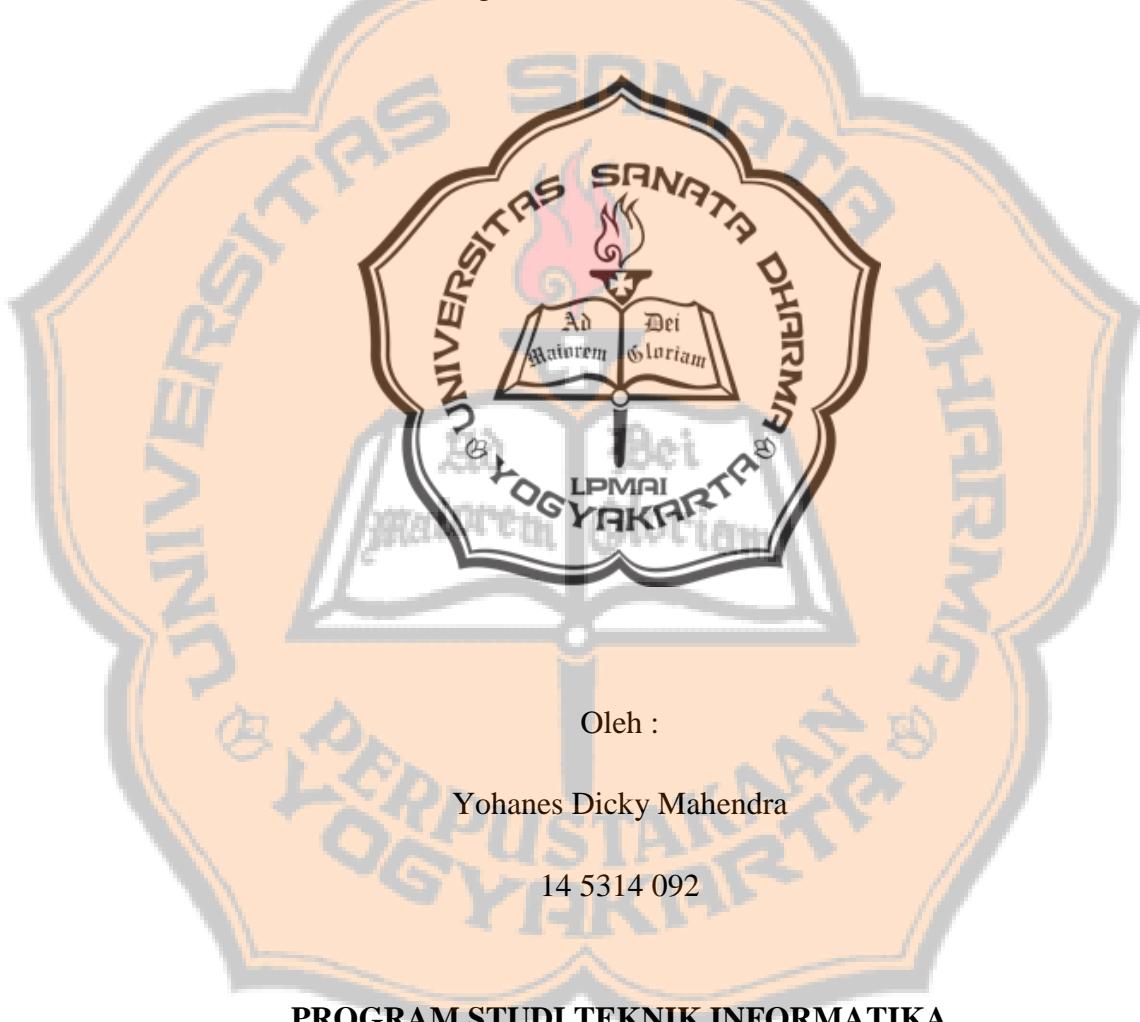


**SISTEM REKOMENDASI OBJEK WISATA YOGYAKARTA DENGAN
PENDEKATAN *ITEM-BASED COLLABORATIVE FILTERING***

SKRIPSI

Diajukan Untuk Memenuhi Salah Satu Syarat
Guna Memperoleh Gelar Sarjana Komputer (S.Kom)
Pada Program Studi Teknik Informatika



PROGRAM STUDI TEKNIK INFORMATIKA

FAKULTAS SAINS DAN TEKNOLOGI

UNIVERSITAS SANATA DHARMA

YOGYAKARTA

2018

**YOGYAKARTA TOURISM RECOMMENDATION SYSTEM USING
ITEM-BASED COLLABORATIVE FILTERING APPROACH**

THESIS

Present as Partial Fulfillment of Requirement

To Obtain the *Sarjana komputer* Degree

In Informatics Engineering Study Program



By :

Yohanes Dicky Mahendra

14 5314 092

INFORMATICTS ENGINEERING STUDY PRROGRAM

FACULTY OF SCIENCE AND TECHNOLOGY

SANATA DHARMA UNIVERSITY

YOGYAKARTA

2018

HALAMAN PERSETUJUAN

SKRIPSI

**SISTEM REKOMENDASI OBJEK WISATA YOGYAKARTA DENGAN
PENDEKATAN *ITEM-BASED COLLABORATIVE FILTERING***



Robertus Adi Nugroho S.T., M.Eng.

Tanggal : 17 Januari 2019

HALAMAN PENGESAHAN

SKRIPSI

SISTEM REKOMENDASI OBJEK WISATA YOGYAKARTA DENGAN PENDEKATAN *ITEM-BASED COLLABORATIVE FILTERING*

Disiapkan dan ditulis oleh :

Yohanes Dicky Mahendra

NIM : 145314092

Telah dipertahankan di depan panitia pengaji

Pada tanggal 24 Januari 2019

dan dinyatakan memenuhi syarat

Susunan Panitia Pengaji

Nama Lengkap

Ketua P. H. Prima Rosa, S.Si., M.Sc.

Sekretaris Dr. Cyprianus Kuntoro Adi, S.J. M.A., M.Sc.

Anggota Agnes Maria Polina, S.Kom., M.Sc.

Anggota Robertus Adi Nugroho, S.T., M.Eng.

Tanda Tangan



Yogyakarta, 29 Januari 2019

Fakultas Sains dan Teknologi

Universitas Sanata Dharma

Dekan



Sudi Mungkasi, S.Si., M.Mat., Ph.D

HALAMAN PERSEMBAHAN

“Focus, Consistence, Success”



Tugas akhir ini saya persembahkan kepada :

Tuhan yang Maha Esa

Kedua orang tua tercinta

Keluarga besar tercinta

Teman-teman seperjuangan Teknik Informatika 2014

PERNYATAAN KEASLIAN KARYA

Saya menyatakan dengan sesungguhnya bahwa skripsi yang saya tulis tidak memuat karya atau bagian orang lain, kecuali yang telah disebutkan dalam kutipan daftar pustaka selayaknya karya ilmiah.

Yogyakarta, 24 Januari 2019

Penulis,


Yohanes Dicky Mahendra

LEMBAR PERNYATAAN PERSETUJUAN PUBLIKASI

Yang bertandatangan dibawah ini, saya mahasiswa Universitas Sanata Dharma:

Nama : Yohanes Dicky Mahendra

NIM : 14 5314 092

Demi pengembangan ilmu pengetahuan, saya memberikan kepada Perpustakaan Universitas Sanata Dharma karya ilmiah saya yang berjudul :

SISTEM REKOMENDASI OBJEK WISATA YOGYAKARTA DENGAN PENDEKATAN *ITEM-BASED COLLABORATIVE FILTERING*

Beserta perangkat yang diperlukan (bila ada). Dengan demikian saya memberikan kepada Perpustakaan Sanata Dharma hak untuk menyimpan, mengalihkan dalam bentuk media lain untuk kepentingan akademis tanpa perlu meminta ijin dari saya maupun memberikan royalti kepada saya selama tetap mencantumkan nama saya sebagai penulis.

Demikian pernyataan ini saya buat dengan sebenarnya

Dibuat di Yogyakarta

Pada tanggal 24 Januari 2019

Yang menyatakan,

Yohanes Dicky Mahendra



ABSTRAK

Sistem rekomendasi adalah sistem yang bertanggung jawab atas mesin rekomendasi yang mampu mengidentifikasi serta memberikan konten berpotensi besar dipilih oleh pengguna berdasarkan penyaringan informasi yang mengambil preferensi dari perilaku maupun riwayat pengguna. Metode *collaborative filtering* merupakan salah satu metode dalam sistem rekomendasi, yang bekerja dengan menyaring informasi dari profil pengguna lain berupa *rating* untuk memprediksi *item* yang mungkin disukai pengguna. Metode *Collaborative Filtering (CF)* mempunyai 2 metode umum, yaitu *item-based cf* dan *user-based cf*.

Pada penelitian ini, penulis membangun sistem yang dapat merekomendasikan *item* berupa objek wisata kepada pengguna dan menghitung keakuratan sistem dalam menghitung prediksi *rating*. Sistem dibangun dengan menggunakan algoritma *item-based cf* dan menggunakan perhitungan *similarity pearson correlation*. Skenario perhitungan prediksi dilakukan dengan mengubah-ubah maksimum *neighbor*. Kesalahan dari hasil prediksi *rating* terhadap *rating* sesungguhnya dihitung menggunakan *Mean Absolute Error (MAE)*. Dataset yang digunakan berasal dari survey berupa *rating* 100 user terhadap 10 objek wisata.

Hasil akhir dari penelitian menunjukkan bahwa metode *item-based cf* dapat digunakan untuk membangun sistem rekomendasi objek wisata dan cukup akurat dalam memprediksi *rating* objek wisata. Hal ini dibuktikan dengan hasil uji MAE, dimana pada maksimum *neighbor* 6 metode *item-based cf* dapat memprediksi *rating* paling baik dengan MAE sebesar 0.6254.

Kata kunci : *Collaborative Filtering; Item-Based Collaborative Filtering; MAE; User-Based Collaborative Filtering; Neighbor; Rating; Sistem Rekomendasi;*

ABSTRACT

Recommendation system is a system that is responsible for recommendation engines that are able to identify and provide potentially large content chosen by users based on information filtering that takes preferences from user behavior and history. Collaborative Filtering method is one of the methods in the recommendation system, which works by filtering information from other user profiles in the form of ratings to predict items that users might like. Collaborative Filtering (CF) method has 2 general methods, namely Item-Based CF and User-Based CF.

In this research, the authors build a system that can recommend items in the form of attractions to users and calculate the accuracy of the system in calculating rating predictions. The system is built using the Item-Based CF algorithm and uses the Pearson Correlation similarity calculation. The scenario for calculating predictions is done by changing the maximum of the neighbors. Errors from the rating prediction of the true rating actually calculated using the Mean Absolute Error (MAE). Dataset that used is derived from a survey in the form of a rating of 100 users towards 10 tourist attractions.

The final results of the research showed that the Based-Item CF method can be used to build Attraction recommendation system and fairly accurate in predicting Attraction rating. This is evidenced by the results of the MAE test, where in the maximum neighbor 6 the Item-Based CF method can predict the best rating with MAE of 0.6254.

Keywords: Collaborative Filtering; Item-Based Collaborative Filtering; MAE; User-Based Collaborative Filtering; Neighbor; Rating; Recommendation System;

KATA PENGANTAR

Puji dan syukur penulis panjatkan ke hadirat Tuhan Yang Maha Esa atas berkat dan rahmat-Nya sehingga penulis dapat menyelesaikan tugas akhir dengan judul **“SISTEM REKOMENDASI OBJEK WISATA YOGYAKARTA DENGAN PENDEKATAN ITEM-BASED COLLABORATIVE FILTERING”**

Ucapan terimakasih sebesar-bersarnya penulis ucapkan kepada yang terhormat Ibu Agnes Maria Polina, S.Kom., M.Sc. selaku pembimping pertama dan Bapak Robertus Adi Nugroho, S.T., M.Eng. selaku pembimping kedua, yang dengan penuh kesabaran dan selalu meluangkan waktu dan memberikan pengarahan kepada penulis dalam menyelesaikan tugas akhir ini.

Penulis sadar, dalam menyelesaikan tugas akhir ini tidak lepas dari dukungan, bantuan dan doa dari berbagai pihak. Untuk itu penulis dengan tulus ingin mengucapkan terimakasih kepada :

1. Kedua orang tua penulis yang telah memberikan motivasi, doa dan perhatian kepada penulis.
2. Bapak Sudi Mungkasi, S.Si., M.Math.Sc., Ph.D. selaku Dekan Fakultas Sains dan Teknologi Universitas Sanata Dharma.
3. Ibu Dr. Anastasia Rita Widiarti, S.Si., M.Kom. selaku Ketua Program Studi teknik Informatika Universitas Sanata Dharma.
4. Bapak Henricus Agung Hernawan, S.T., M.Kom. selaku Dosen Pembimbing Akademik.
5. Seluruh dosen Teknik Informatika Universitas Sanata Dharma yang telah memberikan ilmu, mengajarkan pengetahuan dan pengalaman selama proses perkuliahan.
6. Teman-teman penulis, antara lain:
 - Lobak Family yang selalu memberikan semangat dan motivasi.
 - Teman seperantauan kalimantan barat.
 - Teman seperjuangan Teknik Informatika 2014 yang telah berdinamika bersama selama kuliah serta memberikan kesan dan pengalaman kepada penulis.

Penulis berharap penelitian ini dapat membantu dan berguna bagi pembaca. Penulis menyadari penelitian ini tidak sepenuhnya sempurna, oleh karena itu penulis mengharapkan kritik dan saran agar penelitian ini dapat berkembang dan dapat menjadi penelitian yang lebih baik untuk kedepan.

Yogyakarta, 24 Januari 2019

Penulis,



Yohanes Dicky Mahendra



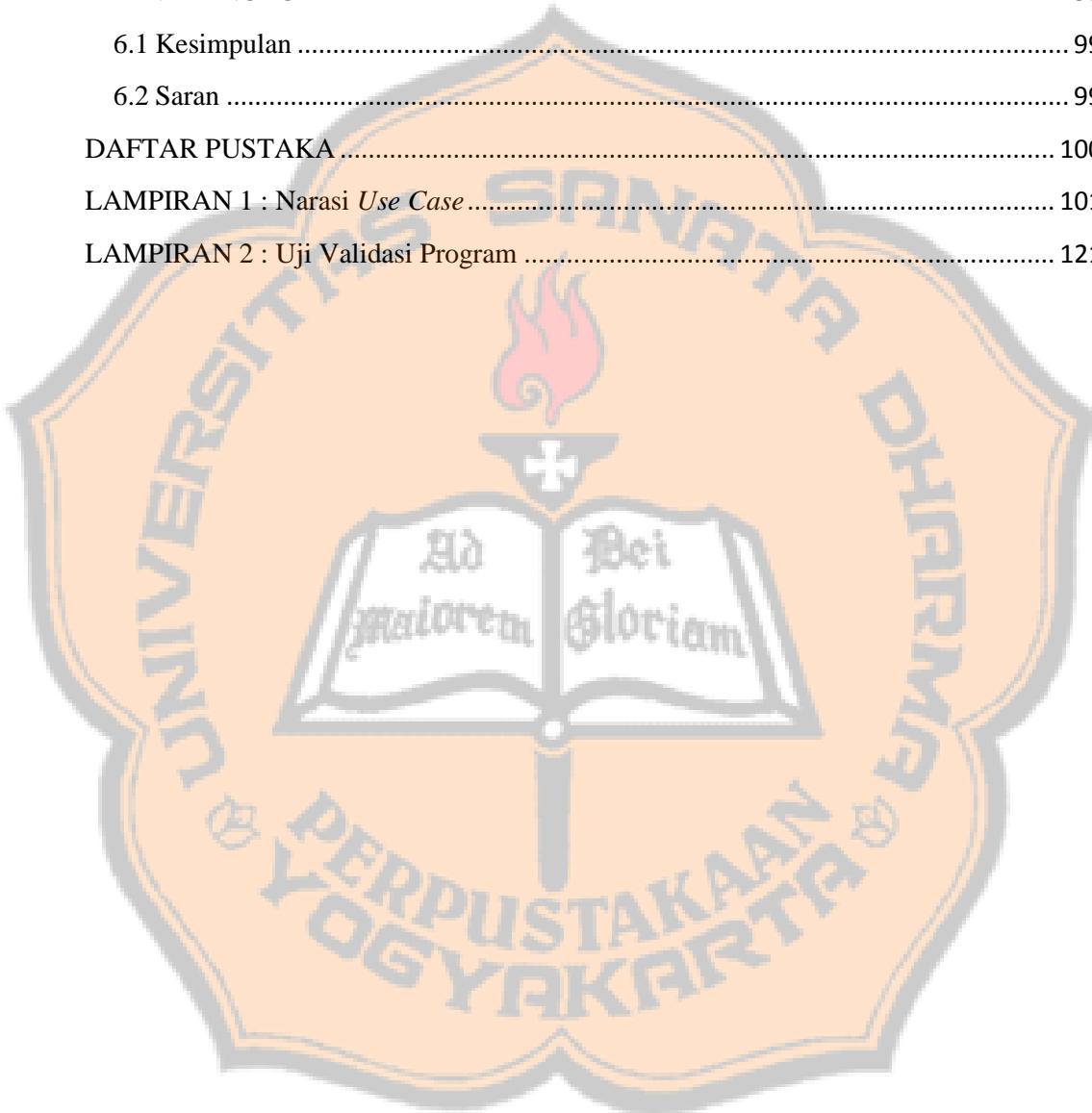
DAFTAR ISI

HALAMAN PERSETUJUAN.....	Error! Bookmark not defined.
HALAMAN PENGESAHAN.....	Error! Bookmark not defined.
HALAMAN PERSEMBAHAN	iv
PERNYATAAN KEASLIAN KARYA	Error! Bookmark not defined.
LEMBAR PERNYATAAN PERSETUJUAN PUBLIKASI	Error! Bookmark not defined.
ABSTRAK.....	viii
ABSTRACT.....	ix
KATA PENGANTAR	x
DAFTAR ISI.....	xii
DAFTAR GAMBAR.....	xvi
DAFTAR TABLE.....	xx
DAFTAR RUMUS	xxi
DAFTAR QUERY	xxii
BAB I PENDAHULUAN	1
1.1 Latar Belakang	1
1.2 Rumusan Masalah	2
1.3 Batasan Masalah	2
1.4 Tujuan Penelitian	3
1.5 Manfaat Penelitian	3
1.6 Sistematika Penulisan	3
BAB II LANDASAN TEORI	5
2.1 Sistem Rekomendasi	5
2.1.1 <i>Collaborative Filtering</i>	5
2.1.1.1 <i>Item-Based Collaborative Filtering</i>	7
2.1.1.2 Alur Penetapan Metode <i>Collaborative Filtering</i>	8
2.1.1.3 <i>Pearson Correlation</i>	9
2.1.1.4 Perhitungan Prediksi	9
2.1.1.5 Perhitungan <i>Mean Absolute Error</i> (MAE).....	10
2.2 <i>Content Based Filtering</i>	11
BAB III METODOLOGI PENELITIAN	13
3.1 Alur Penelitian	13

3.2 Tahapan Penelitian	14
3.2.1 Survey Pendahuluan.....	14
3.2.2 Studi Pustaka.....	14
3.2.3 Pengambilan Data	15
3.2.4 Perancangan dan Implementasi Sistem.....	17
3.2.5 Pengujian Sistem.....	18
3.2.6 Analisa Hasil	18
3.2.7 Penarikan Kesimpulan dan Saran.....	18
3.3 Spesifikasi <i>Hardware</i> dan <i>Software</i>	18
3.3.1 <i>Hardware</i>	18
3.3.2 <i>Software</i>	19
BAB IV ANALISA DAN PERANCANGAN SISTEM.....	19
4.1 Analisa Sistem.....	19
4.1.1 Diagram Konteks	19
4.1.2 Use Case Diagram	20
4.1.3 Narasi Use Case	22
4.2 Perancangan <i>Input</i>	22
4.3 Perancangan Basisdata.....	23
4.3.1 Perancangan Basisdata secara Konseptual	23
4.3.2 Desain Basisdata secara Logikal	24
4.3.3 Desain Basisdata secara Fisikal	24
4.4 Perancangan Proses	27
4.4.1 Algoritma Blok Proses	28
4.4.2 Contoh Perhitungan Metode	45
4.4.2.1 Metode <i>Item-Based CF</i>	45
4.4.2.2 Perhitungan <i>Mean Absolute Error</i> (MAE).....	51
4.4.3 Diagram Kelas Desain.....	52
4.4.3.1 Kelas <i>View</i>	52
4.4.3.1.1 Kelas <i>View Admin</i>	53
4.4.3.1.2 Kelas <i>View User</i>	54
4.4.3.1.3 Kelas <i>View Admin</i> dan <i>User</i>	54
4.4.3.2 Kelas <i>Control</i>	55
4.4.3.3 Kelas <i>Model</i>	56

4.5 Perancangan <i>Output</i>	60
4.5.1 Interface Aktor <i>User</i>	61
4.1.1.1 Halaman <i>Login</i>	61
4.1.1.2 Halaman Utama	61
4.1.1.3 Halaman Hasil Prediksi	62
4.1.1.4 Halaman <i>Update Data</i>	63
4.5.2 Interface Aktor <i>Admin</i>	64
4.5.2.1 Halaman <i>Login Admin</i>	64
4.5.2.2 Halaman Utama	64
4.5.2.3 Halaman Tambah <i>Admin</i>	65
4.5.2.4 Halaman Edit <i>Admin</i>	65
4.5.2.5 Halaman Tambah <i>User</i>	66
4.5.2.6 Halaman <i>Edit User</i>	66
4.5.2.7 Halaman Tambah Objek Wisata	67
4.5.2.8 Halaman <i>Edit Objek Wisata</i>	67
4.5.2.9 Halaman Hitung <i>Similarity</i>	68
4.5.2.10 Halaman Hitung Prediksi	69
4.5.3 Interface untuk Aktor <i>User</i> dan <i>Admin</i>	70
3.4.3.1 Halaman <i>Signup</i>	70
3.4.3.2 Halaman <i>Rating</i>	71
BAB V IMPLEMENTASI DAN ANALISA HASIL	72
5.1 Implementasi Basisdata	72
5.1.1 Tabel <i>user</i>	72
5.1.2 Tabel <i>admin</i>	72
5.1.3 Tabel <i>objek_wisata</i>	72
5.1.4 Tabel <i>rating</i>	73
5.1.5 Tabel <i>similarity</i>	73
5.1.6 Tabel <i>rating_prediksi</i>	73
5.2 Implementasi Proses	74
5.2.1 Blok Proses pada Kelas	74
5.2.2 Implementasi Blok Proses dalam Koding	74
5.3 Implementasi <i>Output</i>	87
5.3.1 Aktor <i>User</i>	87

5.3.2 Aktor <i>Admin</i>	90
5.3.3 Aktor <i>Admin</i> dan <i>User</i>	96
5.4 Analisis Hasil dan Pembahasan	97
5.4.1 Hasil Uji Validasi Program	97
5.4.2 Hasil Uji MAE	97
BAB VI PENUTUP	99
6.1 Kesimpulan	99
6.2 Saran	99
DAFTAR PUSTAKA	100
LAMPIRAN 1 : Narasi <i>Use Case</i>	101
LAMPIRAN 2 : Uji Validasi Program	121



DAFTAR GAMBAR

Gambar 2.1 Konsep item-based collaborative filtering (Ricci et.al., 2011)	7
Gambar 2.2 Konsep <i>content-based filtering</i>	11
Gambar 3.1 Diagram Alur Penelitian	13
Gambar 3.2 Tahapan perhitungan prediksi	17
Gambar 4.1 Diagram Konteks.....	19
Gambar 4.2 Use case diagram.....	20
Gambar 4.3 Data <i>rating</i> pada <i>database</i>	21
Gambar 4.4 Model konseptual (ER Diagram).....	22
Gambar 4.5 Model logikal	23
Gambar 4.6 <i>Flowchart</i> perancangan proses pada sistem	26
Gambar 4.7 Diagram uml Rating.java	28
Gambar 4.8 Diagram uml kelas ItemBased.java.....	31
Gambar 4.9 Diagram uml kelas Prediksi.java.....	38
Gambar 4.10 Diagram uml kelas MAE.java.....	43
Gambar 4.11 Diagram kelas desain	51
Gambar 4.12 Halaman login pada sistem	60
Gambar 4.13 Halaman utama aktor <i>user</i> pada sistem.....	61
Gambar 4.14 Halaman hasil prediksi pada sistem	61
Gambar 4.15 Halaman hasil prediksi dengan <i>popup</i> pada sistem.....	62
Gambar 4.16 Halaman <i>update</i> data pada sistem	62
Gambar 4.17 Halaman login <i>admin</i> pada sistem	63
Gambar 4.18 Halaman utama aktor <i>admin</i> pada sistem.....	63
Gambar 4.19 Halaman tambah <i>admin</i> pada sistem	64
Gambar 4.20 Halaman <i>edit admin</i> pada sistem.....	64
Gambar 2.21 Halaman tambah <i>user</i> pada sistem	65
Gambar 4.22 Halaman <i>edit user</i> pada sistem.....	65
Gambar 4.23 Halaman tambah objek wisata di sistem	66
Gambar 4.24 Halaman edit objek wisata pada sistem.....	66
Gambar 4.25 Halaman hitung <i>similarity</i> yang <i>similarity</i> objek wisata belum dihitung	67

Gambar 4.26 Halaman hitung <i>similarity</i> yang <i>similarity</i> objek wisata setelah dihitung	67
Gambar 4.27 Halaman hitung prediksi pada sistem.....	68
Gambar 4.28 Halaman hitung prediksi dengan <i>popup</i>	68
Gambar 4.29 Halaman hitung prediksi hasil prediksi	69
Gambar 4.30 Rancangan halaman signup	69
Gambar 4.31 Rancangan halaman rating	70
Gambar 5.1 <i>Source code method</i> getTotalUserRated().....	73
Gambar 5.2 <i>Source code method</i> getUserId(String)	74
Gambar 5.3 <i>Source code method</i> inputDataRatingTesting(Rating).....	74
Gambar 5.4 <i>Source code method</i> pilihDataTesting()	74
Gambar 5.5 <i>Source code method</i> main(String[])	75
Gambar 5.6 Lama waktu <i>running</i> pemilihan data <i>testing</i>	75
Gambar 5.7 <i>Source code method</i> getRating(String)	76
Gambar 5.8 <i>Source code method</i> getAverage(String).....	76
Gambar 5.9 <i>Source code method</i> getAtas(List<Rating>, List<Rating>,double,double)	76
Gambar 5.10 <i>Source code method</i> getBawah(List<Rating>, List<Rating>,double,double)	77
Gambar 5.11 <i>Source code method</i> similaritasPearsonCorrelation (String,String)	77
Gambar 5.12 <i>Source code method</i> InputSimilarity (String,String,ItemBased)....	77
Gambar 5.13 <i>Source code method</i> similarity()	78
Gambar 5.14 <i>Source code method</i> main(String[])	78
Gambar 5.15 Lama waktu <i>running</i> perhitungan <i>similarity</i>	79
Gambar 5.16 Total data <i>similarity</i> yang tersimpan di <i>database</i>	79
Gambar 5.17 <i>Source code method</i> getRatingTraining(int)	80
Gambar 5.18 <i>Source code method</i> getRatingAndOWSimilaritySort (String,String)	80
Gambar 5.19 <i>Source code method</i> hasilPrediksiItemBased (Rating,int)	81
Gambar 5.20 <i>Source code method</i> inputPrediksi(Prediksi)	81
Gambar 5.21 <i>Source code method</i> prediksiRating(int)	82

Gambar 5.22 <i>Source code method main(String[])</i> untuk <i>running method prediksiRating(int)</i>	82
Gambar 5.23 <i>Running method prediksiRating(int)</i> dengan maksimum <i>neighbor 4</i>	83
Gambar 5.24 <i>Running method prediksiRating(int)</i> dengan maksimum <i>neighbor 6</i>	83
Gambar 5.25 <i>Running method prediksiRating(int)</i> dengan maksimum <i>neighbor 8</i>	83
Gambar 5.26 Total data prediksi pada tabel rating_prediksi	84
Gambar 5.27 <i>Source code method showPrediction(int)</i> untuk menghitung MAE.....	85
Gambar 5.28 Method main(String[])	85
Gambar 5.29 Hasil <i>running</i> dengan maksimum <i>neighbor 4</i>	85
Gambar 5.30 Hasil <i>running</i> dengan maksimum <i>neighbor 6</i>	86
Gambar 5.31 Hasil <i>running</i> dengan maksimum <i>neighbor 8</i>	86
Gambar 5.32 Halaman login <i>user</i>	86
Gambar 5.33 Halaman utama <i>user</i>	87
Gambar 5.34 Halaman Hasil Prediksi	87
Gambar 5.35 Menu <i>popup</i> pada halaman hasil prediksi.....	88
Gambar 5.36 Halaman <i>update</i> data.....	88
Gambar 5.37 Halaman login <i>admin</i>	89
Gambar 5.38 Halaman utama admin.....	89
Gambar 5.39 Halaman tambah <i>admin</i>	90
Gambar 5.40 Halaman <i>edit admin</i>	90
Gambar 5.41 Halaman tambah <i>user</i>	91
Gambar 5.42 Halaman <i>edit user</i>	91
Gambar 5.43 Halaman input objek wisata	92
Gambar 5.44 Halaman <i>edit objek wisata</i>	92
Gambar 4.45 Halaman hitung <i>similarity</i>	93
Gambar 4.46 Halaman hitung prediksi yang masih kosong	93
Gambar 4.47 Menu <i>popup</i> pada halaman hitung prediksi	94
Gambar 4.48 Halaman hitung prediksi dengan tampilan hasil prediksi <i>rating</i>	

dari data <i>testing</i> dan prediksi <i>rating</i> objek wisata yang belum di- <i>rating user</i>	94
Gambar 4.49 Halaman <i>signup</i>	95
Gambar 5.50 Halaman <i>rating</i>	95
Gambar 5.51 Grafik garis hasil perhitungan MAE terhadap jumlah maksimum <i>neighbor</i>	97
Gambar 5.52 Grafik waktu <i>running MAE</i>	97
Gambar 5.56 Nama objek wisata dengan id 1.....	120
Gambar 5.57 Nama objek wisata dengan id 8.....	120
Gambar 5.58 Hasil <i>running query</i> objek wisata 1.....	121
Gambar 5.59 Hasil <i>running query</i> objek wisata 4.....	121
Gambar 5.60 Hasil eksekusi <i>query</i> mencari interseksi <i>rating</i>	122
Gambar 5.61 Menghitung <i>similarity</i> pada menu hitung <i>similarity</i>	126
Gambar 5.62 Hasil perhitungan <i>similarity</i> pada tabel <i>similarity</i>	126
Gambar 5.63 Sampel data <i>testing</i> untuk prediksi.....	127
Gambar 5.64 Hasil eksekusi <i>query</i> untuk mengetahui nama <i>user</i> 5 dan objek wisata 5.....	127
Gambar 5.65 <i>Rating</i> dan nilai <i>similarity</i> 6 <i>neighbor</i>	128
Gambar 5.66 Menghitung prediksi dengan maksimum <i>neighbor</i> 6 pada sistem website.....	130
Gambar 5.67 Prediksi <i>rating</i> untuk objek wisata 4 dan <i>user</i> 5	130

DAFTAR TABLE

Table 3.1 Matrix <i>user-item</i>	14
Tabel 3.2 Matrix <i>user-item</i> data survey	15
Tabel 4.1 Rancangan implementasi blok proses pada kelas .java	27
Tabel 4.2 Data <i>rating</i> untuk contoh perhitungan metode <i>Item-Based CF</i>	45
Tabel 4.3 Data interseksi <i>rating</i> antara Museum TNI Dirgantara Mandala dengan Monumen Jogja Kembali.....	46
Tabel 4.4 <i>Matrix similarity</i> objek wisata dari hasil perhitungan <i>similarity</i> antar objek wisata.....	47
Tabel 4.5 Contoh data <i>testing</i>	48
Tabel 5.1 Implementasi blok proses pada kelas java	73
Tabel 5.2 Hasil perhitungan MAE dengan skenario penggantian maksimum neighbor.....	96
Tabel 5.3 Hasil eksekusi <i>query</i> mencari interseksi <i>rating</i>	123

DAFTAR RUMUS

Rumus Pearson Correlation Item-Based CF (2.1)	9
Rumus Perhitungan Prediksi (2.2)	10
Rumus perhitungan MAE (3.1).....	10



DAFTAR QUERY

<i>Query 5.1 Mengambil nilai rata-rata objek wisata.....</i>	121
<i>Query 5.2 Mengambil interseksi rating objek wisata</i>	122
<i>Query 5.3 Query untuk mendapat nilai similarity objek wisata 4 dengan objek wisata lainnya.....</i>	128



BAB I PENDAHULUAN

1.1 Latar Belakang

Pariwisata pada saat ini telah menjadi salah satu kebutuhan manusia.

Pada hakikatnya pariwisata atau turisme adalah suatu perjalanan yang dilakukan untuk rekreasi atau liburan dan juga persiapan yang dilakukan untuk aktivitas pariwisata itu. Aktifitas pariwisata didorong oleh berbagai kepentingan, baik kepentingan ekonomi, sosial, budaya, agama, menambah pengalaman atau pun untuk belajar. Pariwisata berhubungan erat dengan perjalanan pariwisata, yaitu kegiatan berpindah dari suatu tempat ke tempat lainnya dengan tujuan untuk menikmati objek dan daya tarik wisata.

Pada saat ini industri pariwisata di Yogyakarta berkembang pesat, terbukti dengan banyaknya wisatawan mancanegara yang berkunjung dan juga banyaknya objek wisata baru. Objek wisata mencakup berbagai macam kategori wisata, seperti wisata alam, wisata budaya, wisata museum, dan wisata kebun binatang. Objek wisata yang baru muncul tersebut bisanya menjadi tren baik di sosial media maupun media cetak untuk saat itu, namun kemudian muncul objek wisata yang lebih baru dan lebih populer dan wisatawan mulai meninggalkan objek wisata yang lama. Hal ini menyebabkan wisatawan berwisata ke sebuah objek wisata berdasarkan tren saja dan tidak berdasarkan ketertarikannya terhadap objek wisata. Oleh karena itu dibutuhkan suatu sistem yang bisa memberikan rekomendasi alternatif objek wisata.

Sistem rekomendasi adalah sistem yang bertanggung jawab atas mesin rekomendasi yang mampu mengidentifikasi serta memberikan konten berpotensi besar dipilih oleh pengguna berdasarkan penyaringan informasi yang mengambil preferensi dari perilaku maupun riwayat pengguna (Asanov, 2015). Sistem rekomendasi dapat diterapkan dalam rekomendasi objek wisata. Sistem rekomendasi objek wisata ini menggunakan metode *item-based collaborative filtering*, yaitu menghitung *similarity* (kedekatan) *item* yang di-rating oleh pengguna. Sistem rekomendasi objek wisata ini

diharapkan dapat membantu wisatawan dalam memilih objek wisata yang akan mereka kunjungi.

Dua pendekatan yang umum digunakan dalam *collaborative filtering* yaitu *item-based collaborative filtering* dan *user-based collaborative filtering* (Ricci et.al., 2011). *User-based collaborative filtering* berasumsi bahwa cara yang baik dalam menemukan konten yang dirasa akan disukai oleh seorang *user* adalah dengan menemukan *user* lain dengan ketertarikan yang sama dengan *user* tersebut, kemudian merekomendasikan hal yang disukai oleh *user* lain kepada *user* tersebut. *Item-based collaborative filtering* berasumsi bahwa cara terbaik untuk memberikan rekomendasi kepada seorang *user* adalah dengan melihat pola pemberian *rating* terhadap sebuah item, dan mencoba memprediksi *rating* yang akan diberikan *user* terhadap item lain.

Penulis menerapkan *item-based collaborative filtering* dalam membangun sistem rekomendasi. Penulis memilih metode *item-based collaborative filtering* karena menurut penelitian yang telah dilakukan sebelumnya, komputasi *item-based collaborative filtering* diketahui lebih ringan dibandingkan dengan *user-based collaborative filtering* (Dwicahya, 2018). *Item* yang menjadi objek rekomendasi adalah objek wisata yang telah di-*rating* oleh wisatawan.

1.2 Rumusan Masalah

Dari latar belakang tersebut dapat dirumuskan masalah sebagai berikut :

1. Bagaimana membangun sistem rekomendasi objek wisata dengan metode *item-based collaborative filtering* ?
2. Seberapa akurat metode *item-based collaborative filtering* dapat memberikan rekomendasi objek wisata ?

1.3 Batasan Masalah

Batasan masalah dari penelitian ini adalah :

1. Objek dalam penelitian ini adalah rekomendasi objek wisata di Yogyakarta bagi wisatawan.

2. Objek wisata yang digunakan untuk sampel penelitian berjumlah 10 objek wisata
3. Metode yang digunakan adalah metode *item-based collaborative filtering*.
4. Input yang digunakan adalah dataset *rating user* terhadap objek wisata di Yogyakarta.
5. Skenario jumlah *maximum neighbor* yang digunakan adalah 4, 6 dan 8 *neighbor*.
6. Tidak memperhatikan *cold start problem*.
7. Nilai rating pada data set yang digunakan adalah (1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5 dan 5).

1.4 Tujuan Penelitian

Tujuan dari penelitian ini adalah :

1. Membangun sistem rekomendasi dengan menggunakan metode *item-based collaborative filtering*.
2. Mengetahui keakuratan *item-based collaborative filtering* dalam memberikan rekomendasi objek wisata yang baik.

1.5 Manfaat Penelitian

Manfaat yang bisa didapat dari penelitian ini adalah :

1. Mempermudah wisatawan di Yogyakarta dalam memilih objek wisata.
2. Mempermudah wisatawan di Yogyakarta dalam menentukan tren wisata.

1.6 Sistematika Penulisan

Sistem penulisan penelitian ini terdiri dari beberapa bab yang masing-masing bab mempunyai uraian pokok permasalahan, secara garis besar uraian tiap bab adalah sebagai berikut.

1. BAB I PENDAHULUAN

Bab ini menguraikan latar belakang, rumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian dan sistematika penulisan.

2. BAB II LANDASAN TEORI

Bab ini menguraikan teori-teori yang digunakan penulis dalam penelitian, meliputi sistem rekomendasi, *Item-Based Collaborative Filtering* dan alur penerapan metode seperti perhitungan *similarity pearson correlation* dan perhitungan prediksi.

3. BAB III METODOLOGI PENELITIAN

Bab ini menguraikan tahapan-tahapan penelitian yang dilakukan penulis dari tahap survey pendahuluan sampai pada tahap penarikan kesimpulan dan saran.

4. BAB IV ANALISA DAN PERANCANGAN SISTEM

Bab ini menguraikan analisa dan perancangan sistem yang dibangun, meliputi perancangan input, perancangan basisdata, perancangan proses dan perancangan output.

5. BAB V IMPLEMENTASI DAN ANALISA HASIL

Bab ini menguraikan implementasi dari perancangan pada bab 4 dan menganalisis hasil dari implementasi tersebut.

6. BAB VI PENUTUP

Bab ini menguraikan kesimpulan dari hasil penelitian untuk menjawab rumusan masalah pada bab 2, serta saran penulis untuk penelitian selanjutnya.

BAB II LANDASAN TEORI

2.1 Sistem Rekomendasi

Sistem rekomendasi adalah sebuah sistem yang dirancang dengan tujuan untuk membantu pengguna dalam mencari sesuatu yang mungkin mereka sukai dengan cara memberikan rekomendasi kepada pengguna. Rekomendasi itu berkaitan dengan berbagai proses pengambilan keputusan, seperti barang apa yang harus dibeli, musik apa yang didengarkan, atau berita online apa yang harus dibaca (Ricci et.al., 2011). Cara pencarian *item* yang akan direkomendasikan dapat dilakukan berdasarkan kemiripan baik berupa kemiripan suatu *item* dengan *item* lainnya berdasarkan konten atau kemiripan selera suatu pengguna dengan pengguna lain berdasarkan *rating* yang diberikan pada *item* (A. Djamal et.al., 2010).

Pada tahun awal perkembangan *internet*, banyak riset tentang sistem rekomendasi dilakukan untuk menemukan pendekatan-pendekatan baru untuk mengatasi masalah banyaknya informasi yang tersedia di *Internet*. Pendekatan sistem rekomendasi yang umum digunakan pada sistem rekomendasi adalah pendekatan *content-based filtering* dan *collaborative filtering*.

Terdapat beberapa penelitian terdahulu tentang sistem rekomendasi yang menjadi acuan penulis dalam penelitian ini, diantaranya penelitian tentang perbandingan sistem rekomendasi film *metode user-based cf* dan *item-based CF* (Dwicahya, 2018).

2.1.1 *Collaborative Filtering*

Collaborative filtering merupakan sebuah metode dalam membuat prediksi dengan cara menyaring informasi *item* dari opini orang lain. Ide utama dalam sistem rekomendasi *collaborative filtering* adalah untuk memanfaatkan riwayat opini pengguna aktif lain untuk

memprediksi item yang mungkin akan disukai/diminati oleh seorang pengguna. Implementasi yang paling sederhana dari pendekatan ini adalah membuat rekomendasi kepada pengguna aktif berdasarkan item yang disukai pengguna lain dengan riwayat selera yang serupa (Ricci et.al., 2011).

Sistem rekomendasi berbasis kolaboratif (*collaborative-based*) dibuat untuk mengatasi kelemahan dari sistem rekomendasi berbasis konten (*content-based*) (Adhitya pratama et.al., 2013) yaitu:

- a) Pendekatan *collaborative* dapat bekerja dalam domain dimana terdapat sedikit *content* yang berasosiasi dengan *item* atau ditempat dimana *content* sulit dianalisis menggunakan komputer seperti ide, masukkan, atau opini sehingga menjadi *reliable*.
- b) Pendekatan *collaborative* mempunyai kemampuan untuk menyediakan rekomendasi yang tidak terduga atau tidak disengaja, misalnya dapat merekomendasikan *item* yang relevan kepada pengguna sekaligus tidak mengandung *content* dari profil pengguna tersebut.

Walaupun dalam beberapa penelitian rekomendasi berbasis kolaboratif (*collaborative-based*) dapat menutupi kelemahan dari rekomendasi berbasis konten (*content-based*), rekomendasi berbasis kolaboratif (*collaborative-based*) memiliki kekurangan, antara lain (Adomavicius dan Tuzhilin, 2005):

1) *Cold-start problem*

Cold-start problem atau *new item problem* disebabkan karena *collaborative filtering* menggunakan *rating* atau preferensi pengguna untuk merekomendasikan sesuatu kepada pengguna lain. Ketika *rating* yang dibutuhkan tidak tersedia atau hingga *item* baru dinilai oleh sejumlah besar pengguna, maka sistem rekomendasi tidak akan dapat merekomendasikannya.

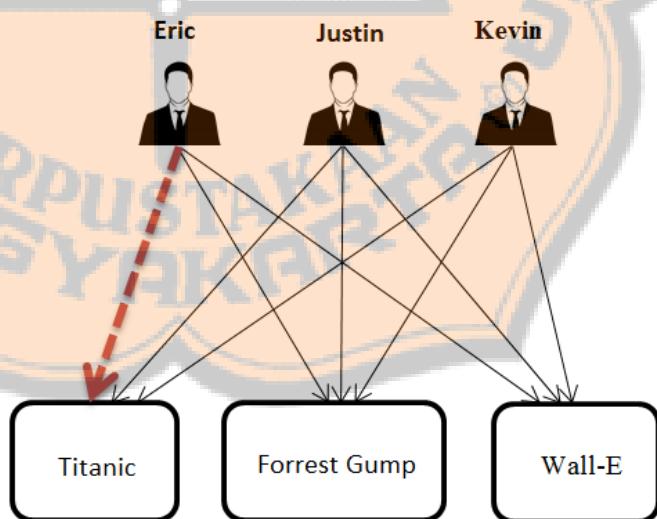
2) *Sparsity problem*

Pada data yang berukuran besar, jika banyak item baru yang sedikit di-*rating* oleh pengguna, maka *item* tersebut memiliki prediksi yang buruk dan menghasilkan rekomendasi yang buruk.

Terdapat dua metode atau pendekatan yang digunakan dalam *collaborative filtering* yaitu *user-based collaborative filtering* dan *item-based collaborative filtering*.

2.1.1.1 *Item-Based Collaborative Filtering*

Ide dari *item-based collaborative filtering* adalah mencari pola pemberian rating terhadap sebuah item dan kemudian mencoba memprediksi rating yang akan diberikan seorang pengguna terhadap item lain. Contohnya Eric menyukai film “Forrest Gump” dan “Wall-E”, namun dia belum menonton film “Titanic”. Maka Eric melihat bahwa Justin dan Kevin sudah memberikan peringkat yang sama pada film “Titanic” dengan dua film sebelumnya, maka Eric menyimpulkan bahwa dia juga akan menyukai film “Titanic” (Ricci et.al., 2011), seperti yang disajikan pada gambar (2.1).



Gambar 2.1 Konsep *item-based collaborative filtering*

2.1.1.2 Alur Penetapan Metode *Collaborative Filtering*

Metode *collaborative filtering* diterapkan untuk menghitung prediksi *rating* yang akan diberikan seorang pengguna kepada suatu *item*. Perediksi *rating* tersebut menggunakan dua pendekatan, baik *user-based collaborative filtering* maupun *item-based collaborative filtering*. Terdapat tiga alur utama dalam memprediksi *rating* (Ricci et.al., 2011), yaitu:

- 1) Menghitung nilai *similarity* antar *item* (*item-based*) atau *similarity* antar *user* (*user-based*).
- 2) Menentukan banyaknya *neighbor* yang memiliki nilai *similarity* terbesar dengan *item A* atau *user A* dengan melihat nilai similaritasnya. Dalam menentukan *neighbor*, jika terdapat dua atau lebih nilai *similarity* yang sama maka dilihat apakah *item* dengan *similarity* tersebut sudah di-*rating* oleh *user*. Nilai *similarity* yang dipakai adalah nilai *similarity item* yang sudah di-*rating* oleh *user*.
- 3) Memprediksi *rating* dari *user A* terhadap *item* tertentu dengan perhitungan yang melibatkan *neighbor* yang telah ditentukan.

Terdapat dua pendekatan dalam menghitung *similarity*, yaitu dengan menggunakan pendekatan *cosine similarity* dan pendekatan *pearson correlation*. Dalam tugas akhir ini dipergunakan pendekatan *pearson correlation*.

2.1.1.3 Pearson Correlation

Pendekatan *Pearson correlation* adalah sebuah metode yang dikembangkan oleh Karl Pearson. Korelasi (*correlation*) adalah sebuah teknik pengukuran yang menentukan seberapa dekat relasi antar dua himpunan bilangan yang berbeda. Dengan syarat himpunan bilangan tersebut harus memiliki urutan yang tetap dan berpasangan satu dengan lainnya antar kedua himpunan. Hasil pengukuran dapat berupa relasi positif ataupun relasi negatif. Relasi positif menunjukkan bahwa kedua himpunan memiliki kecenderungan kenaikan atau penambahan nilai yang sejajar. Sedangkan relasi negatif menunjukkan kedua himpunan memiliki kecenderungan penurunan atau pengurangan nilai yang sejajar. Sejajar dalam konteks ini berarti penurunan atau kenaikan nilai yang saling mengikuti antar kedua variabel tersebut. Salah satu teknik pengukuran korelasi adalah *Pearson Product Moment Correlation* atau biasa disingkat menjadi *Pearson Correlation* (Arvid et.al., 2016). Berikut ini adalah persamaan (2.1) yang digunakan untuk menghitung *similarity* antar *item* yang di-*rating* oleh *user*.

$$PC(i, j) = \frac{\sum_{u \in U_{ij}} (r_{ui} - \bar{r}_i)(r_{uj} - \bar{r}_j)}{\sqrt{\sum_{u \in U_{ij}} (r_{ui} - \bar{r}_i)^2 \cdot \sum_{u \in U_{ij}} (r_{uj} - \bar{r}_j)^2}} \quad (2.1)$$

Dimana :

- $PC(i, j)$ adalah nilai kemiripan (*similarity*) antara *item i* dan *item j*
- $r_{u,i}$ dan $r_{u,j}$ adalah *rating user* i dan j terhadap *item i*
- $\bar{r}_{u,i}$ dan $\bar{r}_{u,j}$ adalah rata-rata *rating item u* dan v

2.1.1.4 Perhitungan Prediksi

Perhitungan prediksi digunakan untuk memprediksi nilai *rating* yang diberikan oleh *user* untuk *item* tertentu. Proses prediksi dilakukan setelah mendapatkan *neighbor* dengan nilai similaritas tertinggi untuk proses prediksi. Proses prediksi merupakan proses akhir dalam *collaborative filtering* dalam memberikan suatu rekomendasi. Salah satu teknik yang digunakan untuk mendapatkan nilai prediksi adalah dengan persamaan *weighted sum* (Ricci et.al.,2011).

Berikut adalah rumus (2.2) untuk menghitung prediksi pada metode *item-based collaborative filtering*.

$$P(a,j) = \frac{\sum_{i \in I} (R_{a,i} \cdot S_{i,j})}{\sum_{i \in I} |S_{i,j}|} \quad (2.2)$$

Dimana :

- $P(a,j)$ adalah prediksi *rating* pada item *j* untuk *user a*
- $R_{a,i}$ adalah *rating* yang diberikan *user a* kepada item *i*
- $S_{i,j}$ adalah nilai *similarity* antara item *i* dan item *j*

2.1.1.5 Perhitungan *Mean Absolute Error* (MAE)

Mean absolute error (MAE) adalah formula yang digunakan untuk menghitung tingkat akurasi atau besar *error* hasil prediksi *rating* dari sistem terhadap *rating* yang sebenarnya yang *user* berikan terhadap suatu *item* (Ricci et.al., 2011). Dengan skenario *range rating* 1 – 5 maksimum MAE yang dihasilkan maksimum adalah 4. MAE diperoleh dengan menghitung *error* absolut dari N pasang *rating* asli dan prediksi, kemudian menghitung rata-ratanya. Dari MAE yang dihasilkan akan ditarik kesimpulan dengan asumsi jika MAE semakin mendekati 0 maka hasil prediksi akan semakin akurat. Rumus *mean absolute error* disajikan pada rumus (3.1).

$$\text{MAE} = \frac{\sum_{i=1}^n |p_i - q_i|}{N} \quad (3.1)$$

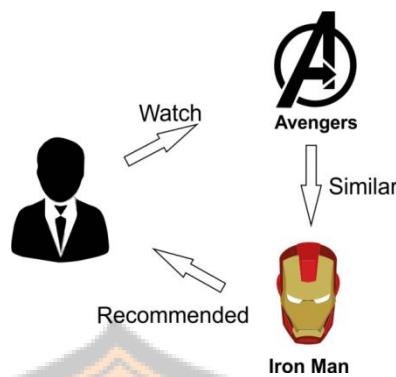
Dimana :

- p_i adalah *rating* yang diprediksi
- q_i adalah *rating* yang sebenarnya
- N adalah banyaknya pasang *rating* asli dan prediksi

2.2 Content Based Filtering

Pada *content-based filtering*, sistem belajar merekomendasikan *item* yang mirip dengan *item* yang disukai pengguna di masa lalu. Kesamaan item dihitung berdasarkan fitur yang terkait dengan *item* yang dibandingkan. Misalnya, jika pengguna telah memberi nilai positif pada film yang termasuk dalam genre *action*, maka sistem dapat belajar untuk merekomendasikan film lain dari genre ini.

Content-based filtering bekerja dengan profil pengguna yang ada. Profil memiliki informasi tentang pengguna dan ketertarikan mereka terhadap suatu *item*. Ketertarikan pengguna didasarkan pada peringkat pengguna untuk *item* yang berbeda. Umumnya, setiap kali pengguna membuat profilnya, mesin rekomendasi melakukan survei pengguna untuk mendapatkan informasi awal tentang pengguna untuk menghindari masalah *new user problem* atau *cold start problem*. Dalam proses rekomendasi, mesin membandingkan *item* yang sudah dinilai positif oleh pengguna dengan *item* yang tidak dinilainya dan mencari kesamaan. *Item* lain yang mirip dengan *item* yang berperingkat positif akan direkomendasikan kepada pengguna (Ricci et.al.,2011). Gambar 2.2 berikut menjelaskan algoritma *content-based filtering* dalam memberikan rekomendasi.



Gambar 2.2 Konsep *content-based filtering*

(Sumber: <https://www.quora.com/What-is-the-difference-between-content-based-filtering-and-collaborative-filtering> dengan modifikasi)

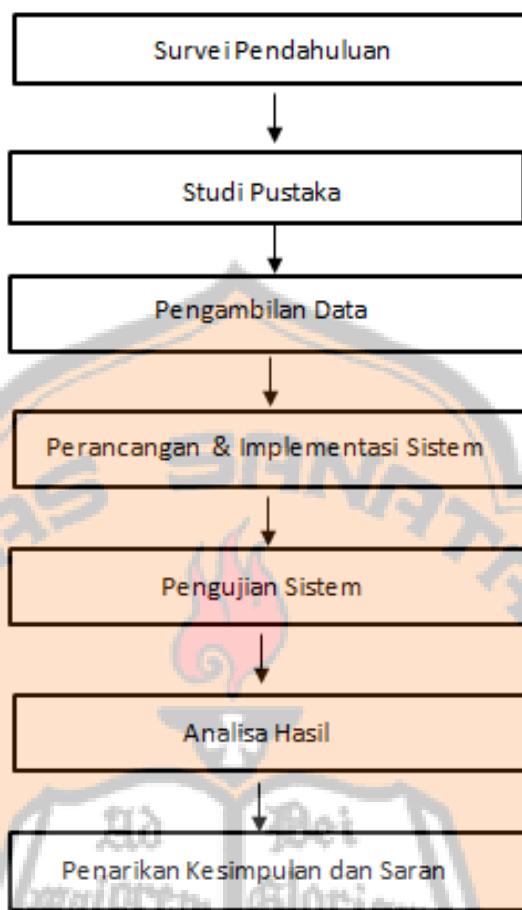
Perbedaan mendasar antara *content-based filtering* dengan *item-based collaborative filtering* yaitu pada cara pemberian rekomendasi kepada *user*. *Content-based filtering* memberi rekomendasi kepada *user* berdasarkan *history* ketertarikan *user* terhadap *item* tertentu dengan melihat beberapa variabel penentu, seperti genre pada film. Sedangkan *item-based collaborative filtering* memberikan rekomendasi berdasarkan kedekatan (*similarity*) rating *item-item* yang telah diberikan oleh *user*.

BAB III METODOLOGI PENELITIAN

Bab ini berisi tentang metodologi penelitian yang digunakan dalam penelitian dan pengembangan sistem. Berikut urutan langkah penelitian yang dilakukan penulis serta cara mengambil dan menganalisis data yang digunakan dalam penelitian.

3.1 Alur Penelitian

Alur penelitian yang dilakukan dalam penelitian ini terdiri dari 7 tahap yang digambarkan pada gambar 3.1 berikut ini.



Gambar 3.1 Diagram Alur Penelitian

3.2 Tahapan Penelitian

3.2.1 Survey Pendahuluan

Pada tahap ini, penulis melakukan observasi kepada wisatawan untuk mengetahui masalah yang dihadapi dalam mengambil keputusan tujuan wisata. Penulis melakukan observasi dengan bertanya langsung kepada wisatawan tentang bagaimana wisatawan memilih objek wisata yang sesuai dengan keinginannya.

3.2.2 Studi Pustaka

Penulis mencari sumber referensi yang terkait dengan topik penelitian seperti jurnal dan *thesis* selain itu juga referensi dari internet. Topik yang terkait dengan penelitian seperti kegunaan sistem rekomendasi, jenis-jenis algoritma rekomendasi dan perbandingan algoritma rekomendasi.

3.2.3 Pengambilan Data

Data yang diambil adalah data *rating* 10 objek wisata yang paling populer berdasarkan jumlah kunjungan terbanyak terhadap objek wisata yang dikunjungi oleh wisatawan berdasarkan buku statistik kepariwisataan DI Yogyakarta 2016. Nilai *rating* yang terdapat pada dataset yaitu (1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5 dan 5). Data set yang diambil kemudian dimasukkan ke dalam matrix *item rating* seperti pada table 3.1.

Table 3.1 Matrix *user-item*

	MTAU	MJK	TB	KRB	MBV	TS	KY	DMA	TP	CP
User 1	3.5	4	4	5	4	3.5	4.5	5	5	5
User 2	4	4	5	4	4.5	4	5	4	5	5
User 3	5	4	5	4	4	4	4.5	4	4	4
User 4	4	3.5	3.5	4	4	3.5	3.5	4	3	4

Keterangan :

- MTAU = Museum TNI AU Dirgantara Mandala
- MJK = Monumen Jogja Kembali
- TB = Tebing Breksi
- KRB = Kraton Ratu Boko
- MBV = Museum Benteng Vredeburg
- TS = Taman sari

- KY = Kraton Yogyakarta
- DMA = De Mata Art Museum
- TP = Taman Pintar
- CP = Candi Prambanan

Data set diambil dari survey yang dilakukan kepada wisatawan. Survey dilakukan dengan google form dan dengan survey secara langsung kepada wisatawan. Data set ini digunakan sebagai *input* untuk metode *item-based collaborative filtering*. Berikut merupakan data yang diperoleh dari survey kepada wisatawan, seperti yang disajikan pada tabel 3.2.

Tabel 3.2 Matrix *user-item* data survey

	MTAU	MJK	TB	KRB	MBV	TS	KY	DMA	TP	CP
Angelia Erlita	5.00	5.00	4.00	4.00	5.00	5.00	5.00	4.00		
Stevania E.M.S		3.50	4.00			3.50	4.00		3.00	4.50
Linda vixna	1.00		2.50	2.50	2.50	3.50	1.00	2.00		3.50
Sumitro		4.00		4.00	4.00	5.00	3.00	5.00	5.00	5.00
Ega	3.50	4.00	4.00	4.00	3.00	3.00	4.50	3.50	4.00	5.00

Keterangan :

- MTAU = Museum TNI AU Dirgantara Mandala
- MJK = Monumen Jogja Kembali
- TB = Tebing Breksi
- KRB = Kraton Ratu Boko
- MBV = Museum Benteng Vredeburg
- TS = Taman sari
- KY = Kraton Yogyakarta
- DMA = De Mata Art Museum
- TP = Taman Pintar

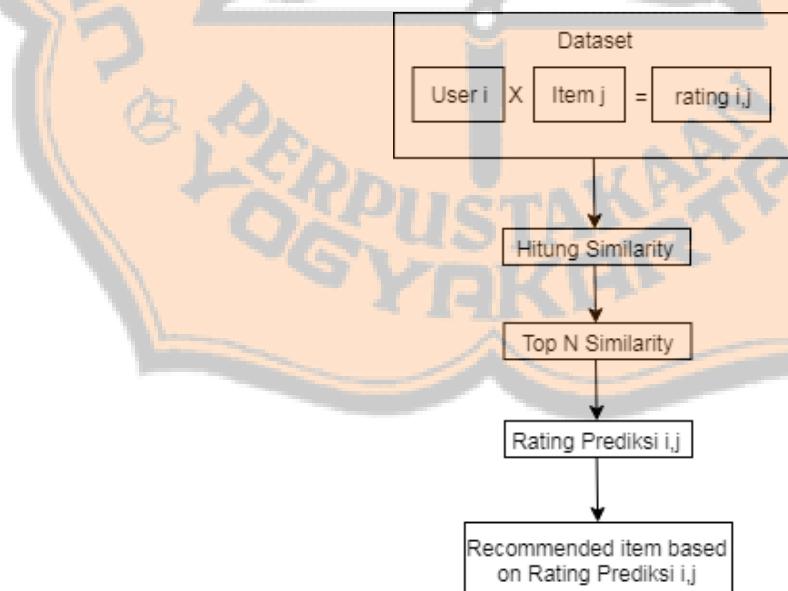
- CP = Candi Prambanan

Item-based collaborative filtering digunakan untuk memprediksi data *rating* yang masih kosong (kolom merah) pada tabel 3.1 dan tabel 3.2.

3.2.4 Perancangan dan Implementasi Sistem

Dengan data *rating user*, penulis mencoba membangun sistem rekomendasi yang menggunakan konsep *client-server* berbasis web dan dibangun menggunakan bahasa pemrograman *java (jsp)* dan basis data *mysql*. Penulis juga melakukan perancangan basis data menggunakan *mysql* untuk data set survey, serta melakukan perancangan sistem untuk mengaplikasikan metode *item-based collaborative filtering* ke dalam sistem.

Pada tahap ini penulis mengimplementasikan proses perhitungan prediksi dari tahap pengambilan data *testing* sampai pada perhitungan prediksi pada program. Tahapan perhitungan prediksi dapat dilihat pada gambar 3.2 berikut ini.



Gambar 3.2 Tahapan perhitungan prediksi

3.2.5 Pengujian Sistem

Pengujian sistem dilakukan untuk menentukan seberapa akurat sistem yang dibangun dengan menggunakan *item-based collaborative filtering*. Perhitungan keakuratan dapat dilakukan dengan menghitung *Mean Absolute Error (MAE)*.

3.2.6 Analisa Hasil

Pada tahap ini penulis menganalisa hasil yang yang sudah didapatkan dari tahap pengujian. Analisa hasil dilakukan dengan melakukan pengujian terhadap pengaruh jumlah maksimal *neighbor* terhadap hasil prediksi *rating*, dengan syarat *neighbor* sebagai berikut.

1. *Neighbor* objek wisata telah di-*rating* seperti objek wisata yang akan diprediksi *rating*-nya.
2. Nilai *neighbor (similarity)* dari *item* yang akan diprediksi adalah diatas 0.
3. Jumlah *neighbor* yaitu 4, 6 dan 8 *neighbor*.

Seluruh hasil prediksi akan dihitung nilai MAE nya terhadap nilai *rating* yang sebenarnya.

3.2.7 Penarikan Kesimpulan dan Saran

Pada tahap penarikan kesimpulan, penulis menarik kesimpulan dari seberapa akurat sistem yang akan dibangun sehingga manfaat penelitian dapat terwujud. Penarikan kesimpulan dapat menjadi bahan bagi peneliti lain untuk melanjutkan penelitian lain.

3.3 Spesifikasi *Hardware* dan *Software*

Spesifikasi *hardware* dan *software* yang akan digunakan penulis dalam mengimplementasikan sistem ini adalah :

3.3.1 *Hardware*

1. *Processor* yang digunakan :

Intel® Core(TM) i5-2450M CPU @ 2.50Ghz (4 CPUs), ~2.5Ghz.

2. *Memory (RAM)* yang digunakan adalah 4.00 GB.

3.3.2 *Software*

1. Sistem operasi yang digunakan adalah *microsoft windows 7* 64-bit.
2. Bahasa pemrograman yang digunakan adalah *java (jsp)* dengan aplikasi netbeans.
3. Basis data yang digunakan adalah *MYSQL* dengan menggunakan web server Apache Tomcat pada aplikasi XAMPP.

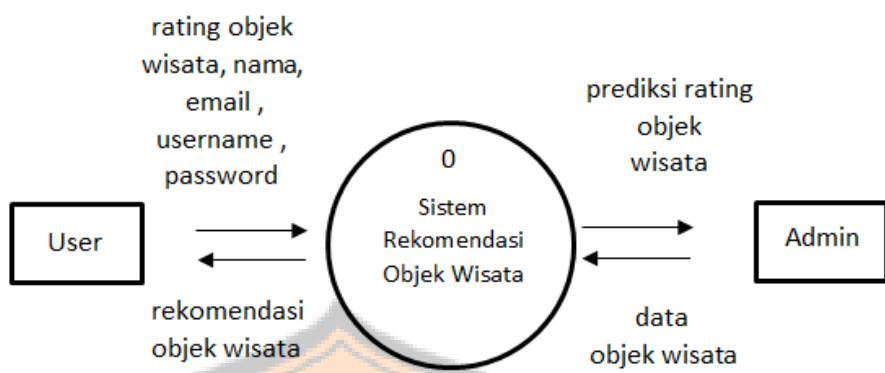
BAB IV ANALISA DAN PERANCANGAN SISTEM

4.1 Analisa Sistem

Sistem yang akan dibangun menggunakan arsitektur *client-server* berbasis web. Sistem akan melakukan perhitungan *similarity* antar item, serta melakukan prediksi *rating*. *User interface (GUI)* memberikan *action* kepada *user* untuk memasukkan data mereka dan memberikan *action* kepada administrator untuk mengetahui prediksi *rating*.

4.1.1 Diagram Konteks

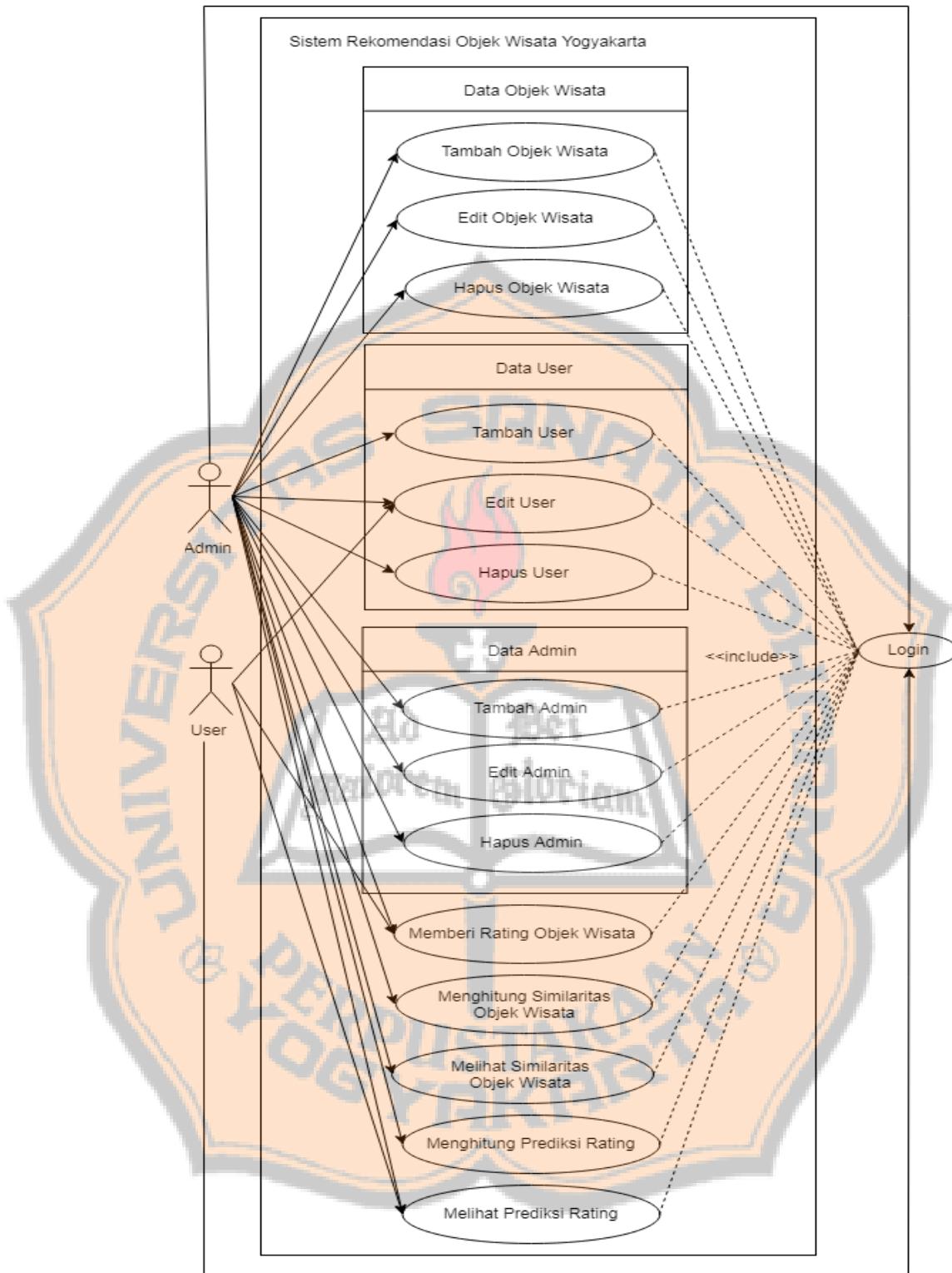
Diagram konteks menggambarkan *input* dan *output* ke dalam sistem yang akan dibangun, seperti pada gambar 4.1.



Gambar 4.1 Diagram Konteks

4.1.2 Use Case Diagram

Use case diagram menggambarkan iteraksi antar aktor dengan sistem dan berguna untuk mengetahui fungsi-fungsi dan hak aktor terhadap fungsi di dalam sistem yang akan dibangun. Berikut ini adalah diagram *Use Case* yang akan disajikan dalam Gambar 4.2.



Gambar 4.2 Use case diagram

4.1.3 Narasi *Use Case*

Narasi *use case* berisi langkah-langkah yang dilakukan aktor (*administrator* atau *user*) terhadap sistem dan reaksi sistem terhadap perintah aktor. Narasi *use case* disajikan pada lampiran 1.

4.2 Perancangan Input

Input yang akan dimasukkan ke dalam sistem adalah data *rating* yang didapat dari survey kepada wisatawan. Data set tersimpan dalam file dengan format .sql, adapun rincian dari data tersebut sebagai berikut.

1. Data *user* sebanyak 100 data dalam tabel *user*.
2. Data objek wisata sebanyak 10 data dalam tabel *objek_wisata*.
3. Range (jangkauan) *rating* yang diberikan adalah 1 – 5 dalam tabel *rating*.

Berikut merupakan contoh data *rating* pada basis data mysql yang disajikan pada gambar 4.3.

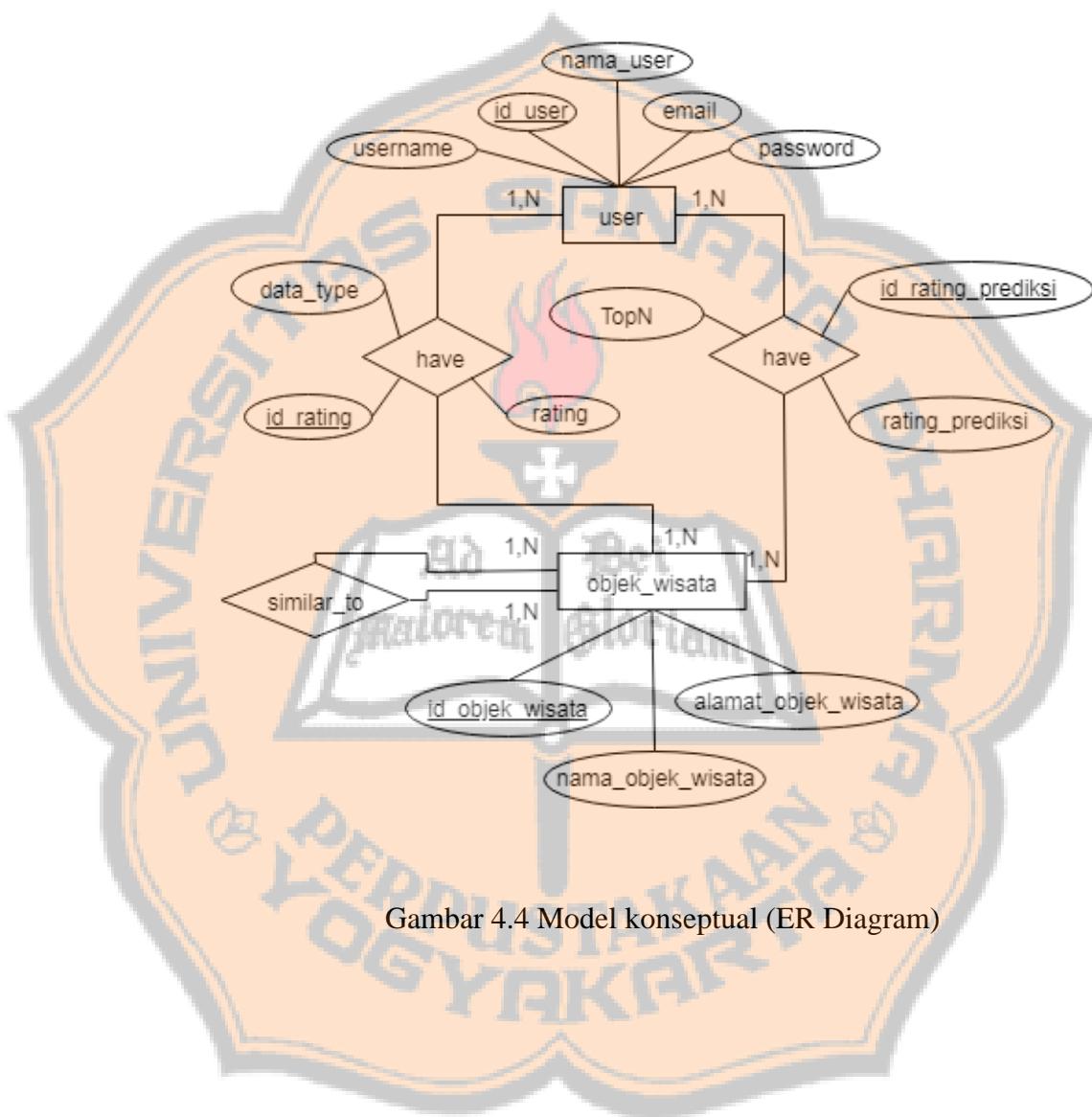
+ Opsi	id_rating	id_user	id_objek_wisata	rating	data_type
	1	1	1	3.00	0
	2	1	2	4.00	0
	3	1	3	3.00	0
	4	1	4	4.00	0
	5	1	5	4.00	0
	6	1	6	4.00	0
	8	1	8	3.00	0
	9	1	9	4.00	0
	11	2	1	4.00	0
	12	2	2	3.00	0
	13	2	3	3.00	0
	14	2	4	3.00	0
	15	2	5	4.00	0
	16	2	6	3.00	0
	17	2	7	3.00	0
	18	2	8	4.00	0
	21	3	1	4.00	0
	22	3	2	4.00	0
	23	3	7	5.00	0
	24	3	9	4.00	0
	25	3	10	5.00	0
	26	4	1	3.00	0
	27	4	2	1.00	0
	28	4	7	1.00	0
	29	4	9	1.00	0

Gambar 4.3 Data *rating* pada database

4.3 Perancangan Basisdata

4.3.1 Perancangan Basisdata secara Konseptual

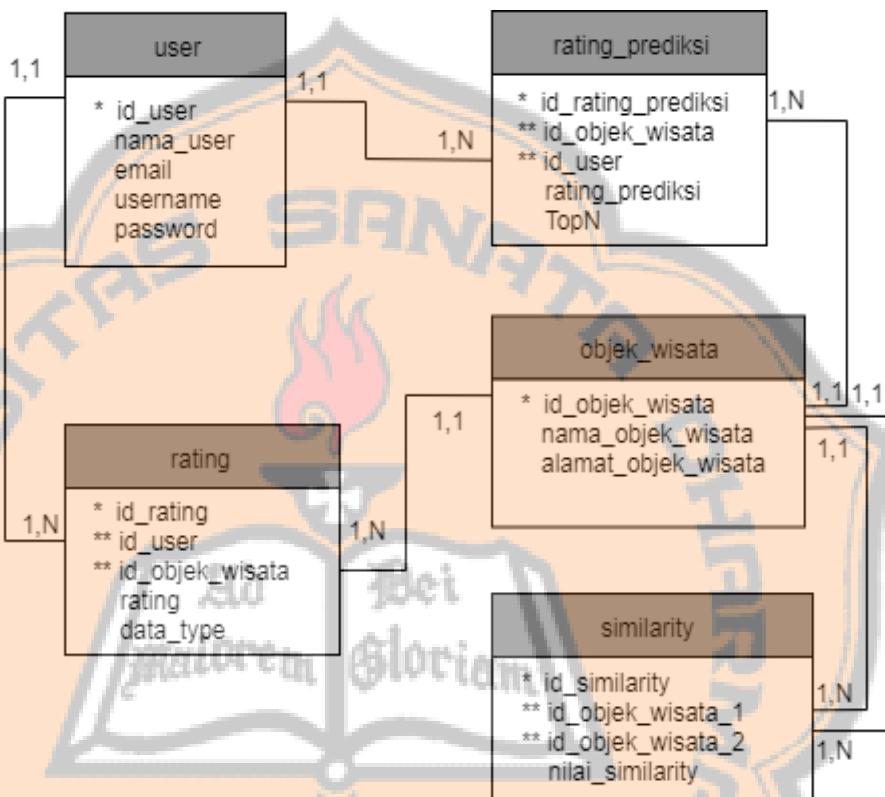
Berikut ini merupakan rancangan basis data yang menggambarkan hubungan antar entitas, seperti yang disajikan pada gambar 4.4.



Gambar 4.4 Model konseptual (ER Diagram)

4.3.2 Desain Basisdata secara Logikal

Berikut ini merupakan model logikal dari rancangan ER diagram yang disajikan pada gambar 4.5.



Gambar 4.5 Model logikal

4.3.3 Desain Basisdata secara Fisikal

Berikut ini merupakan model fisikal dari setiap tabel.

4.3.3.1 Tabel user

No	Atribut	Type	Length	Constraint
1	id_user	int	10	NOT NULL, PRIMARY KEY
2	nama_user	varchar	100	NOT NULL

No	Atribut	Type	Length	Constraint
3	email_user	varchar	100	NOT NULL
4	username	varchar	50	NOT NULL
5	password	varchar	50	NOT NULL

4.3.3.2 Tabel admin

No	Atribut	Type	Length	Constraint
1	id_adm	int	10	NOT NULL, PRIMARY KEY
2	nama_adm	varchar	100	NOT NULL
3	email_adm	varchar	100	NOT NULL
4	username_adm	varchar	50	NOT NULL
5	password_adm	varchar	50	NOT NULL

4.3.3.3 Tabel objek_wisata

No	Atribut	Type	Length	Constraint
1	id_objek_wisata	int	10	NOT NULL, PRIMARY KEY
2	nama_objek_wisata	varchar	100	NOT NULL
3	alamat_objek_wisata	varchar	250	NOT NULL

4.3.3.4 Tabel rating

No	Atribut	Type	Length	Constraint
1	id_rating	int	10	NOT NULL, AUTO_INCREMENT, PRIMARY KEY
2	id_user	int	10	NOT NULL, FOREIGN KEY

No	Atribut	Type	Length	Constraint
3	id_objek_wisata	int	10	NOT NULL, FOREIGN KEY
4	rating	double	5,2	
5	data_type	int	2	

4.3.3.5 Tabel rating_prediksi

No	Atribut	Type	Length	Constraint
1	id_rating_prediksi	int	10	NOT NULL, AUTO_INCR EMENT, PRIMARY KEY
2	id_objek_wisata	int	10	NOT NULL, FOREIGN KEY
3	id_user	int	10	NOT NULL, FOREIGN KEY
4	rating_prediksi	double	5,2	
5	TopN	int	5	

4.3.3.6 Tabel similarity

No	Atribut	Type	Length	Constraint
1	id_similarity	int	10	NOT NULL, AUTO_INC REMENT, PRIMARY KEY
2	id_objek_wisata_1	int	10	NOT NULL, FOREIGN

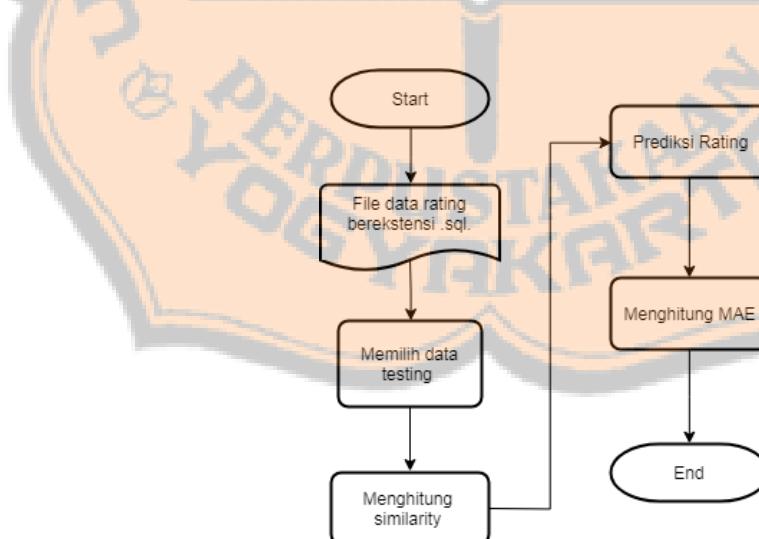
No	Atribut	Type	Length	Constraint
				KEY
3	id_objek_wisata_2	int	10	NOT NULL, FOREIGN KEY
4	Nilai_similarity	double	5,2	

4.4 Perancangan Proses

Pada bagian ini akan dijelaskan setiap proses dari sistem sampai mendapatkan prediksi dan menghitung *error*. Terdapat beberapa tahapan proses dengan metode *item-based collaborative filtering* ini, sebagai berikut.

1. Menyimpan setiap data set ke dalam basis data yang berekstensi .sql.
2. Memilih data *testing* pada tabel *rating*.
3. Menghitung *similarity* antar item dengan pendekatan *pearson correlation* untuk metode *item-based collaborative filtering*.
4. Melakukan prediksi *rating* pada data *testing*.
5. Menghitung *error* pada prediksi *rating* dengan MAE.

Proses pada sistem ini disajikan dalam *flowchart* pada gambar 4.6 berikut ini.



Gambar 4.6 *Flowchart* perancangan proses pada sistem

Terdapat 5 kelas program untuk mengimplementasikan proses pada blok proses yang disajikan pada tabel 4.1 berikut ini.

Tabel 4.1 Rancangan implementasi blok proses pada kelas .java

No	Nama Blok Proses	Kelas
1	Memilih data <i>testing</i>	Rating.java
2	Menghitung <i>similarity</i>	ItemBased.java
3	Memprediksi <i>rating</i>	Prediksi.java
4	Menghitung MAE	MAE.java

4.4.1 Algoritma Blok Proses

Berikut ini merupakan *method* dan atribut pada setiap kelas pada implementasi blok proses serta algoritmanya.

1. Memilih Data *Testing*

Setiap *user* yang me-*rating* lebih dari 6 objek wisata akan diambil 2 data *rating*-nya, sedangkan *user* yang me-*rating* kurang dari 6 objek wisata tidak akan diambil *rating*-nya untuk data *testing*. Penerapan blok proses “Memilih data *testing*” terdapat pada kelas Rating.java dan berikut merupakan uml diagram pada gambar 4.7 beserta dengan algoritmanya.

Rating.java
<pre> - id_user : int - id_objek_wisata : int - rating : String - id_rating : int - conn : Connection + Rating() <<konstruktor>> + Rating(Connection) <<konstruktor>> + getKoneksi() : Rating + getId_user() : int + setId_user(int) : void + getId_objek_wisata() : int + setId_objek_wisata(int) : void + getRating() : String + setRating(String) : void + getId_rating() : int + setId_rating(int) : void + inputRating(String,String,String) : void + inputRatingUser(String,String,String) : void + getAmbilRating(String,String) : List<Rating> + getRatingTesting() : List<Rating> + getTotalUserRated() : int + getUserbyId(String) List<Rating> + UpdateDataRatingType(Rating) : void + pilihDataTesting() : void </pre>

Gambar 4.7 Diagram uml Rating.java

a. Method **getTotalUserRated()**

Kelas ini digunakan untuk menghitung total *user* yang telah me-*rating* objek wisata.

1. Bangun koneksi ke *database MySQL*.
2. Deklarasi *query* untuk melakukan perintah *select* total jumlah *rating* oleh masing-masing *user*.
3. Eksekusi *query*.
4. Deklarasi variabel **totalUser** bertipe *integer*.
5. Lakukan perulangan selama *result* masih ada :
 - a. Hitung total baris pada *result* dengan deklarasi **totalUser++**
6. Kembalikan variabel **totalUser** sebagai nilai *return method*.

b. Method *getUserbyId(string)*

Digunakan untuk menampung riwayat *rating* yang telah diberikan *user* sebelumnya.

1. Membaca parameter *method* sebagai *id_user*.
2. Membangun koneksi ke *database MySQL*.
3. Deklarasi *query* untuk *select* semua data pada tabel *rating_training* dimana *id_user* sama dengan parameter *method* dan urutkan berdasarkan *id_user*.
4. Eksekusi *query*.
5. Deklarasi *ArrayList rating* dari kelas *Rating* dengan objek bernama **rat**.
6. Lakukan perulangan selama *result* masih ada :
 - a. Deklarasi objek baru dari kelas *Rating* dengan nama **rt**.
 - b. Masukkan semua data dari *result* ke objek tersebut.
 - c. Masukkan objek kelas *Rating* ke objek *ArrayList*.
7. Kembalikan objek **rat** sebagai nilai *return method*.

c. Method *pilihDataTesting()*

Digunakan untuk memilih data *testing* dari setiap *user* berdasarkan kondisi yang telah dirancang sebelumnya.

1. Deklarasi objek bernama **r** dari kelas *Rating*.
2. Deklarasi *ArrayList Rating* dengan objek bernama **rat**.
3. Panggil *method getTotalUserRated()* dan menampung nilainya di variabel **totalUser** bertipe integer.
4. Deklarasi variabel **totalDataRating** bertipe integer.
5. Deklarasi objek bernama **random** dari kelas *Random*.
6. Lakukan perulangan sebanyak nilai variabel **totalUser**.
 - a. Panggil *method getUserbyId(String)* dan tampung nilainya di objek **rat**.
 - b. Lakukan percabangan jika banyaknya objek wisata yang di-*rating user* lebih dari 6 maka :

1. Deklarasi variabel bernama **ranawal** bertipe integer dengan nilai 0.
 2. Deklarasi variabel bernama **looping** bertipe integer dengan nilai 0.
 3. Lakukan perulangan selama nilai variabel **looping** tidak sama dengan 2.
 - a. Panggil *method* **nextInt(int)** dengan objek **random** dan ditampung di variabel **ran** bertipe integer.
 - b. Lakukan percabangan dengan kondisi nilai variabel **ran** tidak sama dengan nilai variabel **ranawal**
 - c. Masukkan data result ke dalam objek **r**.
 - d. Panggil *method* **inputDataRatingTesting(r)**.
 - e.Tambahkan nilai variabel **totalDataRating** dengan 1.
 - f. *Update* nilai variabel **ranawal** dengan nilai variabel **ran**.
 - g. Tambahkan nilai variabel **looping** dengan 1.
 7. Tutup koneksi ke *database*.
- d. Method UpdateDataRatingType(Rating)**
- Digunakan untuk meng-*update* tipe data menjadi data testing:
1. Membaca parameter *method* sebagai objek bernama kelas **Rating** rat.
 2. Membangun koneksi ke *database* MYSQL.
 3. Deklarasi *query* untuk meng-*update* **data_type** pada tabel **rating**.
 4. Eksekusi *query*.
 5. Tutup koneksi ke *database*.

2. Menghitung Similarity

Proses perhitungan similarity dilakukan untuk menghitung kedekatan atau *neighbor* antar objek wisata. Terdapat 6 *method* untuk menghitung similarity pada kelas ItemBased.java. Berikut merupakan rancangan kelas pada gambar 4.8 dan rancangan algoritma pada setiap *method*.

ItemBased.java	
- id_similarity_objek_wisata : int	
- id_objek_wisata1 : int	
- id_objek_wisata2 : int	
- nilai_similarity : Double	
- conn : Connection	
+ ItemBased() <<konstruktor>>	
+ ItemBased(Connection) <<konstruktor>>	
+ getKoneksi(): ItemBased	
+ getId_similarity_Objek_wisata() : int	
+ setId_similarity_Objek_wisata(int) : void	
+ getId_objek_wisata1() : int	
+ setId_objek_wisata1(int) : void	
+ getId_objek_wisata2() : int	
+ setId_objek_wisata2(int) : void	
+ getNilai_similarity() : Double	
+ setNilai_similarity(Double) : void	
+ getRating(String) : List<Rating>	
+ getAverage(String) : double	
+ getAtas(List<Rating>,List<Rating>,double,double) : double	
+ getBawah(List<Rating>,List<Rating>,double,double) : double	
+ similaritasPearsonCorrelation(String,String) : double	
+ InputSimilarity(String,String,ItemBased) : void	
+ getSimilarity1ObjekWisata(String) : List<ItemBased>	
+ getSimilarity() : List<ItemBased>	
+ similarity() : void	

Gambar 4.8 Diagram uml kelas ItemBased.java

a. Method *getRating(String)*

Digunakan untuk menampung semua rating yang dimiliki oleh objek wisata tertentu.

1. Membaca parameter di method sebagai id_objek_wisata.
2. Membangun koneksi ke *database* MYSQL.
3. Deklarasi *query* untuk mengambil *rating* dengan kondisi data yang diambil adalah data objek wisata tertentu.
4. Ekseskusi *query*.
5. Deklarasi ArrayList Rating dengan objek bernama **rt**.
6. Lakukan perulangan selama *result* ada.

7. Deklarasi objek baru bernama **r** dari kelas Rating.
8. Masukkan *result* ke dalam objek **r**.
9. Masukkan objek **r** ke dalam objek **rt**.
10. Kembalikan objek **rt** sebagai nilai *return method*.

b. Method *getAverage(String)*

Digunakan untuk menghitung rata-rata semua *rating* pada objek wisata tertentu.

1. Baca parameter pada *method* sebagai **id_objek_wisata**.
2. Buat koneksi ke *database MySQL*.
3. Deklarasi *query* untuk mengambil rata-rata semua *rating* yang dimiliki objek wisata tertentu.
4. Eksekusi *query*.
5. Deklarasi variabel bernama **avg** dengan tipe double dan nilai 0.
6. Lakukan perulangan selama *result* ada :
 - a. Tampung *result* ke variabel **avg**.
7. Kembalikan variabel **avg** sebagai nilai *return method*.

c. Method *getAtas(List<Rating>,*

List<Rating>,Double,Double

Digunakan untuk menghitung bagian pembilang pada rumus *Pearson Correlarion*.

1. Membaca parameter 2 objek *ArrayList* dari kelas Rating dan membaca 2 variabel parameter *method*.
2. Deklarasi variabel bernama
totalPerkalianRatingMinAvg bertipe double dengan nilai 0.
3. Deklarasi variabel bernama **ratingMinAvg1** bertipe double dengan nilai 0.
4. Deklarasi variabel bernama **ratingMinAvg2** bertipe double dengan nilai 0.

5. Deklarasi variabel bernama **perkalianItem** bertipe double dengan nilai 0.
 6. Lakukan perulangan sebanyak ukuran dari parameter **objek_wisata1** :
 - a. Lakukan perulangan sebanyak ukuran dari parameter **objek_wisata2** :
 1. Lakukan percabangan dimana parameternya jika id user dari objek_wisata1 sama dengan id user objek_wisata2 :
 - a. Nilai *rating* dari **objek objek_wisata1** dikurangi dengan nilai **average1** dari variabel parameter *method* dan ditampung di variabel **ratingMinAvg1**.
 - b. Nilai *rating* dari objek objek_wisata2 dikurangi dengan nilai **average2** dari variabel parameter *method* dan ditampung di variabel **ratingMinAvg2**.
 - c. Kalikan variabel **ratingMinAvg1** dengan variabel **ratingMinAvg2** dan ditampung di variabel **totalPerkalianRatingMinAvg**.
 7. Kembalikan variabel **totalPerkalianRatingMinAvg** sebagai nilai *return method*.
- d. **Method getBawah(List<Rating>, List<Rating>,Double,Double)**
- Digunakan untuk menghitung bagian penyebut pada rumus *Pearson Correlation*.
1. Membaca parameter 2 objek ArrayList dari kelas Rating dan membaca 2 variabel parameter *method*.
 2. Deklarasi variabel bernama **ratingMinAvg1** bertipe double dengan nilai 0.

3. Deklarasi variabel bernama **totalratingMinAvg1** bertipe double dengan nilai 0.
4. Deklarasi variabel bernama **ratingMinAvg2** bertipe double dengan nilai 0.
5. Deklarasi variabel bernama **totalratingMinAvg2** bertipe double dengan nilai 0.
6. Deklarasi variabel bernama **perkalianPangkat** bertipe double dengan nilai 0.
7. Deklarasi variabel bernama **akarPerkalianJumlahRatingMinAvg** bertipe double dengan nilai 0.
8. Lakukan perulangan sebanyak ukuran dari parameter **objek_wisata1** :
 - a. Lakukan perulangan sebanyak ukuran dari parameter **objek_wisata2** :
 1. Lakukan percabangan dimana parameternya jika id *user* dari **objek_wisata1** sama dengan id *user* **objek_wisata2** :
 - a. Lakukan perhitungan *rating* dari objek **objek_wisata1** dikurangi variabel **average1**, kemudian dipangkat 2 dan diakarkan dan disimpan di variabel **ratingMinAvg1**.
 - b. Lakukan penjumlahan antara variabel **totalratingMinAvg1** dengan variabel **ratingMinAvg1** dan ditampung di variabel **totalratingMinAvg1**.
 - c. Lakukan perhitungan rating dari objek **objek_wisata2** dikurangi variabel **average2**, kemudian dipangkat 2 dan diakarkan dan disimpan di variabel **ratingMinAvg2**.
 - d. Lakukan penjumlahan antara variabel **totalratingMinAvg2** dengan variabel

ratingMinAvg2 dan ditampung di variabel **totalratingMinAvg2**.

9. Lakukan perkalian antara variabel **totalratingMinAvg1** dengan variabel **totalratingMinAvg2** dan ditampung di variabel **perkalianPangkat**.
10. Akarkan variabel **perkalianPangkat** dan ditampung di variabel **akarPerkalianJumlahRatingMinAvg**.
11. Kembalikan variabel **akarPerkalianJumlahRatingMinAvg** sebagai nilai *return method*.

e. **similaritasPearsonCorrelation(String, String)**

Digunakan untuk menghitung *similarity* antar objek wisata dengan menjalankan *method method* yang melakukan perhitungan dengan rumus *Pearson Correlation*.

1. Membaca 2 parameter pada *method* sebagai objek_wisataX dan objek_wisataY bertipe String.
2. Deklarasi variabel **hasil** bertipe double dengan nilai 0,0.
3. Lakukan percabangan jika nilai objek_wisataX dan objek_wisataY sama, maka:
 - a. Nilai variabel **hasil** sama dengan 1.
4. Lakukan percabangan:
 - a. Panggil *method* **getRating(String)** parameter yang dimasukkan adalah parameter objek_wisataX dan hasilnya ditampung di variabel **objek_wisata1** bertipe ArrayList.
 - b. Panggil *method* **getRating(String)** parameter yang dimasukkan adalah parameter objek_wisataY dan hasilnya ditampung di variabel **objek_wisata2** bertipe ArrayList.

- c. Panggil method **getAverage(String)** dengan parameter id_objek_wisata dari objek objek_wisata1 dan disimpan di variabel **average1** bertipe double.
- d. Panggil method **getAverage(String)** dengan parameter id_objek_wisata dari objek objek_wisata2 dan disimpan di variabel **average2** bertipe double.
- e. Panggil method **getAtas(List<Rating>, List<Rating>, Double, Double)** dengan parameter objekwisata1, objekwisata2, average1 dan average2 dan disimpan di variabel **bagianAtasPC** dengan tipe double.
- f. Panggil method **getBawah(List<Rating>, List<Rating>, Double, Double)** dengan parameter objekwisata1, objekwisata2, average1 dan average2 dan disimpan di variabel **bagianBawahPC** dengan tipe double.
- g. Hitung variabel **bagianAtasPC** dibagi dengan variabel **bagianBawahPC** dan ditampung di variabel **hasil**.
- h. Lakukan percabangan untuk mengecek apakah variabel **hasil** adalah sebuah nomor atau bukan:
 1. Jika *true* maka variabel hasil di set 0.
 5. Tampilkan *output*.
 6. Kembalikan variabel **hasil** sebagai nilai **return** method.

f. Method *InputSimilarity(String, String, ItemBased)*

Digunakan untuk menyimpan *similarity* dan meng-update *similarity* objek wisata yang sudah ada di *database*.

- 1. Membaca parameter pada method sebagai objek_wisataX, objek_wisataY, dan ib.
- 2. Bangun koneksi ke *database* MYSQL.

3. Deklarasi *query* untuk cek apakah sudah ada *similarity* untuk dua objek wisata tertentu yang tersimpan di *database*.
4. Eksekusi *query*.
5. Lakukan percabangan selama *result* masih ada:
 - a. Bangun koneksi ke *database MySQL*.
 - b. Deklarasi *query* untuk meng-update *similarity* antara dua objek wisata tertentu.
 - c. Eksekusi *query*.
6. Lakukan percabangan jika percabangan sebelumnya tidak dieksekusi:
 - a. Bangun koneksi ke *database MySQL*.
 - b. Deklarasi *query* untuk menambahkan data *similarity* dua objek wisata tertentu ke *database*.
 - c. Eksekusi *query*.

3. Menghitung *Prediksi*

Perhitungan prediksi *item-based collaborative filtering* membutuhkan beberapa langkah dan menggunakan *rating* objek wisata lain dan nilai *similarity* antar objek wisata untuk memprediksi sebuah *rating*. Terdapat 6 *method* pada kelas Prediksi.java untuk menerapkan blok proses ini. Berikut ini merupakan diagram uml pada gambar 4.9, beserta dengan algoritma dari setiap *method*.



Gambar 4.9 Diagram uml kelas Prediksi.java

a. Method *getRatingTraining(int)*

Digunakan untuk mengambil id objek wisata dari tabel objek_wisata dimana id objek wisata dengan *user* tertentu itu tidak ada di dalam tabel rating_training.

1. Baca parameter pada method sebagai *id_user* bertipe *integer*.
2. Bangun koneksi ke *database MySQL*.

3. Deklarasi *query* untuk mengambil data id objek wisata dari tabel objek_wisata dengan kondisi id objek wisata tidak ada di dalam tabel rating_training dengan id *user* tertentu.
4. Eksekusi *query*.
5. Deklarasi objek ArrayList bernama **rat** dari kelas Rating.
6. Lakukan perulangan selama *result* ada:
 - a. Deklarasi objek bernama **rt** dari kelas Rating.
 - b. Masukkan semua result ke **rt**.
 - c. Masukkan objek **rt** ke objek **rat**.
7. Kembalikan objek **rat** sebagai nilai *return method*.

b. *getRatingAndOWSimilaritySort(String, String)*

Digunakan untuk mengambil *similarity* antar objek wisata dan diurutkan dari yang tertinggi.

1. Baca parameter *method* sebagai *id_user* dan *id_objek_wisata*.
2. Buat koneksi ke *database MySQL*.
3. Deklarasi *query* untuk mengambil *similarity* objek wisata pada tabel *similarity* dengan kondisi nilai *similarity* lebih dari 0, urutkan berdasarkan nilai *similarity* dari nilai terbesar ke terkecil.
4. Eksekusi *query*.
5. Deklarasi objek ArrayList bernama **otherOWRatingSort** dari kelas Prediksi.
6. Deklarasi variabel bernama **sql1** bertipe string;
7. Lakukan perulangan selama *result* ada:
 - a. Deklarasi objek bernama **p** dari kelas Prediksi.
 - b. Masukkan data *result* ke objek **p**
 - c. Deklarasi *query* untuk mengambil *rating* dari tabel rating_training dengan kondisi id objek wisata dan id *user* tertentu.
 - d. Eksekusi *query*.

- e. Lakukan perulangan selama *result* ada:
 1. Masukkan *result* ke objek **p**.
 - f. Masukkan *result* dari *query* pertama ke objek **p**.
 - g. Masukkan objek **p** ke objek **otherOWRatingSort**.
8. Kembalikan objek **otherOWRatingSort** sebagai nilai *return method*.

c. **hasilPredksiItemBased(Rating,int)**

Digunakan untuk menghitung prediksi berdasarkan rumus perhitungan prediksi *Item-Based CF* dengan memanggil *method-method* lain untuk perhitungan dan dangan jumlah maksimal *neighbor* yang sudah ditentukan.

1. Baca parameter pada method sebagai *dataTraining* yang berisi data *user* dan *topN* yang berisi jumlah maksimal *neighbor* yang telah ditentukan.
2. Panggil *method* `getRatingAndOWSimilaritySort(String, String)` dan tampung hasilnya ke objek **otherOWRatingSort** bertipe *ArrayList* dari kelas Prediksi.
3. Deklarasi variabel:
 - a. **NilaiSimilarity**.
 - b. **parameterLoop**.
 - c. **ratingUserForOw**.
 - d. **RatingXOWSimilarityValue**.
 - e. **totalOWSimilarityValue**.
 - f. **prediksi**.
4. Lakukan percabangan jika maksimal *neighbor* sama dengan 10:
 - a. Tampung ukuran objek **otherOWRatingSort** sebagai nilai variabel **parameterLoop**.

5. Lakukan percabangan jika ukuran data objek **otherOWRating Sort** lebih kecil dari jumlah maksimum **neighbor**:
 - a. Tampung ukuran objek **otherOWRatingSort** sebagai nilai variabel **parameterLoop**.
6. Lakukan percabangan jika percabangan sebelumnya tidak dieksekusi:
 - a. Masukkan nilai maksimum **neighbor** sebagai nilai **parameterLoop**.
7. Lakukan perulangan dengan parameter perulangan sebanyak nilai **parameterLoop**.
 - a. Masukkan nilai similarity dari objek **otherOWRatingSort** ke variabel **NilaiSimilarity**
 - b. Lakukan percabangan dengan parameter nilai variabel **NilaiSimilarity** lebih dari 0.
 1. Masukkan nilai rating training dari objek **otherOWRatingSort** ke dalam variabel **ratingUserForOW**.
 2. Lakukan perkalian antara variabel **totalRatingXOWSimilarityValue** dengan **RatingXOWSimilarityValue** dan ditampung di variabel **totalRatingXOWSimilarityValue**.
 3. Tampung nilai penjumlahan antara variabel **totalOWSimilarityValue** dengan **NilaiSimilarity** pada variabel **totalOWSimilarityValue**.
 8. Tampung hasil pembagian antara variabel **totalRatingXOWSimilarityValue** dibagi dengan variabel **totalOWSimilarityValue** pada variabel **prediksi**.
 9. Deklarasi variabel **NaN** bernilai null.
 10. Lakukan percabangan untuk mengecek apakah nilai variabel **prediksi** adalah sebuah angka atau bukan dengan method **isNaN()**:

- a. Set nilai variabel **prediksi** dengan 0 jika nilainya bukan angka.
11. Deklarasi objek bernama **p** dari kelas Prediksi.
12. Masukkan data dari objek parameter ke objek **p** dan masukkan variabel **prediksi** ke objek **p**.
13. Panggil method inputPrediksi(Prediksi) untuk memasukkan data ke **database**.
14. Kembalikan variabel **prediksi** sebagai nilai **return method**.

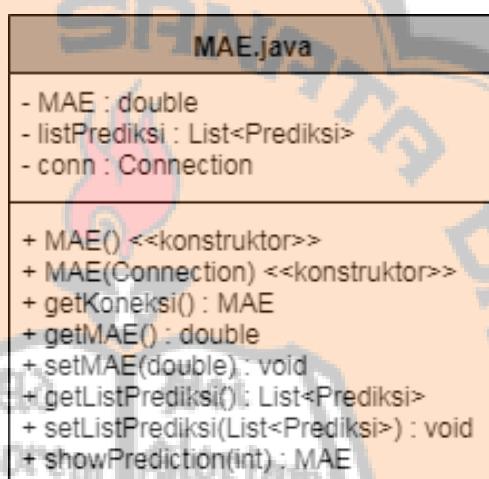
d. *inputPrediksi(Prediksi)*

Digunakan untuk memasukkan data hasil prediksi beserta data maksimal *neighbor,id user* dan id objek wisata ke dalam *database*. Dalam *method* ini, jika prediksi sebuah objek wisata dari seorang *user* dengan maksimal *neighbor* tertentu sudah ada di *database*, maka data akan *di-update* dan jika belum ada maka data akan *di-input*.

1. Baca parameter di *method* yang menyimpan data dari objek *p* di *method* **hasilPrediksiItemBased(Rating,int)**.
2. Buat koneksi ke *database* MYSQL.
3. Deklarasi *query* untuk mengambil data *rating* dari tabel *rating_prediksi* dengan *id user* dan id objek wisata tertentu.
4. Eksekusi *query*.
5. Lakukan percabangan jika *result* ada:
 - a. Deklarasi *query* untuk meng-*update* data *rating* di tabel *prediksi* dengan kondisi *id user* dan id objek wisata tertentu.
 - b. Eksekusi *query*.
6. Lakukan percabangan jika percabangan pertama tidak dieksekusi atau jika *result* tidak ada:
 - a. Deklarasi *query* untuk meng-*input* data ke tabel *rating_prediksi*.
 - b. Eksekusi *query*.

4. Menghitung MAE

Perhitungan MAE dilakukan untuk menguji seberapa akurat sebuah prediksi terhadap *rating* sebuah objek wisata, perhitungan MAE dilakukan dengan melakukan pengurangan *rating* prediksi dengan *rating* sebenarnya. Terdapat satu *method* untuk menghitung MAE pada kelas MAE.java. Berikut ini merupakan diagram uml yang disajikan pada gambar 4.10 dan implementasi blok proses pada program.



Gambar 4.10 Diagram uml kelas MAE.java

a. *showPrediction(int)*

Digunakan untuk menghitung MAE dan mengambil data dari tabel rating_prediksi dan tabel rating_testing.

1. Baca parameter *method* sebagai topN yang merupakan maksimum *neighbor* dari *rating* prediksi.
2. Bangun koneksi ke *database* MYSQL.
3. Deklarasi *query* untuk mengambil data dari tabel rating_prediksi dan rating_testing dari tabel rating kemudian di *join* dengan id *user* dan id objek wisata dimana maksimal *neighbor*-nya sama dengan nilai di parameter *method*.

4. Eksekusi *query*.
5. Deklarasi objek ArrayList bernama **predictionList** dari kelas Prediksi.
6. Deklarasi variabel **bantuMAE** dan MAE bertipe double dan bernilai 0.
7. Lakukan perulangan selama *result* berisi:
 - a. Deklarasi objek bernama **p** dari kelas Prediksi.
 - b. Masukkan data *result* ke objek **p**.
 - c. Masukkan objek **p** ke objek **predictionList**.
 - d. Hitung nilai mutlak **rating** sebenarnya dikurangi **rating** prediksi dengan objek **p** ditambah dengan nilai variabel **bantuMAE** dan ditampung di variabel **bantuMAE**.
8. Lakukan pembagian antara variabel **bantuMAE** dibagi ukuran objek **predictionList** dan disimpan di variabel MAE sebagai hasil *Mean Average Error*.
9. Deklarasi objek baru bernama **predictionListWithMAE** dari kelas MAE.
10. Masukkan objek dan data ke objek **predictionListWithMAE**.
11. Kembalikan objek **predictionListWithMAE** sebagai nilai *return method*.

4.4.2 Contoh Perhitungan Metode

Penulis menjabarkan tentang penerapan algoritma ke perhitungan dengan sampel data asli dari *user*.

4.4.2.1 Metode *Item-Based CF*

Proses pertama adalah proses perhitungan untuk mengetahui nilai *similarity* dua objek wisata yang telah di-*rating* oleh *user*. Penulis akan menjabarkan perhitungan antar objek wisata pada tabel 3.2 antara **Museum TNI Dirgantara Mandala** dengan **Monumen Jogja Kembali**. Penulis tidak melibatkan 14 data *rating* dalam perhitungan, yaitu 4 data *rating*

karena data tersebut belum di-*rating user* dan 10 data *rating* yang digunakan untuk data *testing*.

1. Langkah pertama yang dilakukan adalah mengambil semua data *rating* yang dimiliki oleh semua *user* untuk kedua objek wisata, seperti yang disajikan pada tabel 4.2 berikut ini.

Tabel 4.2 Data *rating* untuk contoh perhitungan metode *Item-Based CF*

	MTAU	MJK
Angelina Erlita	5.00	5.00
Stevania E.M.S		3.50
Linda vixna	1.00	
Sumitro		4.00
Ega	3.50	4.00

Keterangan :

- MTAU = Museum TNI AU Dirgantara Mandala
- MJK = Monumen Jogja Kembali

2. Langkah kedua yang dilakukan adalah menghitung nilai *similarity* Museum TNI Dirgantara Mandala dengan Monumen Jogja Kembali dengan rumus 2.1. Langkah perhitungan *similarity* adalah sebagai berikut :

- a. Menghitung rata-rata *rating* yang dimiliki sebuah objek wisata :

Rata-rata *rating* Museum TNI Dirgantara Mandala =

$$\frac{5 + 1 + 3.5}{3} = 3.17$$

Rata-rata *rating* Monumen Jogja Kembali =

$$\frac{5 + 3.5 + 4 + 4}{4} = 4.10$$

- b. Mengambil semua interseksi *rating* antara Museum TNI Dirgantara Mandala dan Monumen Jogja Kembali:

Tabel 4.3 Data interseksi *rating* antara Museum TNI Dirgantara Mandala dengan Monumen Jogja Kembali.

	MTAU	KRB
Angelina Erlita	5.00	5.00
Ega	3.50	4.00

Keterangan :

- MTAU = Museum TNI AU Dirgantara Mandala
- MJK = Monumen Jogja Kembali

Seperti yang diketahui bahwa terdapat 2 *user* yang sudah memberikan *rating* kepada kedua objek wisata (interseksi).

- c. Menghitung *similarity* dengan rumus *Pearson Correlation*:

PC(Museum TNI Dirgantara Mandala, Monumen Jogja Kembali) =

$$\frac{(5 - 3.17)(5 - 4.10) + (3.5 - 3.17)(4 - 4.10)}{\sqrt{(5 - 3.17)^2 + (3.5 - 3.17)^2} \sqrt{(5 - 4.10)^2 + (4 - 4.10)^2}}$$

PC(Museum TNI Dirgantara Mandala, Monumen Jogja Kembali) =

$$\frac{1.62}{1.69}$$

PC(Museum TNI Dirgantara Mandala, Monumen Jogja Kembali) = 0.96

Setelah dilakukan perhitungan *similarity* didapat hasil *similarity* **Museum TNI Dirgantara Mandala** dengan **Monumen Jogja Kembali** adalah **0.96**. Dengan langkah perhitungan yang sama dilakukan juga perhitungan *similarity* antara 1 objek wisata dengan objek wisata lainnya, sehingga menghasilkan sebuah *matrix similarity* objek wisata seperti pada tabel 4.4 berikut ini.

Tabel 4.4 *Matrix similarity* objek wisata dari hasil perhitungan *similarity* antar objek wisata

	MTAU	MJK	TB	KRB	MBV	TS	KY	DMA	TP	CP
MTAU	1.00	0.96	0.81	0.81	0.83	0.60	0.97	0.87	-1.00	0.98
MJK	0.96	1.00	-0.07	-0.04	0.58	0.58	0.47	0.16	0.22	0.91
TB	0.81	-0.07	1.00	1.00	0.78	0.29	0.86	0.98	0.22	0.91
KRB	0.81	-0.04	1.00	1.00	0.78	0.31	0.84	0.80	0.46	0.89
MBV	0.83	0.58	0.78	0.78	1.00	0.78	0.67	0.68	0.45	0.89
TS	0.60	0.58	0.29	0.31	0.78	1.00	0.17	0.69	0.73	0.31
KY	0.97	0.47	0.86	0.84	0.67	0.17	1.00	0.56	-0.64	0.84
DMA	0.87	0.16	0.98	0.80	0.68	0.69	0.56	1.00	1.00	0.87
TP	-1.00	0.61	0.22	0.46	0.45	0.73	-0.64	1.00	1.00	0.54
CP	0.98	-0.14	0.91	0.89	0.89	0.31	0.84	0.87	0.54	1.00

Keterangan :

- MTAU = Museum TNI AU Dirgantara Mandala
- MJK = Monumen Jogja Kembali
- TB = Tebing Breksi
- KRB = Kraton Ratu Boko
- MBV = Museum Benteng Vredeburg
- TS = Taman sari
- KY = Kraton Yogyakarta
- DMA = De Mata Art Museum
- TP = Taman Pintar
- CP = Candi Prambanan

3. Langkah ketiga yang dilakukan adalah menghitung prediksi *rating*. Tahapan ini merupakan tahap akhir dari metode dimana prediksi dihitung menggunakan nilai *similarity* dari tahap sebelumnya.
1. Pada tabel 3.2 penulis mengasumsikan bahwa *rating* **Sumitro terhadap Museum TNI AU Dirgantara Mandala** belum pernah di-*rating* sehingga data tersebut dijadikan data *testing*.

Tabel 4.5 Contoh data *testing*

	MTAU	MJK	TB	KRB	MBV	TS	KY	DMA	TP	CP
Sumitro	?	4.00	?	4.00	4.00	5.00	3.00	5.00	5.00	5.00

Keterangan :

- MTAU = Museum TNI AU Dirgantara Mandala
- MJK = Monumen Jogja Kembali
- TB = Tebing Breksi
- KRB = Kraton Ratu Boko
- MBV = Museum Benteng Vredeburg
- TS = Taman sari
- KY = Kraton Yogyakarta
- DMA = De Mata Art Museum
- TP = Taman Pintar
- CP = Candi Prambanan

Pada tabel 4.5 diketahui bahwa objek wisata yang kosong sudah diberikan *rating*, namun data Museum TNI AU Dirgantara Mandala dan Tebing Breksi dijadikan data *testing* dan Sumitro dianggap belum memberikan *rating* terhadap kedua objek wisata tersebut. Pada kenyataannya, *rating* yang telah diberikan Sumitro terhadap Museum

TNI AU Dirgantara Mandala adalah 3 dan terhadap Tebing Breksi adalah 4.

2. Langkah selanjutnya adalah menentukan jumlah *neighbor* untuk perhitungan. Pada contoh ini penulis akan menggunakan 6 *neighbor* dalam perhitungan prediksi.
3. Langkah selanjutnya adalah menentukan beberapa kondisi dalam menentukan jumlah *neighbor*, yaitu:
 - a. *Neighbor* objek wisata yang telah di-*rating* Sumitro.
 - b. *Similarity* antara Museum TNI AU Dirgantara Mandala dengan *neighbor*-nya adalah bernilai diatas 0. Pada tabel 4.5 diketahui bahwa ada 8 *neighbor* yang nilainya diatas 0, yaitu Monumen Jogja Kembali, Tebing Breksi, Museum Benteng Vredeburg, Taman sari, Kraton Yogyakarta, De Mata Art Museum dan Candi Prambanan. Diketahui juga bahwa terdapat nilai *similarity* yang sama, yaitu Museum TNI AU Dirgantara Mandala dengan Tebing Breksi dan Museum TNI AU Dirgantara Mandala dengan Kraton Ratu Boko dengan nilai *similarity* 0.81. Namun karena Sumirto belum me-*rating* Tebing Breksi, maka *similarity* yang digunakan adalah *similarity* Museum TNI AU Dirgantara Mandala dengan Kraton Ratu Boko.
 - c. Langkah selanjutnya adalah melakukan perhitungan prediksi menggunakan *weighted sum* sesuai dengan rumus 2.2:

Prediksi *rating* Sumitro terhadap Museum TNI AU Dirgantara Mandala:

$$\frac{(5 * 0.98) + (3 * 0.97) + (4 * 0.96) + (5 * 0.87) +}{|(0.89 + 0.97 + 0.96 + 0.87) +}$$

$$\frac{(4 * 0.83) + (4 * 0.81)}{(0.83 + 0.81)} |$$

$$= 4.16$$

Hasil prediksi *rating* yang akan diberikan Sumitro terhadap Museum TNI AU Dirgantara Mandala adalah 4.16, sedangkan *rating* aslinya adalah 3.00. Selisih antara *rating* prediksi dengan *rating* sebenarnya inilah yang akan dihitung seberapa besar kesalahannya dengan menggunakan MAE.

4.4.2.2 Perhitungan *Mean Absolute Error* (MAE)

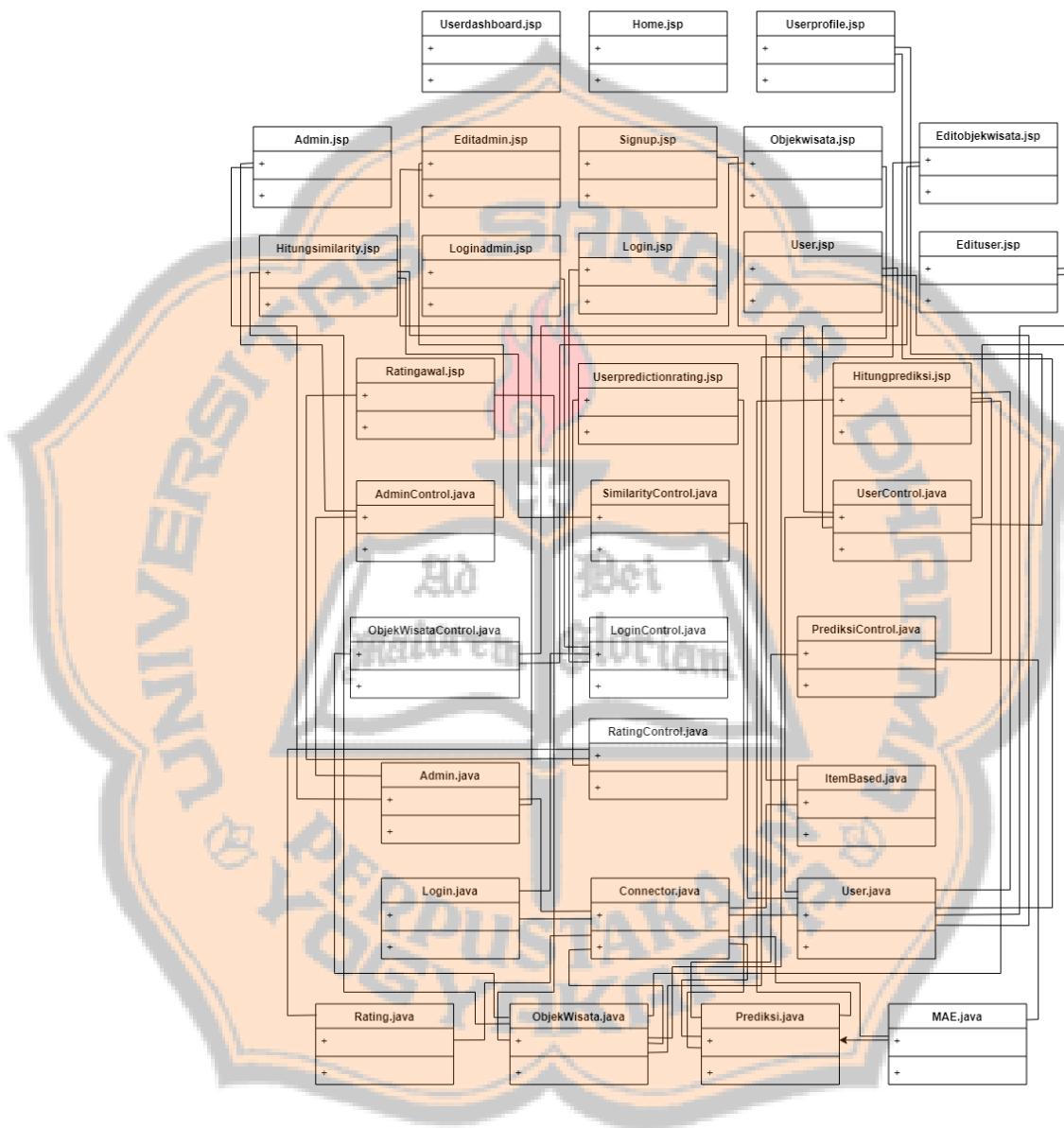
Diketahui bahwa *rating* asli yang diberikan Sumitro terhadap Museum TNI AU Dirgantara Mandala adalah 3.00 dan *rating* prediksinya adalah 4.16. Dengan data ini akan dihitung nilai kesalahannya, dan apabila nilai hasil pengujian MAE semakin mendekati 0 maka hasil prediksi semakin akurat. Berikut menggunakan perhitungan MAE sesuai dengan rumus 3.1:

$$MAE = \frac{|4.16 - 3.00|}{1} = 1.16$$

Dari perhitungan diatas diketahui bahwa MAE dari prediksi *rating* Sumitro terhadap Museum TNI AU Dirgantara Mandala adalah 1.16.

4.4.3 Diagram Kelas Desain

Pada gambar 4.11 akan dijabarkan diagram kelas desain yang menunjukkan hubungan antar kelas pada sistem aplikasi web.



Gambar 4.11 Diagram kelas desain

4.4.3.1 Kelas View

Terdapat 16 kelas *interface* sistem kepada aktor, 4 kelas *interface* untuk aktor *user*, 10 kelas *interface* untuk aktor *admin*

dan 2 kelas *interface* untuk aktor *user* dan *admin*. Berikut adalah kelas *view* yang terdapat pada kelas desain.

4.4.3.1.1 Kelas View Admin

	Admin.jsp
-	- nama : text - email : email - username : text - password : password - submit : input - reset : input - hapus : input
+	+ getAmbilAdmin()
	User.jsp
-	- nama : text - email : email - username : text - password : password - submit : input - reset : input - edit : input - hapus : input
+	+ getAmbilUser()
	Edituser.jsp
-	- nama : text - email : email - username : text - password : password - submit : input - reset : input
+	+ getAmbilUser()
	Objekwisata.jsp
-	- namaow : text - alamatow : text - submit : input - reset : input
+	+ getAmbilOW()
	Editobjekwisata.jsp
-	- namaow : text - alamatow : text - edit : input - reset : input
+	+ getAmbilOW()
	Hitungsimilarity.jsp
-	- similarity : input
+	+ getSimilarity()

Hitungprediksi.jsp	Loginadmin.jsp
<ul style="list-style-type: none"> - hitungprediksi : button - selectprediksi : select - selectneighbor : select - buttonprediksi : button - cancel : button 	<ul style="list-style-type: none"> - username : text - password : password - submitadmin : input - reset : input
+ showPrediction(int)	+

4.4.3.1.2 Kelas View User

Userdashboard.jsp	Userpredictionrating.jsp
<ul style="list-style-type: none"> - 	<ul style="list-style-type: none"> - rating : input - idow : button - rating : input - ratingasli : button - close : button
+	<ul style="list-style-type: none"> + getAmbilOW() + getPrediksiRatingUser(int)

Userprofile.jsp	Login.jsp
<ul style="list-style-type: none"> - nama : text - email : email - username : text - password : text - update : input 	<ul style="list-style-type: none"> - username : text - password : password - submituser : input - reset : input - signup : button - loginadmin : button
+ getAmbil1User()	+

4.4.3.1.3 Kelas View Admin dan User

Signup.jsp	Ratingawal.jsp
<ul style="list-style-type: none"> - nama : text - email : email - username : text - password : password - sign up : input - reset : input 	<ul style="list-style-type: none"> - simpan : input - reset : input - simpan : input - reset : input
+	<ul style="list-style-type: none"> + getAmbilOW()

4.4.3.2 Kelas *Control*

Terdapat 7 kelas *control* untuk menghubungkan antara *view* dengan *model*. Berikut adalah detail 7 kelas pada diagram kelas desain.

AdminControl.java
-
+ processRequest(HttpServletRequest,HttpServletResponse) : protected void + doGet(HttpServletRequest,HttpServletResponse) : protected void + doPost(HttpServletRequest,HttpServletResponse) : protected void
LoginControl.java
-
+ processRequest(HttpServletRequest,HttpServletResponse) : protected void + doGet(HttpServletRequest,HttpServletResponse) : protected void + doPost(HttpServletRequest,HttpServletResponse) : protected void
ObjekWisataControl.java
-
+ processRequest(HttpServletRequest,HttpServletResponse) : protected void + doGet(HttpServletRequest,HttpServletResponse) : protected void + doPost(HttpServletRequest,HttpServletResponse) : protected void
PrediksiControl.java
-
+ processRequest(HttpServletRequest,HttpServletResponse) : protected void + doGet(HttpServletRequest,HttpServletResponse) : protected void + doPost(HttpServletRequest,HttpServletResponse) : protected void
RatingControl.java
-
+ processRequest(HttpServletRequest,HttpServletResponse) : protected void + doGet(HttpServletRequest,HttpServletResponse) : protected void + doPost(HttpServletRequest,HttpServletResponse) : protected void

AdminControl.java	
-	
+ processRequest(HttpServletRequest,HttpServletResponse) : protected void	
+ doGet(HttpServletRequest,HttpServletResponse) : protected void	
+ doPost(HttpServletRequest,HttpServletResponse) : protected void	

UserControl.java	
-	
+ processRequest(HttpServletRequest,HttpServletResponse) : protected void	
+ doGet(HttpServletRequest,HttpServletResponse) : protected void	
+ doPost(HttpServletRequest,HttpServletResponse) : protected void	

4.4.3.3 Kelas Model

Terdapat 9 kelas *model*, berikut detail dari kelas *model* yang terdapat pada diagram kelas desain.

ObjekWisata.java	
- id_objek_wisata : int	
- nama_objek_wisata : String	
- alamat_objek_wisata : String	
- conn : Connection	
+ ObjekWisata() <<konstruktor>>	
+ ObjekWisata(Connection) <<konstruktor>>	
+ ObjekWisata(String,String,String,String) <<konstruktor>>	
+ getId_objek_wisata() : int	
+ setId_objek_wisata(int) : void	
+ getNama_objek_wisata() : String	
+ setNama_objek_wisata(String) : void	
+ getAlamat_objek_wisata() : String	
+ setAlamat_objek_wisata(String) : void	
+ IdObjekWisataOtomatis() : String	
+ inputObjekWisata() : void	
+ updateObjekWisata() : void	
+ HapusOW() : void	
+ getAmbilOW() : List<ObjekWisata>	
+ getAmbil1OW(int) : String	

Admin.java
- nama_adm : String - username_adm : String - password_adm : String - id_adm : int - email_adm : String - conn : Connection
+ Admin() <<konstruktor>> + getConn() : Connection + setConn(Connection) : void + getEmail_adm() : String + setEmail_adm(String) : void + getNama_adm() : String + setNama_adm(String) : void + getUsername_adm() : String + setUsername_adm(String) : void + getPassword_adm() : String + setPassword_adm(String) : void + getId_adm() : int + setId_adm(int) : void + IdAdminOtomatis() : String + inputAdm() : void + UpdateAdm() : void + HapusAdm() : void + getAmbilAdmin() : List<Admin> + ValidasiAdminLogin() : boolean + ValidasiUsernameAdmin() : boolean

MAE.java
- MAE : double - listPrediksi : List<Prediksi> - conn : Connection
+ MAE() <<konstruktor>> + MAE(Connection) <<konstruktor>> + getKoneksi() : MAE + getMAE() : double + setMAE(double) : void + getListPrediksi() : List<Prediksi> + setListPrediksi(List<Prediksi>) : void + showPrediction(int) : MAE

ItemBased.java
<pre>- id_similarity_objek_wisata : int - id_objek_wisata1 : int - id_objek_wisata2 : int - nilai_similarity : Double - conn : Connection + ItemBased() <<konstruktor>> + ItemBased(Connection) <<konstruktor>> + getKoneksi() : ItemBased + getId_similarity_Objek_wisata() : int + setId_similarity_Objek_wisata(int) : void + getId_objek_wisata1() : int + setId_objek_wisata1(int) : void + getId_objek_wisata2() : int + setId_objek_wisata2(int) : void + getNilai_similarity() : Double + setNilai_similarity(Double) : void + getRating(String) : List<Rating> + getAverage(String) : double + getAtas(List<Rating>,List<Rating>,double,double) : double + getBawah(List<Rating>,List<Rating>,double,double) : double + similaritasPearsonCorrelation(String,String) : double + InputSimilarity(String,String,ItemBased) : void + getSimilarity1ObjekWisata(String) : List<ItemBased> + getSimilarity() : List<ItemBased> + similarity() : void</pre>

Connector.java
<pre>- conn : Connection + getKoneksi() : Connection</pre>

Prediksi.java
<pre>- conn : Connection - rating_training : String - ratingId : int - topN : int - prediksiValue : Double - objek1 : String - objek2 : String - similarityValue : Double - userid : int</pre>
<pre>+ Prediksi() <<konstruktor>> + Prediksi(Connection) <<konstruktor>> + getKoneksi() : Prediksi + getPrediksiValue() : Double + setPrediksiValue(Double) : void + getUserId() : int + setUserid(int) : void + getRatingId() : int + setRatingId(int) : void + getTopN() : int + setTopN(int) : int + getPredictionValue() : Double + setPredictionValue(Double) : void + getObjek1() : String + setObjek1(String) : void + getObjek2() : String + setObjek2(String) : void + getSimilarityValue() : Double + setSimilarityValue(Double) : void + getRating_training() : String + setRating_training(String) : void + getRating() : List<Rating> + getRatingTraining(int) : List<Rating> + getRatingAndOWSimilaritySort(String, String) : List<Prediksi> + hasilPrediksiItemBased(Rating,int) : double + inputPrediksi(Prediksi) : void + getPrediksiRatingUser(int) : List<Prediksi> + getPrediksiRatingUserdiAdmin(int,int) : List<Prediksi> + prediksiTesting(int) : void + prediksiRating(int) : void</pre>

Rating.java
<pre> - id_user : int - id_objek_wisata : int - rating : String - id_rating : int - conn : Connection + Rating() <<konstruktor>> + Rating(Connection) <<konstruktor>> + getKoneksi() : Rating + getId_user() : int + setId_user(int) : void + getId_objek_wisata() : int + setId_objek_wisata(int) : void + getRating() : String + setRating(String) : void + getId_rating() : int + setId_rating(int) : void + inputRating(String, String, String) : void + inputRatingUser(String, String, String) : void + getAmbilRating(String, String) : List<Rating> + getRatingTesting() : List<Rating> + getTotalUserRated() : int + getUserId(String) List<Rating> + inputDataRatingTesting(Rating) : void + pilihDataTesting() : void </pre>

4.5 Perancangan *Output*

Pada perangkat lunak yang akan dibangun terdapat *output* yang dihasilkan untuk aktor *user* dan aktor *admin*. Terdapat beberapa *output* yang akan dihasilkan oleh sistem berbasis web untuk aktor admin, yaitu:

1. Tampilan data *user*.
2. Tampilan data *admin*.
3. Tampilan data objek wisata.
4. Tampilan nilai *similarity* antar objek wisata.
5. Tampilan nilai prediksi objek wisata dari data *testing* dan dari objek wisata yang belum di-*rating* oleh *user*.
6. Nilai kesalahan dari hasil prediksi terhadap *rating* asli dengan MAE.

Terdapat beberapa *output* yang akan dihasilkan sistem berbasis web untuk aktor *user*, yaitu:

1. Tampilan data objek wisata yang telah di-*rating*, prediksi *rating* objek wisata dari data *testing* dan prediksi *rating* objek wisata yang belum di-*rating user*.
2. Tampilan data *user*.

Berikut merupakan desain *interface* yang terdapat pada sistem yang dapat membantu aktor dalam menggunakan sistem yang akan dibangun, baik berupa fitur pengolahan data maupun hasil perhitungan algoritma.

4.5.1 Interface Aktor User

4.1.1.1 Halaman Login

Halaman ini merupakan halaman yang digunakan oleh *user* yang telah membuat akun untuk masuk ke dalam sistem dan menggunakan sistem. Rancangan halaman *login* dapat dilihat pada gambar 4.12 berikut ini.

SISTEM REKOMENDASI OBJEK WISATA YOGYAKARTA	
LOGIN USER	
Username <input type="text"/>	
Password <input type="password"/>	
<input type="button" value="submit"/> <input type="button" value="reset"/>	
Belum punya akun ? <input type="button" value="Sign Up"/>	
Admin ? <input type="button" value="Login Admin"/>	

Gambar 4.12 Halaman login pada sistem

4.1.1.2 Halaman Utama

Halaman ini adalah halaman yang pertama ditampilkan setelah *user* login di halaman login, sehingga *user* dapat menggunakan sistem. Rancangan halaman utama dapat dilihat pada gambar 4.13 berikut ini.

Gambar 4.13 Halaman utama aktor *user* pada sistem

4.1.1.3 Halaman Hasil Prediksi

Halaman ini merupakan halaman yang menampilkan *rating* objek wisata yang telah di-*rating* oleh *user* dan prediksi *rating* dari data *testing* maupun prediksi *rating* dari objek wisata yang belum di-*rating* oleh *user*. Rancangan halaman hasil prediksi dapat dilihat pada gambar 4.14 berikut ini.

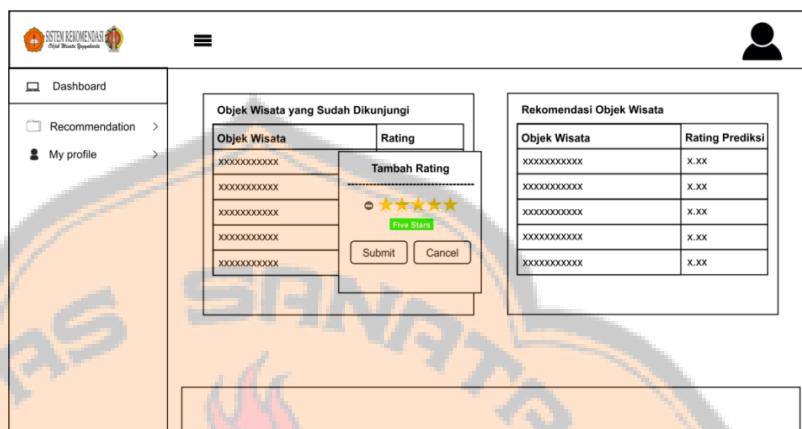
Objek Wisata yang Sudah Dikunjungi	
Objek Wisata	Rating
xxxxxxxxxx	x.xx

Rekomendasi Objek Wisata	
Objek Wisata	Rating Prediksi
xxxxxxxxxx	x.xx

Gambar 4.14 Halaman hasil prediksi pada sistem

Pada halaman hasil prediksi *user* dapat menambahkan *rating* objek wisata pada list *rating* objek wisata prediksi yang belum di-*rating* namun telah dikunjungi dengan meng-klik

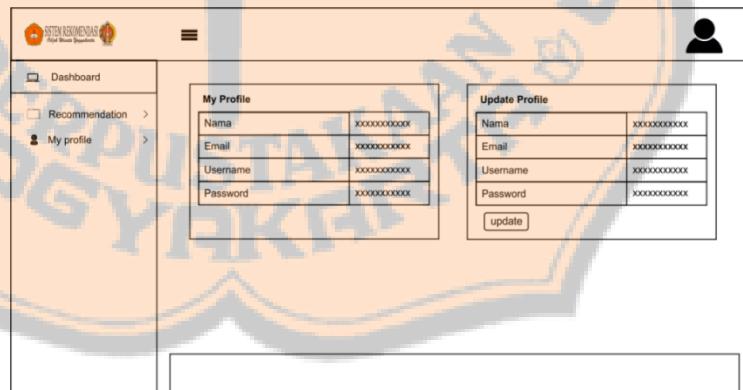
rating prediksi objek wisata. Jika *rating* prediksi ditekan maka akan ditampilkan *popup* untuk menambahkan *rating* baru. Rancangan *popup* dapat dilihat pada gambar 4.15 berikut ini.



Gambar 4.15 Halaman hasil prediksi dengan *popup* pada sistem

4.1.1.4 Halaman *Update Data*

Halaman ini menampilkan data *user*, *user* dapat meng-*update* datanya pada halaman ini. Rancangan halaman *update* data dapat dilihat pada gambar 4.16 berikut ini.



Gambar 4.16 Halaman *update* data pada sistem

4.5.2 Interface Aktor Admin

4.5.2.1 Halaman Login Admin

Halaman ini merupakan halaman yang digunakan oleh *admin* untuk masuk ke dalam sistem dan menggunakan sistem. Rancangan halaman *login admin* dapat dilihat pada gambar 4.17 berikut ini.

SISTEM REKOMENDASI OBJEK WISATA YOGYAKARTA

LOGIN ADMIN

Username
Password

submit reset

Gambar 4.17 Halaman *login admin* pada sistem

4.5.2.2 Halaman Utama

Halaman ini adalah halaman yang pertama ditampilkan setelah *admin login* di halaman login *admin*, sehingga *admin* dapat menggunakan sistem. Rancangan halaman utama dapat dilihat pada gambar 4.18 berikut ini.



Gambar 4.18 Halaman utama aktor *admin* pada sistem

4.5.2.3 Halaman Tambah Admin

Pada halaman ini *admin* dapat membuat hak akses untuk *admin* lain dengan menambahkan data *admin* lainnya. Halaman ini juga menampilkan tabel data *admin* yang memiki hak akses ke dalam sistem. Pada halaman ini *admin* yang sedang *login* dapat menghapus admin lain dengan menekan tombol hapus di tabel *admin*, namun tidak bisa menghapus akun *admin* itu sendiri. Rancangan halaman ini dapat dilihat pada gambar 4.19 berikut ini.

No	Id Admin	Nama Admin	Email Admin	Username	Password
XX	xxxxxxxxxxxxxx	xxxxxxxxxxxxxx	xxxxxxxxxxxxxx	xxxxxxxxxxxxxx	xxxxxxxxxxxxxx

Gambar 4.19 Halaman tambah *admin* pada sistem

4.5.2.4 Halaman Edit Admin

Halaman ini menampilkan data *admin* yang akan di-*edit* pada kolom *input* dan menampilkan data *admin* yang sedang *login* dan *admin* lain pada tabel *admin*. Rancangan halaman ini dapat dilihat pada gambar 4.20 berikut ini.

No	Id Admin	Nama Admin	Email Admin	Username	Password
XX	xxxxxxxxxxxxxx	xxxxxxxxxxxxxx	xxxxxxxxxxxxxx	xxxxxxxxxxxxxx	xxxxxxxxxxxxxx

Gambar 4.20 Halaman *edit admin* pada sistem

4.5.2.5 Halaman Tambah User

Pada halaman ini *admin* dapat menambah dan menghapus data *user*. Halaman ini juga menampilkan data dari semua *user* yang memiliki hak akses ke sistem. Rancangan halaman ini dapat dilihat pada gambar 2.21 berikut ini.

No	Id User	Nama User	Email User	Username	Password	edit	hapus
XX	xx	xxxxxxxxxxxxxx	xxxxxxxxxxxxxx	xxxxxxxxxxxxxx	xxxxxxxxxxxxxx	edit	hapus

Gambar 2.21 Halaman tambah *user* pada sistem

4.5.2.6 Halaman Edit User

Pada halaman ini sistem menampilkan data *user* yang akan di-edit di form *input* dan *admin* dapat mengedit data *user*, sistem menampilkan data *user* pada tabel *user*. Rancangan halaman ini dapat dilihat pada gambar 4.22 berikut ini.

No	Id User	Nama User	Email User	Username	Password	edit
XX	xx	xxxxxxxxxxxxxx	xxxxxxxxxxxxxx	xxxxxxxxxxxxxx	xxxxxxxxxxxxxx	edit

Gambar 4.22 Halaman *edit user* pada sistem

4.5.2.7 Halaman Tambah Objek Wisata

Halaman ini menampilkan data objek wisata di tabel objek wisata. *Admin* dapat menambahkan data objek wisata di form *input* dan menghapus objek wisata dengan menekan tombol hapus di tabel objek wisata. Rancangan halaman ini dapat dilihat pada tabel 4.23 berikut ini.

No	Id Objek Wisata	Nama Objek Wisata	Alamat Objek Wisata	Action
1	XX	xxxxxxxxxxxxxx	xxxxxxxxxxxxxx	edit hapus
2	XX	xxxxxxxxxxxxxx	xxxxxxxxxxxxxx	edit hapus
3	XX	xxxxxxxxxxxxxx	xxxxxxxxxxxxxx	edit hapus
4	XX	xxxxxxxxxxxxxx	xxxxxxxxxxxxxx	edit hapus

Gambar 4.23 Halaman tambah objek wisata di sistem

4.5.2.8 Halaman Edit Objek Wisata

Halaman ini menampilkan data objek wisata di tabel objek wisata. Sistem menampilkan data objek wisata yang akan di-edit di form *input* dan admin dapat mengedit data objek wisata. Rancangan halaman ini dapat dilihat pada gambar 4.24 berikut ini.

No	Id Objek Wisata	Nama Objek Wisata	Alamat Objek Wisata	Action
1	XX	xxxxxxxxxxxxxx	xxxxxxxxxxxxxx	edit
2	XX	xxxxxxxxxxxxxx	xxxxxxxxxxxxxx	edit
3	XX	xxxxxxxxxxxxxx	xxxxxxxxxxxxxx	edit
4	XX	xxxxxxxxxxxxxx	xxxxxxxxxxxxxx	edit

Gambar 4.24 Halaman edit objek wisata pada sistem

4.5.2.9 Halaman Hitung *Similarity*

Halaman ini merupakan halaman yang digunakan untuk menghitung *similarity* antar objek wisata dengan menekan tombol “hitung *similarity*”. Halaman ini juga menampilkan *similarity* objek wisata yang telah dihitung sebelumnya di tabel *similarity*. Rancangan halaman ini dapat dilihat pada gambar 4.25 dan gambar 4.26 berikut ini.

No	Id Similarity	Objek Wisata 1	Objek Wisata 2	Similarity
X	X	XXXXXXXXXXXXXX	XXXXXXXXXXXXXX	X.XX
X	X	XXXXXXXXXXXXXX	XXXXXXXXXXXXXX	X.XX
X	X	XXXXXXXXXXXXXX	XXXXXXXXXXXXXX	X.XX
X	X	XXXXXXXXXXXXXX	XXXXXXXXXXXXXX	X.XX
X	X	XXXXXXXXXXXXXX	XXXXXXXXXXXXXX	X.XX
X	X	XXXXXXXXXXXXXX	XXXXXXXXXXXXXX	X.XX
X	X	XXXXXXXXXXXXXX	XXXXXXXXXXXXXX	X.XX
X	X	XXXXXXXXXXXXXX	XXXXXXXXXXXXXX	X.XX

Nilai Similarity antar objek wisata belum dihitung, untuk menghitung tekan tombol “hitung similarity” pada halaman ini.

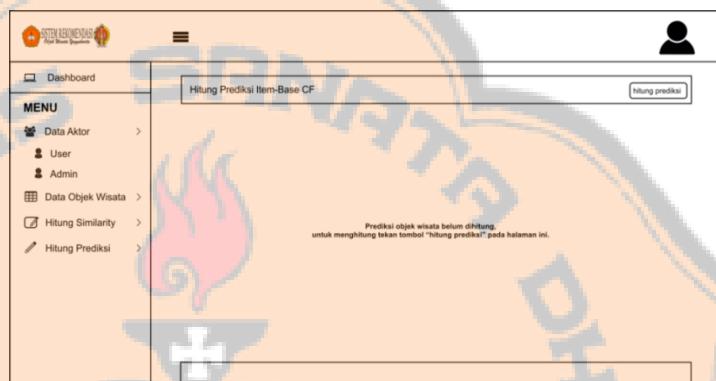
Gambar 4.25 Halaman hitung *similarity* yang *similarity* objek wisata belum dihitung

No	Id Similarity	Objek Wisata 1	Objek Wisata 2	Similarity
X	X	XXXXXXXXXXXXXX	XXXXXXXXXXXXXX	X.XX
X	X	XXXXXXXXXXXXXX	XXXXXXXXXXXXXX	X.XX
X	X	XXXXXXXXXXXXXX	XXXXXXXXXXXXXX	X.XX
X	X	XXXXXXXXXXXXXX	XXXXXXXXXXXXXX	X.XX
X	X	XXXXXXXXXXXXXX	XXXXXXXXXXXXXX	X.XX
X	X	XXXXXXXXXXXXXX	XXXXXXXXXXXXXX	X.XX
X	X	XXXXXXXXXXXXXX	XXXXXXXXXXXXXX	X.XX
X	X	XXXXXXXXXXXXXX	XXXXXXXXXXXXXX	X.XX
X	X	XXXXXXXXXXXXXX	XXXXXXXXXXXXXX	X.XX

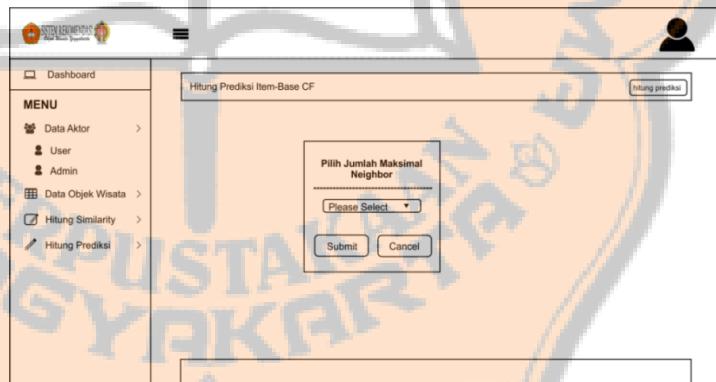
Gambar 4.26 Halaman hitung *similarity* yang *similarity* objek wisata setelah dihitung

4.5.2.10 Halaman Hitung Prediksi

Halaman ini digunakan untuk menghitung prediksi *rating* objek wisata dengan menekan tombol “hitung prediksi”. Sistem akan memunculkan menu *popup* untuk memilih data prediksi dan jumlah maksimum *neighbor* jika *admin* menekan tombol “hitung prediksi”. Rancangan halaman dan *popup* ini dapat dilihat pada gambar 4.27 dan gambar 4.28 berikut ini.



Gambar 4.27 Halaman hitung prediksi pada sistem



Gambar 4.28 Halaman hitung prediksi dengan *popup*

Setelah *admin* memilih data pada menu *popup*, maka sistem menghitung prediksi *rating* objek wisata dan menampilkan *rating* prediksi objek wisata berdasarkan maksimum *neighbor* yang dipilih. Rancangan halaman hitung

prediksi hasil hitung prediksi dapat dilihat pada gambar 4.29 berikut ini.

No	id Prediksi	Name Objek Wisata	Name User	Rating Asli	Rating Prediksi
XX	XXX	XXXXXXXXXXXXXX	XXXXXXXXXXXXXX	XXX	XXX
XX	XXX	XXXXXXXXXXXXXX	XXXXXXXXXXXXXX	XXX	XXX
XX	XXX	XXXXXXXXXXXXXX	XXXXXXXXXXXXXX	XXX	XXX
XX	XXX	XXXXXXXXXXXXXX	XXXXXXXXXXXXXX	XXX	XXX
XX	XX	XXXXXXXXXXXXXX	XXXXXXXXXXXXXX	XXX	XXX

Gambar 4.29 Halaman hitung prediksi hasil prediksi

Pada perhitungan prediksi selesai dilakukan, sistem menampilkan MAE berdasarkan perhitungan akumulasi selisih *rating testing* dengan *rating* prediksi dibagi dengan jumlah data.

4.5.3 Interface untuk Aktor *User* dan *Admin*

3.4.3.1 Halaman Signup

Halaman ini digunakan oleh *user* dan *admin* untuk memasukkan data *user* baru dan membuat akun pada sistem. Rancangan halaman ini dapat dilihat pada gambar 4.30 berikut ini.

SISTEM REKOMENDASI OBJEK WISATA YOGYAKARTA

SIGN UP

Nama
Email
Username
Password

sign up reset

Gambar 4.30 Rancangan halaman signup

3.4.3.2 Halaman Rating

Halaman ini digunakan untuk memberikan *rating* terhadap objek wisata agar *profile user* terbentuk. Halaman ini ditampilkan setelah *user* sukses *signup* dan langsung diberikan survey untuk me-*rating* objek wisata. Rancangan halaman ini dapat dilihat pada gambar 4.31 berikut ini.

SISTEM REKOMENDASI OBJEK WISATA YOGYAKARTA

Gambar 4.31 Rancangan halaman rating

BAB V IMPLEMENTASI DAN ANALISA HASIL

Berikut ini penjabaran tentang implementasi yang dilakukan penulis dalam penelitian ini beserta dengan langkah-langkah mengenai penerapan dari perancangan.

5.1 Implementasi Basisdata

5.1.1 Tabel user

```
CREATE TABLE `user` (
  `id_user` INT( 10 ) NOT NULL PRIMARY KEY ,
  `nama_user` VARCHAR( 100 ) NOT NULL ,
  `email` VARCHAR( 100 ) NOT NULL ,
  `username` VARCHAR( 50 ) NOT NULL ,
  `password` VARCHAR( 50 ) NOT NULL) ENGINE = InnoDB;
```

5.1.2 Tabel admin

```
CREATE TABLE `admin` (
  `id_adm` INT( 10 ) NOT NULL PRIMARY KEY ,
  `nama_adm` VARCHAR( 100 ) NOT NULL ,
  `email_adm` VARCHAR( 100 ) NOT NULL ,
  `username_adm` VARCHAR( 50 ) NOT NULL ,
  `password_adm` VARCHAR( 50 ) NOT NULL) ENGINE = InnoDB;
```

5.1.3 Tabel objek_wisata

```
CREATE TABLE `objek_wisata` (
  `id_objek_wisata` INT( 10 ) NOT NULL PRIMARY KEY ,
  `nama_objek_wisata` VARCHAR( 100 ) NOT NULL ,
  `alamat_objek_wisata` VARCHAR( 250 ) NOT NULL)
ENGINE = InnoDB;
```

5.1.4 Tabel rating

```
CREATE TABLE `rating`(  
    `id_rating` INT(10) NOT NULL AUTO_INCREMENT PRIMARY  
    KEY, `id_user` INT(10) NOT NULL, `id_objek_wisata` INT(10) NOT  
    NULL, `rating` DOUBLE(5,2), `data_type` INT(2),  
    FOREIGN KEY id_user_fk(id_user) REFERENCES USER(id_user),  
    FOREIGN KEY id_objek_wisata_fk(id_objek_wisata) REFERENCES  
    objek_wisata(id_objek_wisata) ON DELETE CASCADE ON  
    UPDATE CASCADE) ENGINE = InnoDB;
```

5.1.5 Tabel similarity

```
CREATE TABLE `similarity`(  
    `id_similarity` INT(10) NOT NULL AUTO_INCREMENT PRIMARY  
    KEY, `id_objek_wisata_1` INT(10) NOT NULL,  
    `id_objek_wisata_2` INT(10) NOT NULL,  
    `nilai_similarity` DOUBLE(5,2),  
    FOREIGN KEY fk_id_objek_wisata_1(id_objek_wisata_1)  
    REFERENCES objek_wisata(id_objek_wisata),  
    FOREIGN KEY fk_id_objek_wisata_2(id_objek_wisata_2)  
    REFERENCES objek_wisata(id_objek_wisata) ON DELETE  
    CASCADE ON UPDATE CASCADE) ENGINE = InnoDB;
```

5.1.6 Tabel rating_prediksi

```
CREATE TABLE `rating_prediksi`(  
    `id_rating_prediksi` INT(10) NOT NULL AUTO_INCREMENT  
    PRIMARY KEY, `id_objek_wisata` INT(10) NOT NULL,  
    `id_user` INT(10) NOT NULL, `rating_prediksi` DOUBLE(5,2),  
    `TopN` INT(5), FOREIGN KEY fk_id_userfk(id_user) REFERENCES  
    USER(id_user), FOREIGN KEY  
    fk_id_objek_wisatafk(id_objek_wisata) REFERENCES  
    objek_wisata(id_objek_wisata) ON DELETE CASCADE  
    ON UPDATE CASCADE) ENGINE = InnoDB;
```

5.2 Implementasi Proses

Bagian implementasi proses menjelaskan tentang penerapan dari perancangan yang telah dibuat pada bab sebelumnya. Perancangan tersebut berupa algoritma dari blok proses yang dimuat pada gambar *flowchart* diagram pada gambar 4.6.

5.2.1 Blok Proses pada Kelas

Implementasi blok proses pada *flowchart* diagram pada gambar 4.6 akan dijabarkan pada tabel 5.1 berikut ini.

Tabel 5.1 Implementasi blok proses pada kelas java

No	Nama Blok Proses	Kelas Java
1	Memilih data <i>testing</i>	Rating.java
2	Menghitung <i>similarity</i>	ItemBased.java
3	Memprediksi <i>rating</i>	Prediksi.java
4	Menghitung MAE	MAE.java

5.2.2 Implementasi Blok Proses dalam Koding

1. Memilih data *testing*

Berikut ini merupakan implementasi blok proses “memilih data *testing*” pada *source code* kelas Rating.java, seperti yang dijabarkan pada gambar 5.1 sampai dengan gambar 5.4 berikut ini.

```

140 public int getTotalUserRated() throws SQLException {
141     PreparedStatement prep = null;
142     ResultSet res = null;
143     conn.setAutoCommit(false);
144     String sql = "SELECT count(*) FROM `rating_training` GROUP by id_user";
145     prep = conn.prepareStatement(sql);
146     res = prep.executeQuery();
147
148     int totalUser = 0;
149     while (res.next()) {
150         totalUser++;
151     }
152     conn.commit();
153     conn.close();
154     return totalUser;
155 }
```

Gambar 5.1 *Source code method* getTotalUserRated()

```

167 public List<Rating> getUserbyId(String id_user) throws SQLException {
168     PreparedStatement prep = null;
169     ResultSet res = null;
170     conn.setAutoCommit(false);
171     String sql = "select * from rating where id_user = " + id_user + " and data_type = 0 order by id_user";
172     prep = conn.prepareStatement(sql);
173     res = prep.executeQuery();
174
175     List<Rating> rat = new ArrayList<Rating>();
176
177     while (res.next()) {
178         Rating rt = new Rating();
179         rt.setId_rating(res.getInt("id_rating"));
180         rt.setId_user(res.getInt("id_user"));
181         rt.setId_objek_wisata(res.getInt("id_objek_wisata"));
182         rt.setRating(String.valueOf(res.getString("rating")));
183         rat.add(rt);
184     }
185
186     conn.commit();
187     conn.close();
188     return rat;
189 }
```

Gambar 5.2 *Source code method getUserbyId(String)*

```

191 public void UpdateDataRatingType(Rating rat) throws SQLException {
192     PreparedStatement prep = null;
193     conn.setAutoCommit(false);
194
195     String sql = "UPDATE rating set data_type = 1 where id_user = "+rat.getId_user();
196     prep = conn.prepareStatement(sql);
197     prep.executeUpdate();
198
199     conn.commit();
200     conn.close();
201 }
```

Gambar 5.3 *Source code method UpdateDataRatingType(Rating)*

```

187 public void pilihDataTesting() throws SQLException {
188     Rating r = new Rating();
189     List<Rating> rat = new ArrayList<Rating>();
190     int totalUser = r.getKoneksi(). getTotalUserRated();
191     int totalDataRating = 0;
192     Random random = new Random();
193     for (int i = 1; i <= totalUser; i++) {
194         rat = r.getKoneksi().getUserbyId(String.valueOf(i));
195         if (rat.size() > 6) {
196             int ranawal = 0;
197             int looping = 0;
198             while (looping != 2) {
199                 int ran = random.nextInt(rat.size());
200                 if (ran != ranawal) {
201
202                     r.setId_rating(rat.get(ran).getId_rating());
203                     r.setId_user(rat.get(ran).getId_user());
204                     r.setId_objek_wisata(rat.get(ran).getId_objek_wisata());
205                     r.setRating(rat.get(ran).getRating());
206
207                     r.getKoneksi().inputDataRatingTesting(r);
208                     System.out.println("Data rating ke - " + (looping + 1) + ", user ke - " + i + " : done");
209                     totalDataRating++;
210                     ranawal = ran;
211                     looping++;
212                 }
213             }
214         }
215     }
216     r.conn.close();
217     System.out.println("Total data rating : " + totalDataRating);
218 }
```

Gambar 5.4 *Source code method pilihDataTesting()*

Setelah semua *method* untuk memilih data *testing* dibuat, maka penulis membuat *method* main untuk *running method* sebelumnya. *Method* main dapat dilihat pada gambar 5.5 berikut ini.

```
241  public static void main(String[] args) throws SQLException {  
242      Rating rating = new Rating();  
243      rating.getKoneksi().pilihDataTesting();  
244      rating.getKoneksi().conn.close();  
245  }
```

Gambar 5.5 Source code method main(String[])

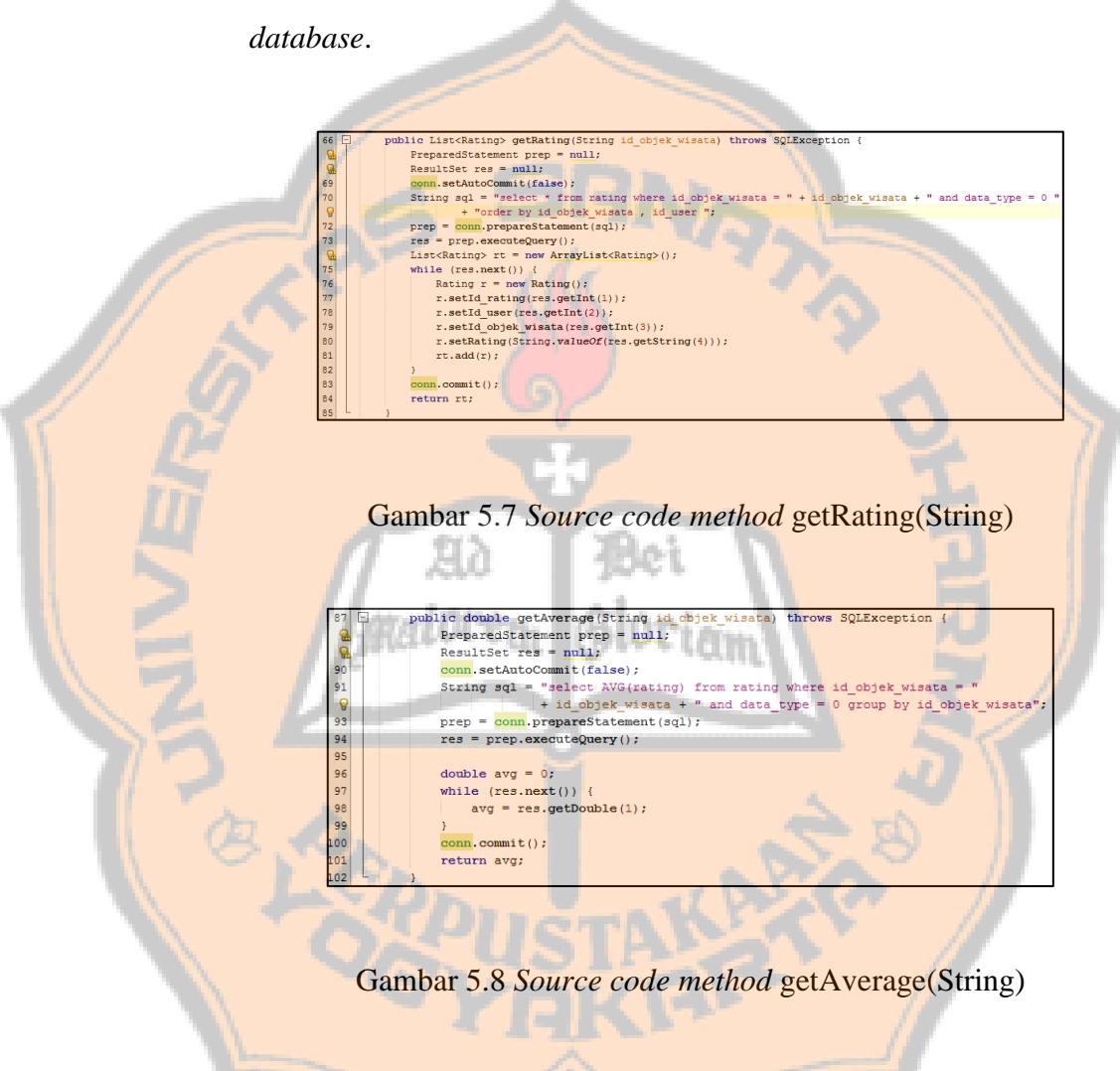
Setelah membuat *method* main, penulis melakukan *running* dan mendapat sebanyak 140 data *testing* dengan waktu *running* selama 17 detik. Lama waktu running dapat dilihat pada gambar 5.6 berikut.

```
Data rating ke - 1, user ke - 89 : done  
Data rating ke - 2, user ke - 89 : done  
Data rating ke - 1, user ke - 90 : done  
Data rating ke - 2, user ke - 90 : done  
Data rating ke - 1, user ke - 91 : done  
Data rating ke - 2, user ke - 91 : done  
Data rating ke - 1, user ke - 93 : done  
Data rating ke - 2, user ke - 93 : done  
Data rating ke - 1, user ke - 94 : done  
Data rating ke - 2, user ke - 94 : done  
Data rating ke - 1, user ke - 96 : done  
Data rating ke - 2, user ke - 96 : done  
Data rating ke - 1, user ke - 97 : done  
Data rating ke - 2, user ke - 97 : done  
Data rating ke - 1, user ke - 99 : done  
Data rating ke - 2, user ke - 99 : done  
Data rating ke - 1, user ke - 100 : done  
Data rating ke - 2, user ke - 100 : done  
Total data rating : 140  
BUILD SUCCESSFUL (total time: 17 seconds)
```

Gambar 5.6 Lama waktu *running* pemilihan data *testing*

2. Menghitung *similarity*

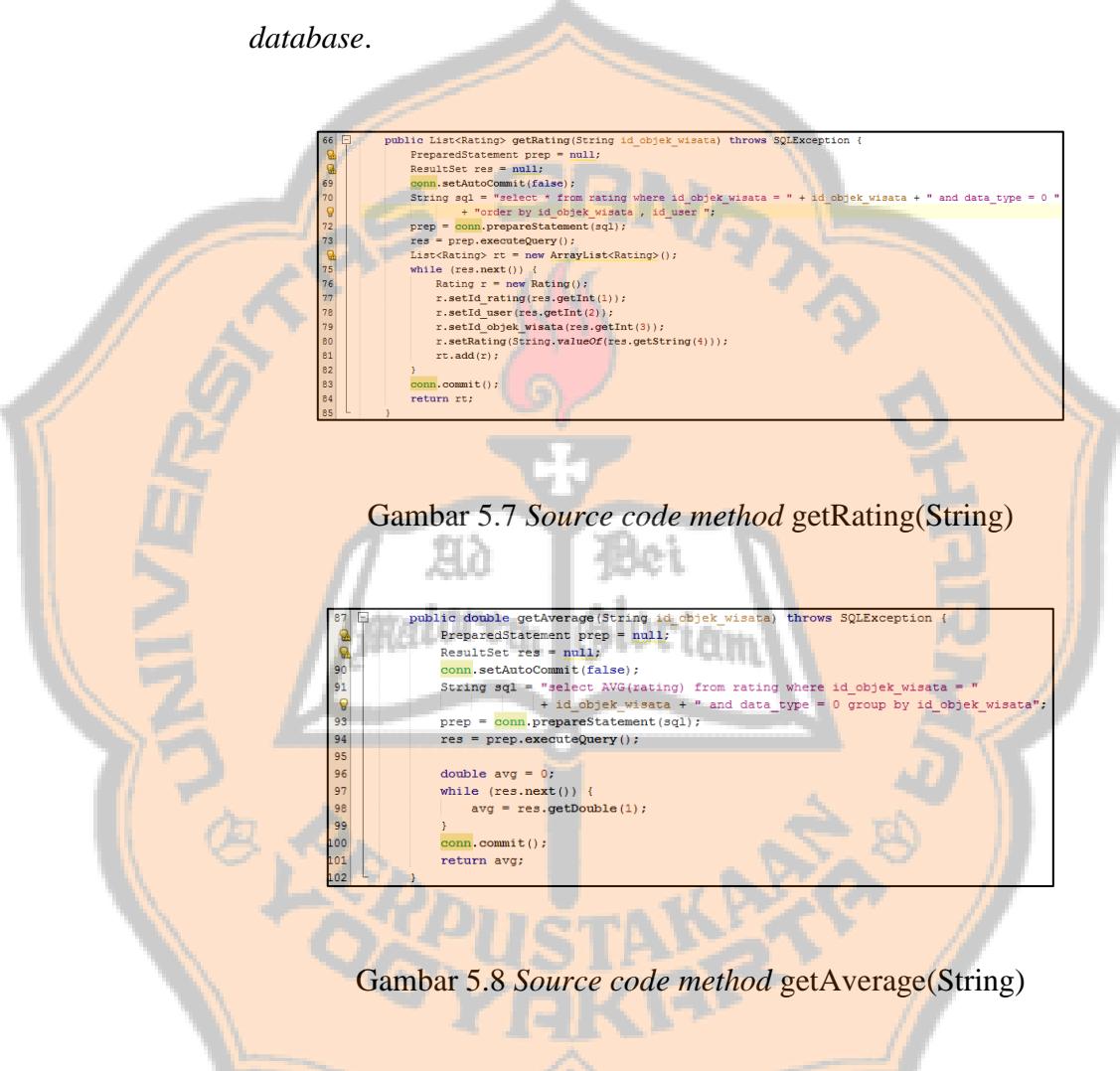
Berikut ini merupakan implementasi blok proses “menghitung *similarity*” pada source code kelas ItemBased.java. Implementasi dapat dilihat pada gambar 5.7 sampai dengan gambar 5.15 berikut ini dan gambar 5.16 untuk melihat data *similarity* yang tersimpan di *database*.



```

66 public List<Rating> getRating(String id_objek_wisata) throws SQLException {
67     PreparedStatement prep = null;
68     ResultSet res = null;
69     conn.setAutoCommit(false);
70     String sql = "select * from rating where id_objek_wisata = " + id_objek_wisata + " and data_type = 0 "
71         + "order by id_objek_wisata , id_user";
72     prep = conn.prepareStatement(sql);
73     res = prep.executeQuery();
74     List<Rating> rt = new ArrayList<Rating>();
75     while (res.next()) {
76         Rating r = new Rating();
77         r.setid_rating(res.getInt(1));
78         r.setid_user(res.getInt(2));
79         r.setid_objek_wisata(res.getInt(3));
80         r.setRating(String.valueOf(res.getString(4)));
81         rt.add(r);
82     }
83     conn.commit();
84     return rt;
85 }
```

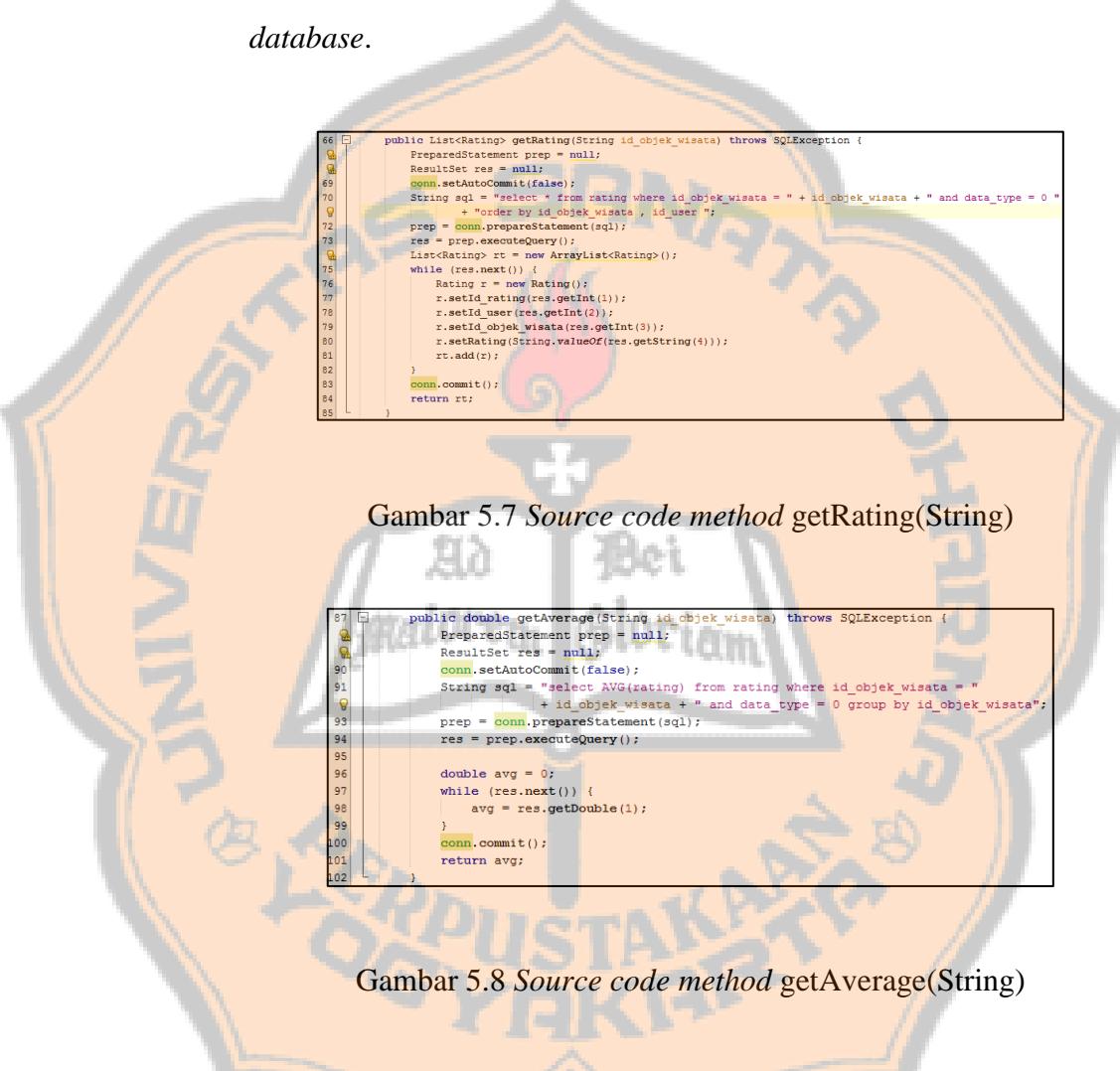
Gambar 5.7 Source code method getRating(String)



```

87 public double getAverage(String id_objek_wisata) throws SQLException {
88     PreparedStatement prep = null;
89     ResultSet res = null;
90     conn.setAutoCommit(false);
91     String sql = "select AVG(rating) from rating where id_objek_wisata = "
92         + id_objek_wisata + " and data_type = 0 group by id_objek_wisata";
93     prep = conn.prepareStatement(sql);
94     res = prep.executeQuery();
95
96     double avg = 0;
97     while (res.next()) {
98         avg = res.getDouble(1);
99     }
100    conn.commit();
101    return avg;
102 }
```

Gambar 5.8 Source code method getAverage(String)



```

104 public double qntAtas(List<Rating> objek_wisatal, List<Rating> objek_wisata2, double average1, double average2) {
105     double totalPerkalianRatingMinAvg = 0;
106     double ratingMinAvg1 = 0;
107     double ratingMinAvg2 = 0;
108     double perkalianItem = 0;
109
110     for (int i = 0; i < objek_wisatal.size(); i++) {
111         for (int j = 0; j < objek_wisata2.size(); j++) {
112             if (objek_wisatal.get(i).getId_user() == objek_wisata2.get(j).getId_user()) {
113
114                 ratingMinAvg1 = Double.valueOf(objek_wisatal.get(i).getRating()) - average1;
115                 ratingMinAvg2 = Double.valueOf(objek_wisata2.get(j).getRating()) - average2;
116                 perkalianItem = ratingMinAvg1 * ratingMinAvg2;
117                 totalPerkalianRatingMinAvg = totalPerkalianRatingMinAvg + perkalianItem;
118             }
119         }
120     }
121     return totalPerkalianRatingMinAvg;
122 }
```

Gambar 5.9 *Source code method getAtas(List<Rating>, List<Rating>,double,double)*

```

125 public double getAtas(List<Rating> objek_wisata1, List<Rating> objek_wisata2, double average1, double average2) throws SQLException {
126     double ratingMinAvg1 = 0;
127     double totalratingMinAvg1 = 0;
128
129     double ratingMinAvg2 = 0;
130     double totalratingMinAvg2 = 0;
131
132     double perkalianPangkat = 0;
133     double akarPerkalianJumlahRatingMinAvg = 0;
134
135     for (int i = 0; i < objek_wisata1.size(); i++) {
136         for (int j = 0; j < objek_wisata2.size(); j++) {
137             if (objek_wisata1.get(i).getId_user() == objek_wisata2.get(j).getId_user()) {
138
139                 ratingMinAvg1 = Math.pow(Double.valueOf(objek_wisata1.get(i).getRating()) - average1, 2);
140                 totalratingMinAvg1 = totalratingMinAvg1 + ratingMinAvg1;
141
142                 ratingMinAvg2 = Math.pow(Double.valueOf(objek_wisata2.get(j).getRating()) - average2, 2);
143                 totalratingMinAvg2 = totalratingMinAvg2 + ratingMinAvg2;
144
145             }
146         }
147     }
148
149     perkalianPangkat = totalratingMinAvg1 * totalratingMinAvg2;
150     akarPerkalianJumlahRatingMinAvg = Math.sqrt(perkalianPangkat);
151
152     return akarPerkalianJumlahRatingMinAvg;
153 }
```

Gambar 5.10 *Source code method getBawah(List<Rating>, List<Rating>,double,double)*

```

155 public double similaritasPearsonCorrelation(String objek_wisataX, String objek_wisataY) throws SQLException {
156     double hasil = 0.0;
157
158     if (objek_wisataX.equalsIgnoreCase(objek_wisataY)) {
159         hasil = 1;
160     } else {
161         List<Rating> objek_wisata1 = ItemBased.getKoneksi().getRating(objek_wisataX);
162         List<Rating> objek_wisata2 = ItemBased.getKoneksi().getRating(objek_wisataY);
163         double average1 = ItemBased.getKoneksi().getAverage(String.valueOf(objek_wisata1.get(0).getId_objek_wisata()));
164         double average2 = ItemBased.getKoneksi().getAverage(String.valueOf(objek_wisata2.get(0).getId_objek_wisata()));
165
166         double bagianAtasPC = this.getAtas(objek_wisata1, objek_wisata2, average1, average2);
167         double bagianBawahPC = this.getBawah(objek_wisata1, objek_wisata2, average1, average2);
168
169         hasil = bagianAtasPC / bagianBawahPC;
170         String NaN = null;
171         if (Double.isNaN(hasil)) {
172             hasil = 0;
173             NaN = "yes";
174         }
175     }
176     System.out.println(objek_wisataX + " x " + objek_wisataY + " = " + hasil);
177     return hasil;
178 }
```

Gambar 5.11 *Source code method similaritasPearsonCorrelation (String, String)*

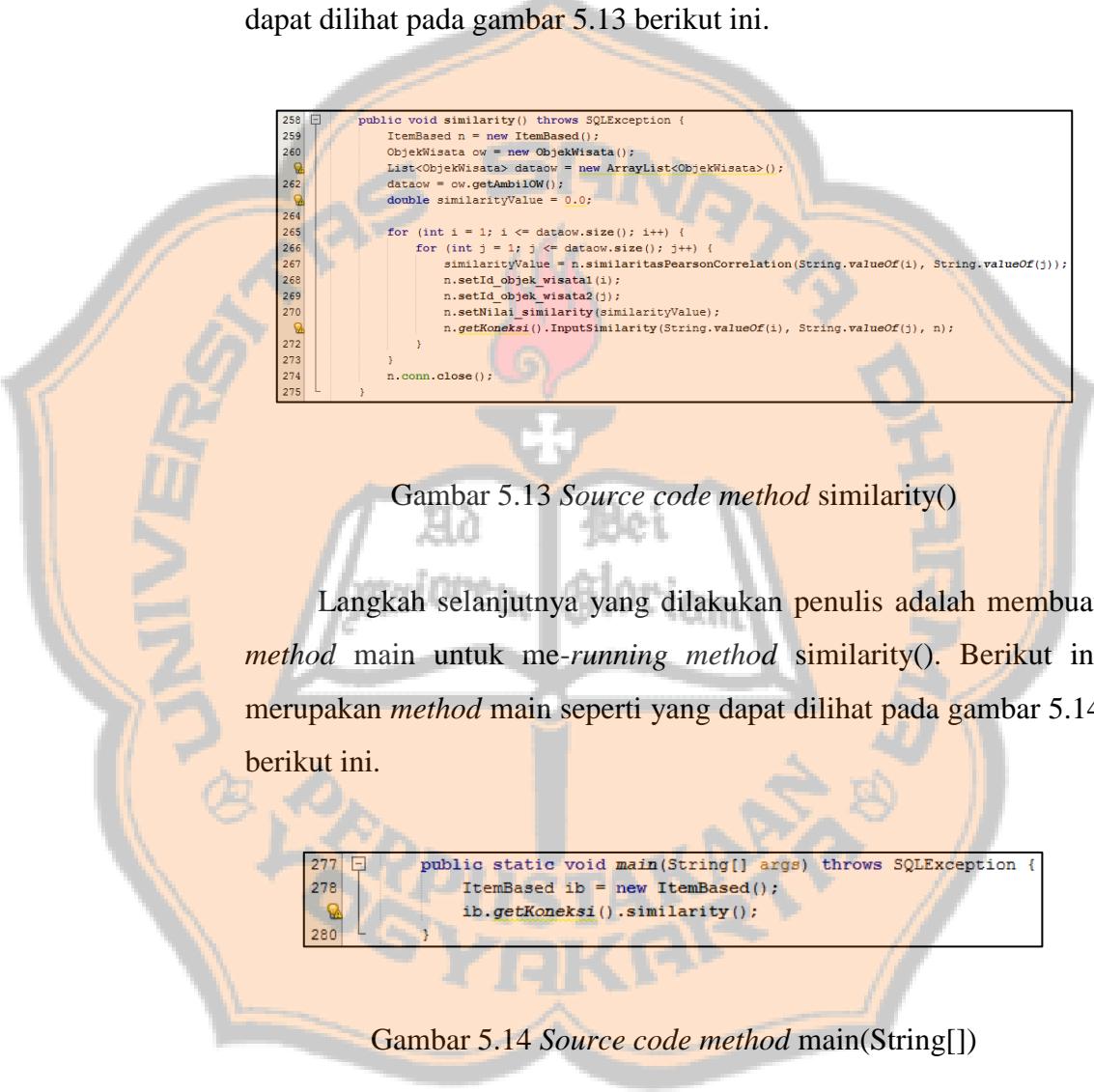
```

182 public void InputSimilarity(String objek_wisataX, String objek_wisataY, ItemBased ib) throws SQLException {
183     PreparedStatement prep = null;
184     ResultSet res = null;
185     conn.setAutoCommit(false);
186     String sql = "commit";
187     String sql1 = "select nilai_similarity from similarity where id_objek_wisata_1 =" + objek_wisataX + " "
188     + "and id_objek_wisata_2 =" + objek_wisataY + "";
189     prep = conn.prepareStatement(sql);
190     res = prep.executeQuery();
191     if (res.next()) {
192         conn.setAutoCommit(false);
193         String sql2 = "update similarity set nilai_similarity = " + ib.getNilai_similarity() + " "
194         + "where id_objek_wisata_1 =" + ib.getId_objek_wisata1() + " and id_objek_wisata_2 = "
195         + ib.getId_objek_wisata2();
196         prep = conn.prepareStatement(sql2);
197         prep.executeUpdate();
198     } else {
199         conn.setAutoCommit(false);
200         String sql3 = "insert into similarity (id_objek_wisata_1, id_objek_wisata_2, nilai_similarity)"
201         + "values (" + ib.getId_objek_wisata1() + ", " + ib.getId_objek_wisata2() + ", "
202         + ib.getNilai_similarity() + ")";
203         prep = conn.prepareStatement(sql3);
204         prep.executeUpdate();
205         conn.commit();
206     }
207     conn.commit();
208 }
```

Gambar 5.12 Source code method InputSimilarity

```
(String, String, ItemBased)
```

Setelah semua *method* untuk menghitung *similarity* dibuat, penulis membuat *method* untuk *running method* yang sudah dibuat sebelumnya. *Method* ini dapat di-*running* pada kelas *control* SimilarityControl.java untuk menghitung *similarity*. *Method* ini dapat dilihat pada gambar 5.13 berikut ini.

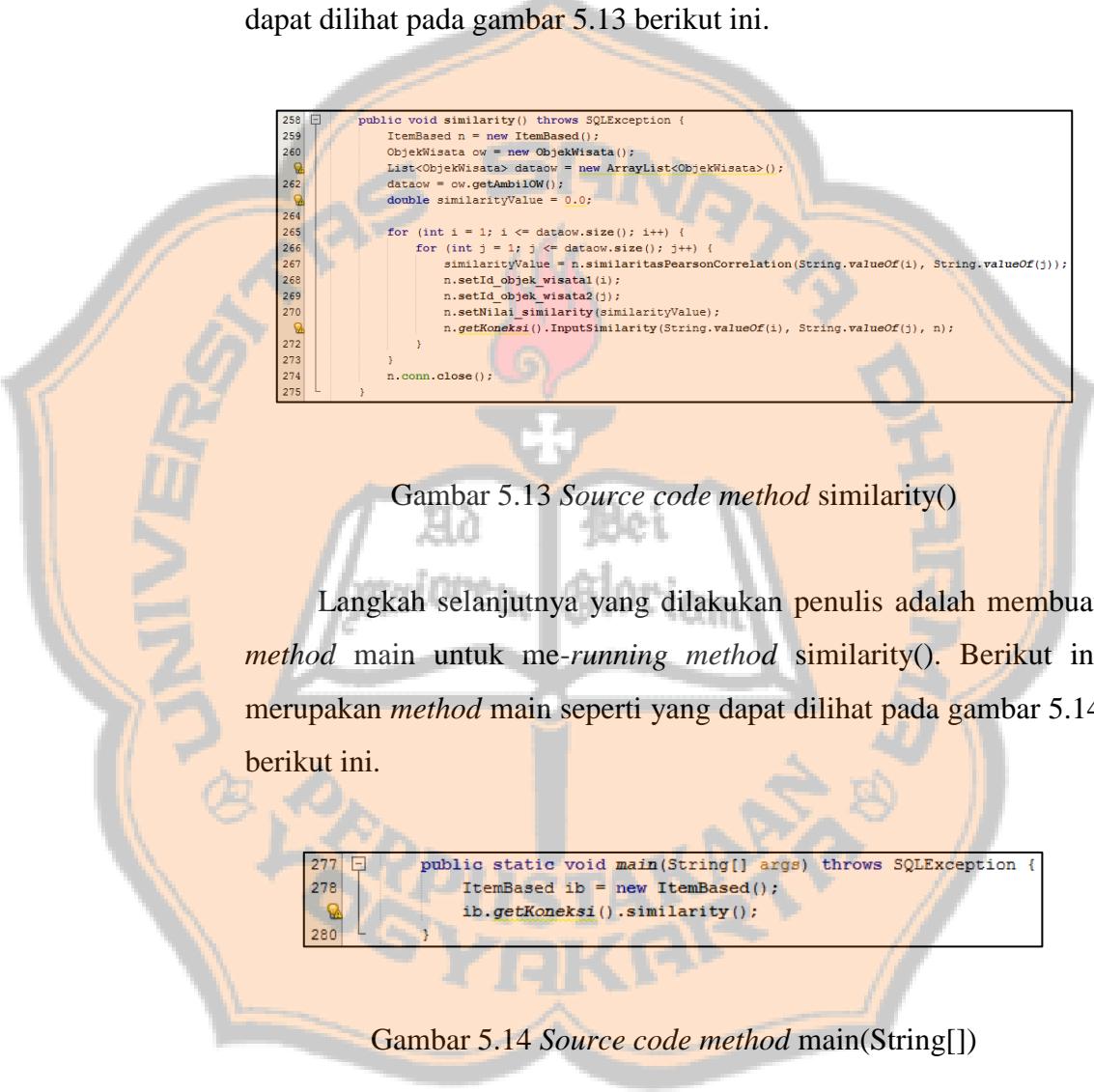


```

258 public void similarity() throws SQLException {
259     ItemBased n = new ItemBased();
260     ObjekWisata cw = new ObjekWisata();
261     List<ObjekWisata> dataow = new ArrayList<ObjekWisata>();
262     dataow = cw.getambilOW();
263     double similarityValue = 0.0;
264
265     for (int i = 1; i <= dataow.size(); i++) {
266         for (int j = 1; j <= dataow.size(); j++) {
267             similarityValue = n.similaritasPearsonCorrelation(String.valueOf(i), String.valueOf(j));
268             n.setNilai_objek_wisata1(i);
269             n.setNilai_objek_wisata2(j);
270             n.setNilai_similarity(similarityValue);
271             n.getKoneksi().InputSimilarity(String.valueOf(i), String.valueOf(j), n);
272         }
273     }
274     n.conn.close();
275 }
```

Gambar 5.13 Source code method similarity()

Langkah selanjutnya yang dilakukan penulis adalah membuat *method* main untuk me-*running method* similarity(). Berikut ini merupakan *method* main seperti yang dapat dilihat pada gambar 5.14 berikut ini.



```

277 public static void main(String[] args) throws SQLException {
278     ItemBased ib = new ItemBased();
279     ib.getKoneksi().similarity();
280 }
```

Gambar 5.14 Source code method main(String[])

Proses *running* membutuhkan waktu selama 24 detik untuk menghitung *similarity* 10 objek wisata. Total *similarity* objek wisata yang dihitung adalah $10 \times 10 = 100$ data *similarity*. Lama waktu *running* dapat dilihat pada gambar 5.15 berikut ini.

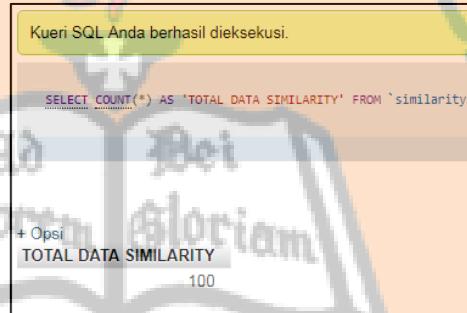
```

10 x 1 = -0.06627999012760508
10 x 2 = 0.5089019385986301
10 x 3 = 0.27359750057103127
10 x 4 = 0.2991314042728638
10 x 5 = 0.5812755977887922
10 x 6 = 0.22328228534306258
10 x 7 = 0.6212885025119195
10 x 8 = 0.020242609414757837
10 x 9 = 0.3166315831945069
10 x 10 = 1.0
BUILD SUCCESSFUL (total time: 24 seconds)

```

Gambar 5.15 Lama waktu *running* perhitungan *similarity*

Setelah *running* program selesai, penulis melakukan pengecekan terhadap jumlah data *similarity* yang tersimpan pada tabel *similarity* di *database*.



```

Kueri SQL Anda berhasil dieksekusi.

SELECT COUNT(*) AS 'TOTAL DATA SIMILARITY' FROM `similarity`

+ Opsi
TOTAL DATA SIMILARITY
100

```

Gambar 5.16 Total data *similarity* yang tersimpan di *database*

Semua data nilai *similarity* antar objek wisata telah tersimpan di tabel *similarity*, dengan total data *similarity* yang tersimpan di *database* adalah sebanyak 100 baris data.

3. Memprediksi *rating*

Berikut ini merupakan implementasi dari perancangan blok proses “memprediksi *rating*” pada *source code* di kelas Prediksi.java. Implementasi blok proses pada *source code* dapat dilihat pada gambar 5.17 sampai dengan gambar 5.21 berikut ini.

```

110 public List<Rating> getRating(int id_user) throws SQLException {
111     PreparedStatement prep = null;
112     ResultSet res = null;
113     conn.setAutoCommit(false);
114     String sql = "SELECT id_objek_wisata from objek_wisata where id_objek_wisata NOT IN "
115         + "(SELECT id_objek_wisata from rating where id_user = " + id_user + " and data_type = 0)";
116     prep = conn.prepareStatement(sql);
117     res = prep.executeQuery();
118
119     List<Rating> rat = new ArrayList<Rating>();
120
121     while (res.next()) {
122         Rating rt = new Rating();
123         rt.setId_user(id_user);
124         rt.setId_objek_wisata(res.getInt(1));
125         rat.add(rt);
126     }
127     conn.commit();
128     return rat;
129 }
```

Gambar 5.17 Source code method getRating(int)

```

131 public List<Prediksi> getRatingAndOWSimilaritySort(String id_user, String id_objek_wisata) throws SQLException {
132     PreparedStatement prep = null;
133     ResultSet res = null;
134     conn.setAutoCommit(false);
135     String sql = "SELECT * FROM similarity WHERE id_objek_wisata_1 = " + id_objek_wisata + " "
136         + "and nilai_similarity > 0 and id_objek_wisata_2 IN "
137         + "(select id_objek_wisata from rating where id_user = " + id_user + " and data_type = 0) "
138         + "order by nilai_similarity desc";
139     prep = conn.prepareStatement(sql);
140     res = prep.executeQuery();
141
142     List<Prediksi> otherOWRatingSort = new ArrayList<Prediksi>();
143
144     PreparedStatement prepl = null;
145     ResultSet res1 = null;
146     String sql1 = "";
147
148     while (res.next()) {
149         Prediksi p = new Prediksi();
150         p.setObjek1(id_objek_wisata);
151         p.setObjek2(res.getString(3));
152
153         sql1 = "select rating from rating where id_objek_wisata = " + res.getString(3) + " "
154             + "and id_user = " + id_user + " and data_type = 0";
155         prepl = conn.prepareStatement(sql1);
156         res1 = prepl.executeQuery();
157
158         while (res1.next()) {
159             p.setRating_training(String.valueOf(res1.getDouble(1)));
160         }
161         p.setSimilarityValue(res.getDouble(4));
162         otherOWRatingSort.add(p);
163     }
164     conn.commit();
165     return otherOWRatingSort;
166 }
```

Gambar 5.18 Source code method getRatingAndOWSimilaritySort
(String, String)

```

188 public double hasilPrediksiItemBased(Rating dataTraining, int topN) throws SQLException {
189     List<Prediksi> otherOWRatingSort = Prediksi.getKoneski().getRatingAndWSimilaritySort(
190         String.valueOf(dataTraining.getId_user()), String.valueOf(dataTraining.getId_objek_wisata()));
191     //mengambil semua objek wisata yang telah dirating oleh user tertentu (rating film dari user X ke cw X)
192     double NilaiSimilarity = 0;
193     int parameterLoop = 0;
194     double ratingUserForOW = 0;
195     double RatingXOWSimilarityValue = 0;
196     double totalRatingXOWSimilarityValue = 0;
197     double totalOWSimilarityValue = 0;
198     double prediksi = 0;
199
200     if (topN == 10) {
201         parameterLoop = otherOWRatingSort.size();
202     } else if (otherOWRatingSort.size() < topN) {
203         parameterLoop = otherOWRatingSort.size();
204     } else {
205         parameterLoop = topN;
206     }
207
208     System.out.println("TopN :" + parameterLoop);
209     for (int i = 0; i < parameterLoop; i++) {
210         NilaiSimilarity = otherOWRatingSort.get(i).getSimilarityValue();
211         System.out.println("ukuran similarity " + otherOWRatingSort.size());
212         if (NilaiSimilarity > 0) {
213             ratingUserForOW = Double.valueOf(otherOWRatingSort.get(i).getRating_training());
214             RatingXOWSimilarityValue = ratingUserForOW * NilaiSimilarity;
215             System.out.print((i + 1) + " ");
216             System.out.println(otherOWRatingSort.get(i).getObjek1() + " x "
217                 + otherOWRatingSort.get(i).getObjek2() + " || "
218                 + ratingUserForOW + " * " + NilaiSimilarity + " == " + RatingXOWSimilarityValue);
219
220             totalRatingXOWSimilarityValue = totalRatingXOWSimilarityValue + RatingXOWSimilarityValue;
221             totalOWSimilarityValue = totalOWSimilarityValue + NilaiSimilarity;
222         }
223     }
224     prediksi = totalRatingXOWSimilarityValue / totalOWSimilarityValue;
225     System.out.println("The Prediction : " + prediksi);
226     String NaN = null;
227     if (Double.isNaN(prediksi)) {
228         prediksi = 0;
229         NaN = "yes";
230     }
231     Prediksi p = new Prediksi();
232     p.setTopN(topN);
233     p.setPredictionValue(prediksi);
234     p.setObjek1(String.valueOf(dataTraining.getId_objek_wisata()));
235     p.setUserId(Integer.valueOf(dataTraining.getId_user()));
236     Prediksi.getKoneski().inputPrediksi(p);
237     return prediksi;
238 }

```

Gambar 5.19 Source code method hasilPrediksiItemBased
(Rating,int)

```

240 public void inputPrediksi(Prediksi prediksiFix) throws SQLException {
241     PreparedStatement prep = null;
242     ResultSet res = null;
243     conn.setAutoCommit(false);
244     String sql = "select rating_prediksi from rating_prediksi where id_user = "
245         + prediksiFix.getUserId() + " and TopN = " + prediksiFix.getTopN()
246         + " and id_objek_wisata = " + prediksiFix.getObjek1();
247     prep = conn.prepareStatement(sql);
248     res = prep.executeQuery();
249     if (res.next()) {
250         String sql1 = "UPDATE `rating_prediksi` SET rating_prediksi = "
251             + prediksiFix.getPredictionValue() + " where id_rating_prediksi = "
252             + prediksiFix.getRatingId() + "";
253         prep = conn.prepareStatement(sql1);
254         prep.executeUpdate();
255         conn.commit();
256     } else {
257         String sql1 = "INSERT INTO `rating_prediksi` "
258             + "( `id_objek_wisata`, `id_user`, `rating_prediksi` , `TopN` ) "
259             + "VALUES (" + prediksiFix.getObjek1() + ", " + prediksiFix.getUserId()
260             + ", " + prediksiFix.getPredictionValue() + ", " + prediksiFix.getTopN() + ")";
261         prep = conn.prepareStatement(sql1);
262         prep.executeUpdate();
263         conn.commit();
264     }
265 }

```

Gambar 5.20 Source code method inputPrediksi(Prediksi)

Setelah membuat *method* untuk perhitungan prediksi sampai dengan *method* untuk memasukkan data prediksi ke *database*, penulis membuat *method* untuk *running* proses prediksi pada kelas PrediksiControl.java yaitu *method* prediksiRating(int). *Source code* *method* dapat dilihat pada gambar 5.21 berikut ini.

```

329  public void prediksiRating(int topN) throws SQLException {
330      Prediksi p = new Prediksi();
331      User usr = new User();
332      double prediksi = 0;
333      List<User> u = usr.getAmbilUser();
334      for (int j = 1; j < u.size() + 1; j++) {
335          List<Rating> r = p.getKoneksi().getRatingTraining(j);
336          for (int i = 0; i < r.size(); i++) {
337              System.out.println("ID Test : " + r.get(i).getId_rating() + " | user ID : "
338                  + r.get(i).getId_user() + " | ID OW : " + r.get(i).getId_objek_wisata());
339              prediksi = p.getKoneksi().hasilPrediksiItemBased(r.get(i), topN);
340          }
341      }
342      p.getKoneksi().conn.close();
343  }

```

Gambar 5.21 *Source code method* prediksiRating(int)

Setelah membuat semua *method* untuk prediksi *rating*, penulis membuat *method* main(String[]) untuk *running* program. *Method* main untuk dapat dilihat pada gambar 5.22 berikut ini.

```

345  public static void main(String[] args) throws SQLException {
346      Prediksi p = new Prediksi();
347      p.prediksiRating(4);
348  }

```

Gambar 5.22 *Source code method* main(String[]) untuk *running* *method* prediksiRating(int)

Setelah membuat *method* main, penulis melakukan *running* pada *method* main dengan beberapa maksimum *neighbor* yang telah ditentukan sebelumnya. Berikut ini waktu *running* dengan beberapa skenario maksimum *neighbor*. Hasil *running* program dapat dilihat pada gambar 5.23 sampai dengan gambar 2.25.

```
TopN : 4
ukuran similarity 4
1. 8 x 3 || 2.0 * 0.7 == 1.4
ukuran similarity 4
2. 8 x 9 || 5.0 * 0.42 == 2.1
ukuran similarity 4
3. 8 x 2 || 4.0 * 0.14 == 0.56
ukuran similarity 4
4. 8 x 10 || 4.0 * 0.02 == 0.08
The Prediction : 3.2343750000000001
BUILD SUCCESSFUL (total time: 8 seconds)
```

Gambar 5.23 *Running method prediksiRating(int) dengan maksimum neighbor 4*

```
TopN : 4
ukuran similarity 4
1. 8 x 3 || 2.0 * 0.7 == 1.4
ukuran similarity 4
2. 8 x 9 || 5.0 * 0.42 == 2.1
ukuran similarity 4
3. 8 x 2 || 4.0 * 0.14 == 0.56
ukuran similarity 4
4. 8 x 10 || 4.0 * 0.02 == 0.08
The Prediction : 3.2343750000000001
BUILD SUCCESSFUL (total time: 33 seconds)
```

Gambar 5.24 *Running method prediksiRating(int) dengan maksimum neighbor 6*

```
TopN : 4
ukuran similarity 4
1. 8 x 3 || 2.0 * 0.7 == 1.4
ukuran similarity 4
2. 8 x 9 || 5.0 * 0.42 == 2.1
ukuran similarity 4
3. 8 x 2 || 4.0 * 0.14 == 0.56
ukuran similarity 4
4. 8 x 10 || 4.0 * 0.02 == 0.08
The Prediction : 3.2343750000000001
BUILD SUCCESSFUL (total time: 34 seconds)
```

Gambar 5.25 *Running method prediksiRating(int) dengan maksimum neighbor 8*

Setelah dilakukan *running* terhadap program, diketahui untuk *running method* prediksiRating(int) dengan maksimum *neighbor* 4 adalah 8 *second*, maksimum *neighbor* 6 adalah 33 *second* dan maksimum *neighbor* 8 adalah 34. Untuk mengetahui apakah data prediksi sudah tersimpan di *database*, penulis melakukan pengecekan dengan *query* pada phpmyadmin. Hasil pengecekan dapat dilihat pada gambar 5.26 berikut ini.

Kueri SQL Anda berhasil dieksekusi.

```
SELECT COUNT(*) AS 'TOTAL DATA PREDIKSI' FROM rating_prediksi
```

+ Opsi
TOTAL DATA PREDIKSI
1149

Gambar 5.26 Total data prediksi pada tabel rating_prediksi

Diketahui jumlah data prediksi yang tersimpan di tabel rating_prediksi sebanyak 1149 record data. Data tersebut meliputi prediksi dari rating testing dan prediksi rating objek wisata yang belum dirating user dengan semua skenario maksimum neighbor yang telah ditentukan.

4. Menghitung MAE

Berikut ini merupakan *source code* pada kelas MAE.java untuk implementasi blok proses “menghitung MAE”, dapat dilihat pada gambar 5.27 berikut ini.

```

48 public MAE showPrediction(int topN) throws SQLException {
49     PreparedStatement prep = null;
50     ResultSet res = null;
51     conn.setAutoCommit(false);
52     String sql = "SELECT r.id_testing, r.id_user, r.id_objek_wisata, r.rating, p.rating_prediksi FROM rating_testing r "
53         + "JOIN rating_prediksi p ON (p.id_user = r.id_user AND p.id_objek_wisata = r.id_objek_wisata) WHERE "
54         + "p.TopN = " + topN + " ORDER BY r.id_user ";
55     prep = conn.prepareStatement(sql);
56     res = prep.executeQuery();
57
58     List<Prediksi> predictionList = new ArrayList<Prediksi>();
59     double bantuMAE = 0;
60     double MAE = 0;
61
62     while (res.next()) {
63         Prediksi p = new Prediksi();
64         p.setRatingId(res.getInt(1));
65         p.setUserId(res.getInt(2));
66         p.setObjek2(res.getString(3));
67         p.setRating_training(res.getString(4));
68         p.setPredictionValue(res.getDouble(5));
69         predictionList.add(p);
70         bantuMAE = bantuMAE + Math.abs(Double.valueOf(p.getRating_training()) - p.getPredictionValue());
71     }
72     MAE = bantuMAE / predictionList.size();
73
74     MAE predictionListWithMAE = new MAE();
75     predictionListWithMAE.setListPrediksi(predictionList);
76     predictionListWithMAE.setMAE(MAE);
77     conn.commit();
78     return predictionListWithMAE;
79 }

```

Gambar 5.27 Source code method showPrediction(int) untuk menghitung MAE

Setelah membuat *method* showPrediction(int) penulis membuat *method* main(String[]) untuk me-*running* *method* tersebut. *Method Main* dapat dilihat pada gambar 5.28 berikut ini.

```

83 public static void main(String[] args) throws SQLException {
84     double start = System.nanoTime();
85     MAE mae = new MAE();
86     System.out.println(" MAE = " + mae.getKoneksi().showPrediction(6).getMAE());
87     mae.getKoneksi().conn.close();
88     double finish = System.nanoTime();
89     System.out.println("");
90     System.out.println("Lama Waktu Running : "+(finish-start));
91 }
92 }

```

Gambar 5.28 Method main(String[])

Berikut ini hasil *running* *method* main(String[]) dengan berbagai skenario maksimum *neighbor*.

```

run:
MAE = 0.6334285714285712

Lama Waktu Running : 1654.0 milisecond
BUILD SUCCESSFUL (total time: 1 second)

```

Gambar 5.29 Hasil *running* dengan maksimum *neighbor* 4

```
run:  
MAE = 0.6254285714285716  
  
Lama Waktu Running : 1797.0 milisecond  
BUILD SUCCESSFUL (total time: 2 seconds)
```

Gambar 5.30 Hasil *running* dengan maksimum *neighbor* 6

```
run:  
MAE = 0.629214285714286  
  
Lama Waktu Running : 1629.0 milisecond  
BUILD SUCCESSFUL (total time: 2 seconds)
```

Gambar 5.31 Hasil *running* dengan maksimum *neighbor* 8

5.3 Implementasi *Output*

5.3.1 Aktor *User*

5.3.1.1 Halaman Login



Gambar 5.32 Halaman login *user*

Halaman ini yang pertama ditampilkan untuk mengakses sistem, *user* harus memasukkan *username* dan *password*.

5.3.1.2 Halaman Utama



Gambar 5.33 Halaman utama *user*

Halaman ini adalah halaman yang pertama kali ditampilkan setelah *user* berhasil *login* ke dalam sistem. Pada halaman ini ditampilkan fitur yang dapat diakses oleh *user*.

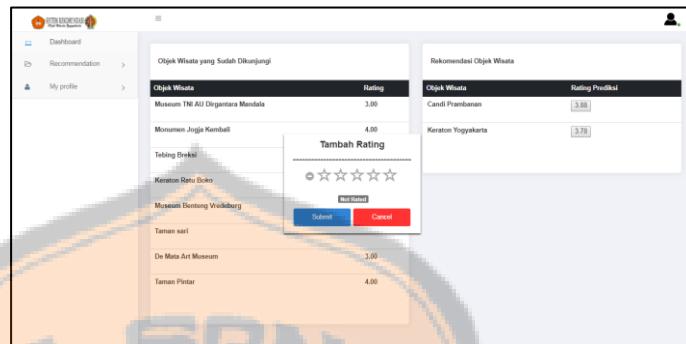
5.3.1.3 Halaman Hasil Prediksi

Objek Wisata yang Sudah Dikunjungi		Rekomendasi Objek Wisata	
Objek Wisata	Rating	Objek Wisata	Rating Prediksi
Museum TNI AU Dirgantara Mandala	3.00	Candi Prambanan	3.88
Monumen Jogja Kembari	4.00	Keraton Yogyakarta	3.78
Teleng Breksi	3.00		
Keraton Ratu Boko	4.00		
Museum Benteng Vredeburg	4.00		
Taman eari	4.00		
De Mata Art Museum	3.00		
Taman Pintar	4.00		

Gambar 5.34 Halaman Hasil Prediksi

Pada halaman ini ditampilkan objek wisata yang telah *rating* oleh *user* dan objek wisata yang belum di-*rating* dan telah diprediksi oleh sistem. *Rating* objek wisata yang telah diprediksi dan belum di-*rating* oleh *user* dapat diubah oleh *user* dengan menekan *rating* prediksi untuk memunculkan *popup*

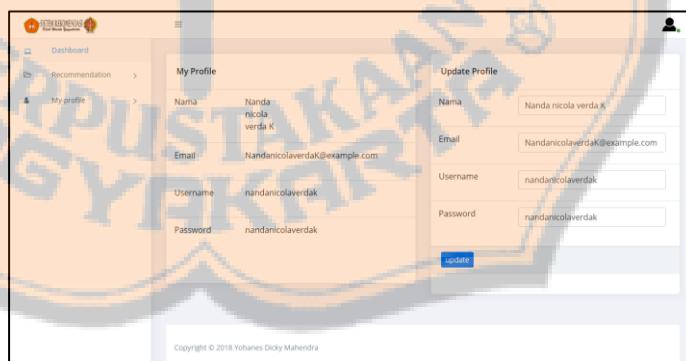
menambah *rating*. *Popup* untuk menambahkan *rating* baru dapat dilihat pada gambar 5.35 berikut ini.



Gambar 5.35 Menu *popup* pada halaman hasil prediksi

Setelah *user* menekan tombol *submit* maka data *rating* baru akan disimpan pada tabel *rating_training* sebagai *rating* sebenarnya dari *user* dan ditampilkan sebagai Objek Wisata yang Sudah Dikunjungi dan *rating* prediksi akan dihapus dari tabel *rating_prediksi*.

5.3.1.4 Halaman *Update Data*



Gambar 5.36 Halaman *update data*

Halaman ini digunakan *user* untuk mengubah datanya dengan menekan tombol *update* dan mengkonfirmasi perubahan data, maka *user* berhasil mengubah datanya.

5.3.2 Aktor Admin

5.5.2.1 Halaman Login



Gambar 5.37 Halaman *login admin*

Halaman ini adalah halaman yang pertama ditampilkan untuk *admin* login dan mengakses sistem.

5.5.2.2 Halaman Utama



Gambar 5.38 Halaman utama admin

Halaman ini adalah halaman yang ditampilkan setelah *admin* berhasil login ke dalam sistem. Pada halaman ini admin dapat menggunakan fitur-fitur dalam sistem.

5.5.2.3 Halaman Tambah Admin



No	M Admin	Name Admin	Email Admin	Username	Password
1	1	Admin	admin@gmail.com	Admin	Admin

Gambar 5.39 Halaman tambah *admin*

Halaman ini merupakan halaman yang digunakan admin untuk menambahkan data *admin* dan juga menghapus data *admin*. *Admin* yang sedang *login* tidak dapat menghapus akun dirinya sendiri. Pada halaman ini *admin* dapat mengakses fitur *edit* data *admin* dengan menekan tombol *edit* pada tabel *admin* dan akan ditampilkan halaman *edit admin*.

5.5.2.4 Halaman Edit Admin



No	M Admin	Name Admin	Email Admin	Username	Password
1	1	Admin	admin@gmail.com	Admin	Admin

Gambar 5.40 Halaman *edit admin*

Halaman ini adalah halaman yang ditampilkan ketika *admin* menekan tombol *edit admin* pada halaman tambah *admin*. Data *admin* yang diklik akan ditampilkan di *form input* untuk di-edit.

5.5.2.5 Halaman Tambah User

No	ID User	Nama User	Email User	Username	Password
1	1	Dominikus Dava	dominikusdava@example.com	dominikusdava	dominikusdava
2	2	Dominika	dominika@example.com	dominika	dominika
3	3	Dominika Andika	dominikaandika@example.com	dominikaandika	dominikaandika
4	4	Dominika Andika P	dominikaandikap@example.com	dominikaandikap	dominikaandikap
5	5	Dominika S	dominikas@example.com	dominikas	dominikas
6	6	Dominika Andika R	dominikaandikar@example.com	dominikaandikar	dominikaandikar
7	7	Dominika Andika R	dominikaandikar@example.com	dominikaandikar	dominikaandikar
8	8	Dominika Andika R	dominikaandikar@example.com	dominikaandikar	dominikaandikar
9	9	Dominika Andika R	dominikaandikar@example.com	dominikaandikar	dominikaandikar

Gambar 5.41 Halaman tambah *user*

Halaman ini merupakan halaman yang digunakan *admin* untuk menambahkan data *user* dan juga menghapus data *user*. Pada halaman ini *admin* dapat mengakses fitur *edit* data *user* dengan menekan tombol *edit* pada tabel *user* dan akan ditampilkan halaman *edit user*.

5.5.2.6 Halaman Edit User

No	ID User	Nama User	Email User	Username	Password
1	1	Dominikus Dava	dominikusdava@example.com	dominikusdava	dominikusdava
2	2	Dominika	dominika@example.com	dominika	dominika
3	3	Dominika Andika	dominikaandika@example.com	dominikaandika	dominikaandika
4	4	Dominika Andika P	dominikaandikap@example.com	dominikaandikap	dominikaandikap
5	5	Dominika S	dominikas@example.com	dominikas	dominikas
6	6	Dominika Andika R	dominikaandikar@example.com	dominikaandikar	dominikaandikar
7	7	Dominika Andika R	dominikaandikar@example.com	dominikaandikar	dominikaandikar
8	8	Dominika Andika R	dominikaandikar@example.com	dominikaandikar	dominikaandikar
9	9	Dominika Andika R	dominikaandikar@example.com	dominikaandikar	dominikaandikar

Gambar 5.42 Halaman *edit user*

Halaman ini adalah halaman yang ditampilkan ketika *admin* menekan tombol *edit user* pada tabel *user* di halaman tambah *user*. Data *user* yang diklik akan ditampilkan di *form input* untuk di-edit.

5.5.2.7 Halaman Tambah Objek Wisata

Gambar 5.43 Halaman tambah objek wisata

Halaman ini digunakan *admin* untuk menambahkan data objek wisata. *Admin* dapat meng-edit objek wisata pada tabel objek wisata dengan menekan tombol *edit* dan dapat menghapus objek wisata dengan menekan tombol *hapus*.

5.5.2.8 Halaman Edit Objek Wisata

Gambar 5.44 Halaman *edit* objek wisata

Halaman ini adalah halaman yang ditampilkan ketika *admin* menekan tombol *edit* pada halaman *input* objek wisata. Data objek wisata yang akan di-*edit* ditampilkan pada *form input*.

5.5.2.9 Halaman Hitung Similarity



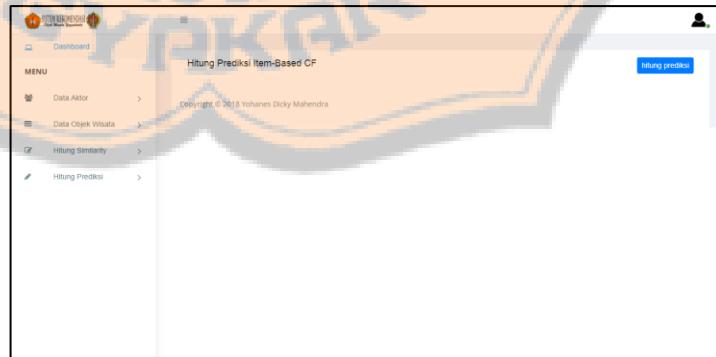
The screenshot shows a table titled 'Tabel Similarity' with 10 entries. The columns are labeled 'No', 'Objek Wisata 1', 'Objek Wisata 2', and 'Similarity'. The data is as follows:

No	Objek Wisata 1	Objek Wisata 2	Similarity
1	Museum TM AU Digringnara Mandala	Museum TM AU Digringnara Mandala	1.0
2	Museum TM AU Digringnara Mandala	Monumen Jago Kembang	0.08
3	Museum TM AU Digringnara Mandala	Tengger Krakal	0.21
4	Museum TM AU Digringnara Mandala	Karawang Ratu Boja	0.21
5	Museum TM AU Digringnara Mandala	Museum Benteng Pendem	0.08
6	Museum TM AU Digringnara Mandala	Taman Zahi	-0.17
7	Museum TM AU Digringnara Mandala	Karang Tengah	0.21
8	Museum TM AU Digringnara Mandala	De Mata Art Museum	0.14
9	Museum TM AU Digringnara Mandala	Taman Pintar	0.26
10	Museum TM AU Digringnara Mandala	Candi Prambanan	-0.01

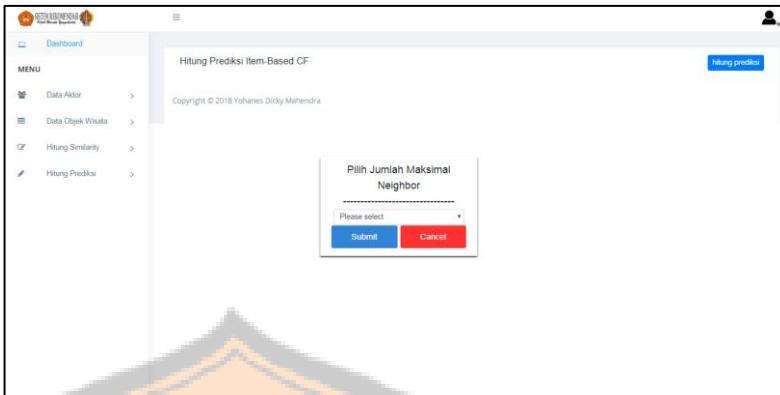
Gambar 4.45 Halaman hitung *similarity*

Pada halaman ini *admin* dapat menghitung *similarity* antar objek wisata dengan menekan tombol “hitung *similarity*”. Tabel *similarity* menampilkan data *similarity* objek wisata yang telah dihitung.

5.5.2.10 Halaman Hitung Prediksi



Gambar 4.46 Halaman hitung prediksi yang masih kosong



Gambar 4.47 Menu popup pada halaman hitung prediksi

A screenshot of the 'Hitung Prediksi Item-Based CF' page. The sidebar shows the 'Hitung Prediksi' item is selected. The main content area displays a table titled 'Tabel Prediksi' with columns: 'No', 'Prediksi', 'Nama Objek Wisata', 'Nama User', and 'Rating Prediksi'. The table contains several rows of data. Above the table, a message states 'MAE = 0.0334285714285712' and 'Maximum Neighbor = 4'. A search bar is located at the top right of the table area. At the bottom of the table, it says 'Showing 1 to 10 of 361 entries'.

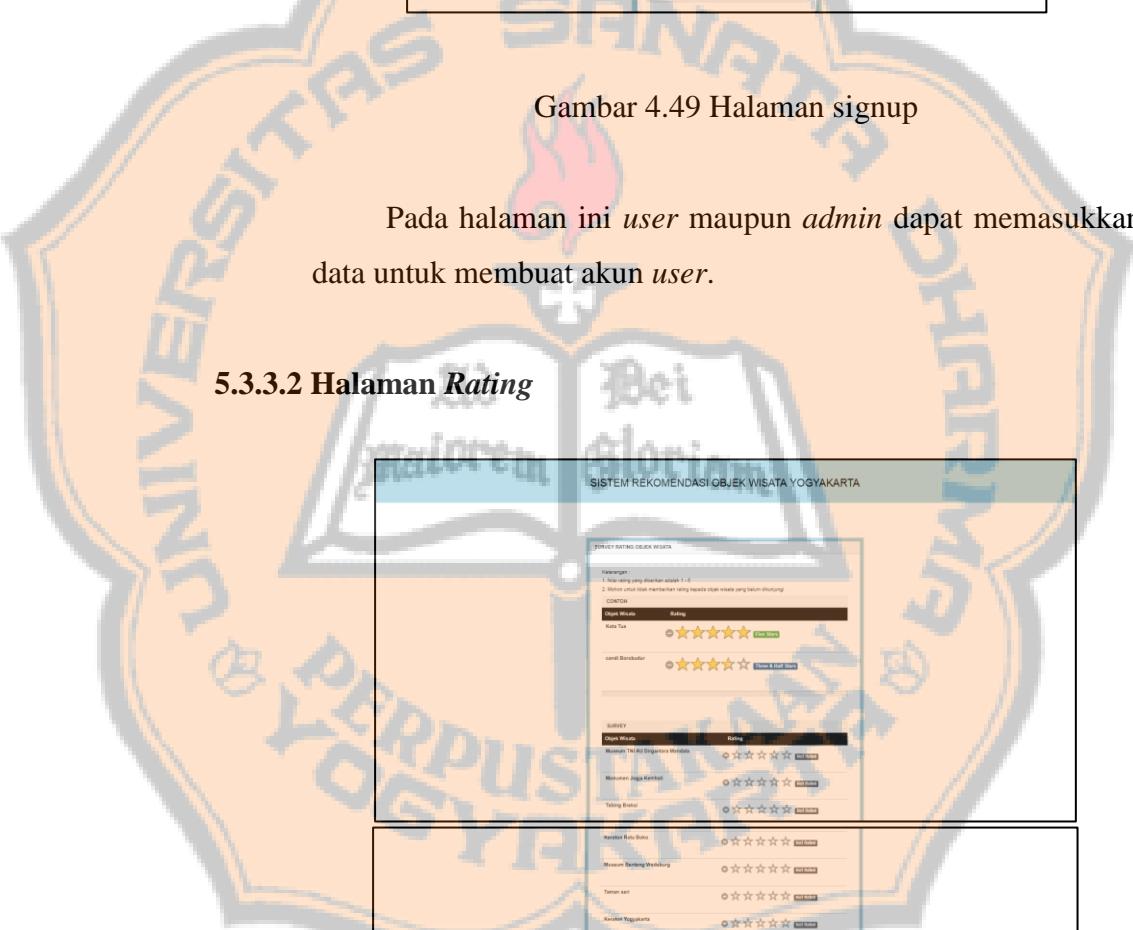
No	Prediksi	Nama Objek Wisata	Nama User	Rating Prediksi
1	364	Keraton Ngayogyakarta	Caryoline Sriyana Clara	3.64
2	365	Candi Prambanan	Caryoline Sriyana Clara	4.0
3	366	Taman Rerat	Deni Intarto	3.53
4	367	Candi Prambanan	Deni Intarto	3.29
5	368	Telaga Warna	Nanda nicola verda K	4.44
6	369	Karang Raja Bopus	Nanda nicola verda K	4.47
7	370	Museum Benteng Vredeburg	Nanda nicola verda K	4.59
8	371	Taman sari	Nanda nicola verda K	4.8
9	372	De Mata Mandala	Nanda nicola verda K	4.02
10	373	Tebing Brumbung	Clara anindia p	1.25

Gambar 4.48 Halaman hitung prediksi dengan tampilan hasil prediksi rating dari data testing dan prediksi rating objek wisata yang belum di-rating user

Pada halaman hitung prediksi, *admin* dapat melakukan perhitungan prediksi *rating* objek wisata dengan menekan tombol ‘‘hitung prediksi’’. Setelah *admin* menekan tombol maka sistem memberikan pilihan pada menu *popup* untuk memilih maksimal *neighbor*.

5.3.3 Aktor Admin dan User

5.3.3.1 Halaman Signup



SISTEM REKOMENDASI OBJEK WISATA YOGYAKARTA

SIGN UP

Nama: budi doremi

Email: budi.doremi@example.com

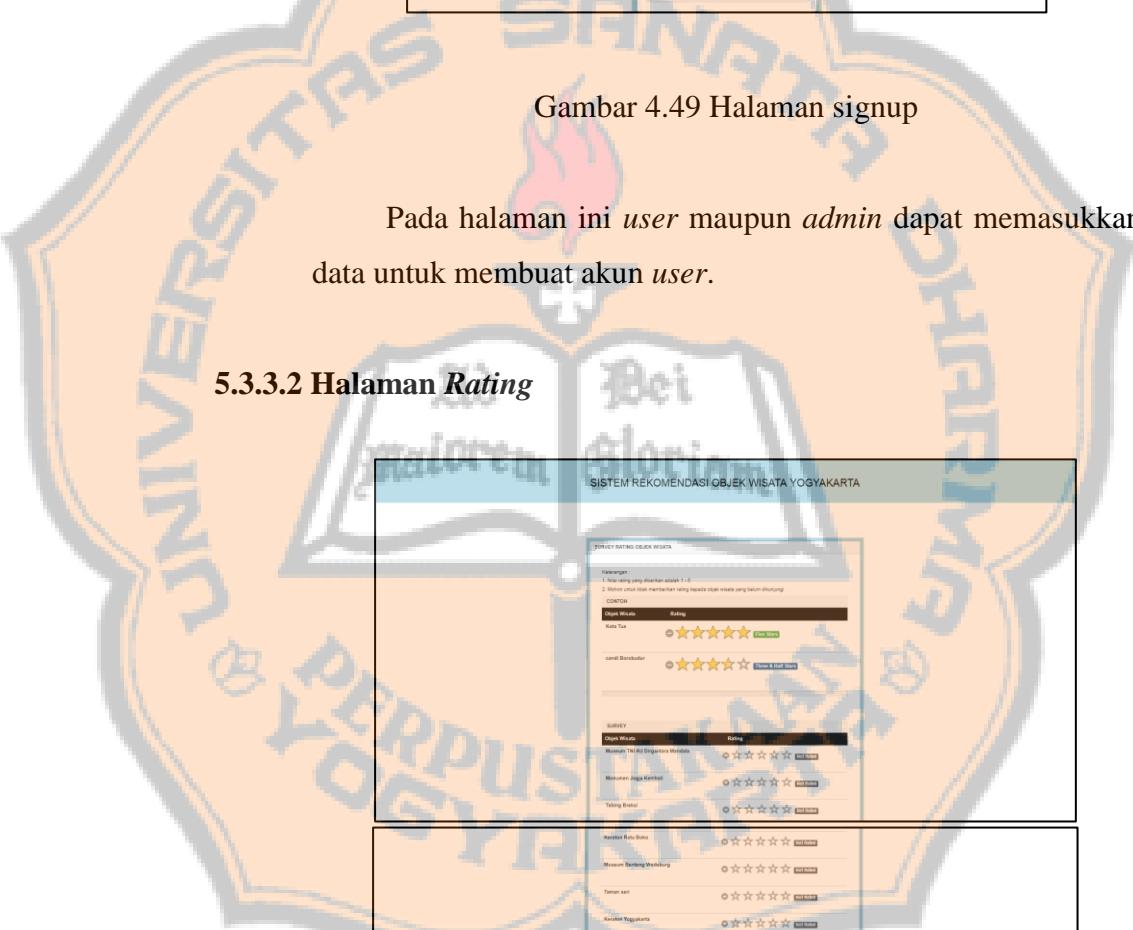
Username: budidoremi

Password:

Gambar 4.49 Halaman signup

Pada halaman ini *user* maupun *admin* dapat memasukkan data untuk membuat akun *user*.

5.3.3.2 Halaman Rating



SISTEM REKOMENDASI OBJEK WISATA YOGYAKARTA

PINJEMAN RATING OBJEK WISATA

Keterangan:
1. Berikan rating yang sebenarnya dengan skala 1-5
2. Banyak orang lain memberikan rating kepada objek wisata yang belum diungkap.

CONTON

Objek Wisata: Kuliner
Rating: ★★★★★ ★★★★★ ★★★★★ ★★★★★ ★★★★★

Rating Dari Bapak: ★★★★★ ★★★★★ ★★★★★ ★★★★★ ★★★★★

SURVEY

Objek Wisata: Kuliner
Rating: ★★★★★ ★★★★★ ★★★★★ ★★★★★ ★★★★★

Mosinan Tki Ali Dogeroro Mosinan: ★★★★★ ★★★★★ ★★★★★ ★★★★★ ★★★★★

Mosinan Juga Kembang: ★★★★★ ★★★★★ ★★★★★ ★★★★★ ★★★★★

Teling Bronté: ★★★★★ ★★★★★ ★★★★★ ★★★★★ ★★★★★

Kelapa Ratu Boko: ★★★★★ ★★★★★ ★★★★★ ★★★★★ ★★★★★

Wisata Borobudur: ★★★★★ ★★★★★ ★★★★★ ★★★★★ ★★★★★

Taman Sari: ★★★★★ ★★★★★ ★★★★★ ★★★★★ ★★★★★

Kawasan Yogyakarta: ★★★★★ ★★★★★ ★★★★★ ★★★★★ ★★★★★

De Mata Art Museum: ★★★★★ ★★★★★ ★★★★★ ★★★★★ ★★★★★

Taman Peler: ★★★★★ ★★★★★ ★★★★★ ★★★★★ ★★★★★

Candi Prambanan: ★★★★★ ★★★★★ ★★★★★ ★★★★★ ★★★★★

Gambar 5.50 Halaman rating

Halaman ini digunakan untuk survey *rating* objek wisata kepada *user* yang baru saja membuat akun (*signup*). Untuk membentuk *profile user* dibutuhkan survey *rating* yang mungkin sudah pernah dikunjungi oleh *user*.

5.4 Analisis Hasil dan Pembahasan

Pada bagian ini penulis melakukan beberapa hal demi tercapainya tujuan penelitian diantaranya adalah menguji validasi program dan menguji keakuratan rekomendasi dengan MAE.

5.4.1 Hasil Uji Validasi Program

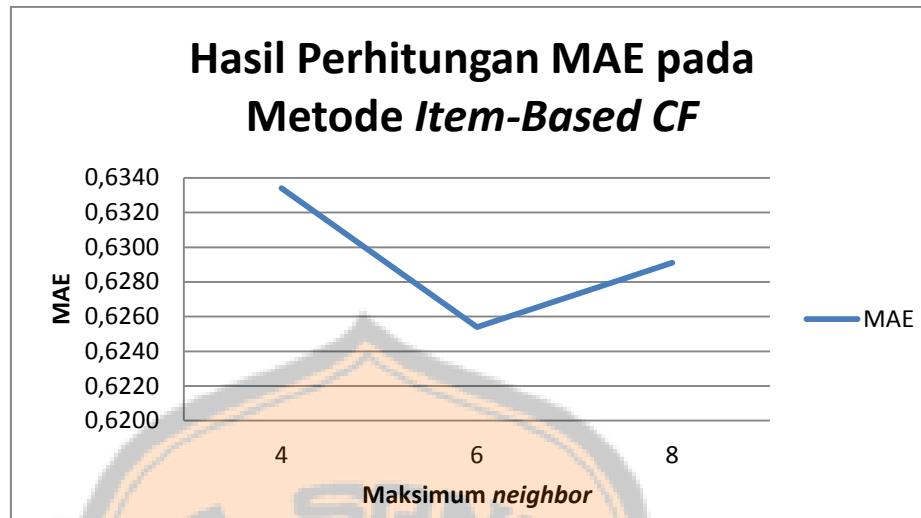
Uji validasi program adalah perbandingan perhitungan manual dengan perhitungan yang dilakukan dengan program. Hasil uji validasi program dijabarkan pada lampiran 2.

5.4.2 Hasil Uji MAE

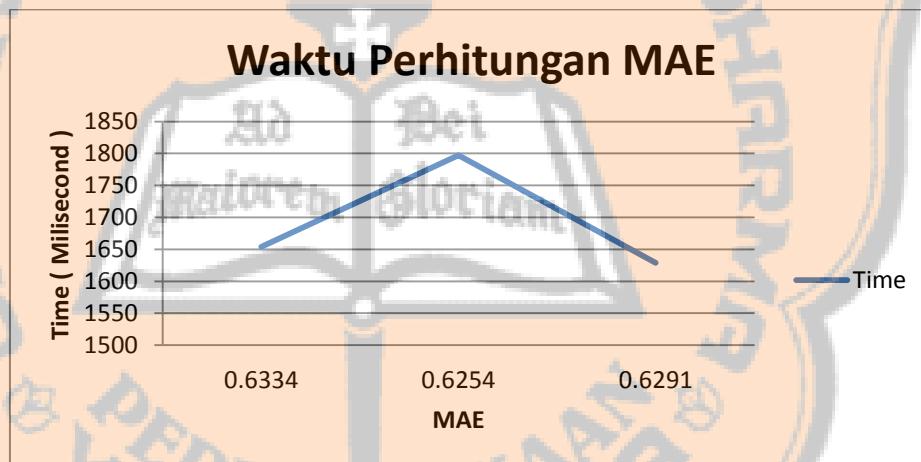
Berikut ini merupakan hasil pengujian MAE dengan skenario penggantian maksimum *neighbor* yang telah ditentukan sebelumnya. Hasil pengujian MAE dapat dilihat pada tabel 5.2 berikut ini.

Tabel 5.2 Hasil perhitungan MAE dengan skenario penggantian maksimum *neighbor*

Jumlah Maksimum <i>Neighbor</i> (Top-N)	MAE	Lama Waktu <i>Running</i>
4	0.6334	1654 milisecond (Gambar 5.29)
6	0.6254	1797 milisecond (Gambar 5.30)
8	0.6291	1629 milisecond (Gambar 5.31)



Gambar 5.51 Grafik garis hasil perhitungan MAE terhadap jumlah maksimum *neighbor*



Gambar 5.52 Grafik waktu *running* MAE

Berdasarkan data grafik pada gambar 5.51, dapat diketahui bahwa metode *Item-Based CF* dapat memprediksi *rating* dengan kesalahan (MAE) dibawah 1. Nilai akumulasi MAE terbesar terdapat pada maksimum *neighbor* 4 dengan *error* sebesar **0.6334** dan nilai terkecil terdapat pada maksimum *neighbor* 6 dengan *error* sebesar **0.6254**.

BAB VI PENUTUP

6.1 Kesimpulan

Berdasarkan penelitian yang telah dilakukan, maka dapat ditarik kesimpulan sebagai berikut:

1. Metode *Item-Based Collaborative Filtering* dapat diterapkan pada sistem rekomendasi objek wisata.
2. Dengan menggunakan *rating* 1 sampai dengan 5, *Item-Based CF* memprediksi *rating* dengan cukup akurat dengan MAE dibawah 1. Selisih MAE dengan tiga maksimum *neighbor* tidak terlalu besar, yaitu hanya 0.004. Dengan maksimum *neighbor* 4 MAE yang dihasilkan serbesar **0.6334** (MAE terbesar), dengan maksimum *neighbor* 6 MAE yang dihasilkan sebesar **0.6254** (MAE terkecil) dan dengan maksimum *neighbor* 8 MAE yang dihasilkan sebesar **0.6291**.

6.2 Saran

Berdasarkan analisis pada penelitian yang sudah dikerjakan, penulis menemukan kekurangan dari metode perhitungan *similarity* dengan *pearson correlation*. Kekurangan itu adalah ketika hasil rumus $\sum_{u \in U_{ij}}(r_{ui} - \bar{r}_i)(r_{uj} - \bar{r}_j)$ sama dengan 0 dan hasil rumus $\sqrt{\sum_{u \in U_{ij}}(r_{ui} - \bar{r}_i)^2 \cdot \sum_{u \in U_{ij}}(r_{uj} - \bar{r}_j)^2}$ sama dengan 0 maka tidak dapat dihitung *similarity*-nya. Hal ini disebabkan metode *pearson correlation* menghitung *similarity* berdasarkan selisih *rating* asli dengan *rating* rata-rata. Jika tidak ada selisih diantara keduanya maka *similarity* tidak dapat dihitung. Diharapkan pada penelitian selanjutnya untuk memperhatikan kekurangan dari metode ini agar hasil prediksi semakin akurat.

DAFTAR PUSTAKA

- Adhitya Pratama, Yudhistira., Wijaya, David., Paulus., & Halim, Arwin. (2013). “*Digital Cakery dengan Algoritma Collaborative Filtering*”. (<https://www.mikroskil.ac.id/ejurnal/index.php/jsm/article/view/94>).
- Adomavicius, Gediminas & Tuzhilin, Alexander. (2005). “*Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions*”. IEEE Transactions on Knowledge and Data Engineering, vol. 17, no. 6.
- A Djamal, Ramadhanuz., Maharani, Warih., & Kurniati, Angelina Prima. (2010). “*Analisis dan Implementasi Metode Item-Based Clustering Hybrid Pada Recommender System*”. Konferensi Nasional Sistem dan Informatika.
- Asanov, D. (2015). “*Algorithm and method in Recommender Systems*”. International Journal of Computer Applications, vol. 18, 2015.
- Dwicahya, Imam. (2018). “*Perbandingan Sistem Rekomendasi Film Metode User-Based dan Item-Based Collaborative Filtering*”.
- Fathoni., Putra, Pacu., & Abdi Sucipta, Rio. (2016). “*Penerapan Metode Item Based Collaborative Filtering pada Sistem Electronic Commerce Berbasis Website*”. Annual Research Seminar, vol. 2, no. 1.
- Ricci, Francesco., Rokach, Lior., & Shapira, Bracha. (2011) “*Recommender System Handbook*”. Springer Science+Business Media, New York, USA.
- Theodorus, Arvid., budiyanto Setyohadi, Djoko., & Ernawati. (2016). “*User-Based Collaborative Filtering Dengan Memanfaatkan Pearson-Correlation Untuk Mencari Neighbors Terdekat Dalam Sistem Rekomendasi*”. (<http://ejournal.uajy.ac.id/8924/>).

LAMPIRAN 1 : Narasi Use Case**4.1.3 Narasi Use Case****4.1.3.1 Login**

Use Case Name :	Login	
Use Case ID :	SR-01	
Priority :	High	
Primary Business Actor :	Admin dan User	
Description :	<p><i>Use case ini mendeskripsikan admin dan user yang akan login untuk menggunakan sistem.</i></p>	
Precondition :	<p>Admin berada pada halaman “login admin” dan user berada pada halaman “login”.</p>	
	Actor Action	System Response
	Step 1 : Masukkan username dan password	
		Step 2 : Memberikan konfirmasi login
	Step 3 : Menekan tombol “oke” pada menu konfirmasi	
Typical Course of Event :	<p>Step 3 : Melakukan pengecekan username dan password ke database.</p>	
	<p>Step 4 : Mengalihkan halaman ke halaman “home” untuk admin dan halaman “user dashboard” untuk</p>	

		user
Alternative :	Jika username atau password salah maka akan ditampilkan pesan kesalahan. Admin dan user akan dialihkan kembali ke halaman login.	
Postcondition :	Admin dialihkan ke halaman “home” dan user dialihkan ke halaman “user dashboard”.	

4.1.3.2 Tambah Objek Wisata

Use Case Name :	Tambah Objek Wisata	
Use Case ID :	SR-02	
Priority :	<i>High</i>	
Primary Business Actor :	Admin	
Description :	Use case ini mendeskripsikan admin yang akan menggunakan fitur menambah data objek wisata.	
Precondition :	Admin berada pada halaman “Tambah Objek Wisata”.	
Typical Course of Event :	Actor Action	System Response
	Step 1 : Memasukkan semua data pada form input data.	
	Step 2 : menekan tombol “submit”	
		Step 3 : Menampilkan konfirmasi input data objek wisata
	Step 4 : Menekan tombol “oke” pada menu konfirmasi	

		Step 5 : Menginputkan data objek wisata ke database.
		Step 6 : Menampilkan data yang telah ditambahkan ke “tabel objek wisata”
		Step 7 : Menampilkan pemberitahuan telah berhasil menambah data objek wisata
Alternative :	Jika data gagal ditambahkan ke database maka akan ada pemberitahuan input data ke database gagal.	
Postcondition :	Admin dapat melihat data yang telah ditambahkan pada “tabel objek wisata”	

4.1.3.3 Edit Objek Wisata

Use Case Name :	Edit Objek Wisata	
Use Case ID :	SR-03	
Priority :	Medium	
Primary Business Actor :	Admin	
Description :	Use case ini mendeskripsikan admin yang akan menggunakan fitur mengedit data objek wisata.	
Precondition :	Admin berada pada halaman “Tambah Objek Wisata”.	
Typical Course	Actor Action	System Response

of Event :	Step 1 : Menekan tombol “edit” pada “tabel objek wisata”	
	Step 2 : Menampilkan halaman Edit “Objek Wisata”	
	Step 3 : Menampilkan data objek wisata yang akan diedit di form input	
	Step 4 : Mengedit data objek wisata di form input	
	Step 5 : Menekan tombol “simpan”	
	Step 6 : Menampilkan konfirmasi simpan perubahan data	
	Step 7 : Menekan tombol “oke” pada menu konfirmasi	
	Step 8 : Mengupdate data objek wisata di database	
	Step 9 : Menampilkan data objek wisata pada “tabel objek wisata”	
Alternative :	Jika data gagal diupdate di database maka akan	

	ada pemberitahuan update data objek wisata gagal.
Postcondition :	Admin dapat melihat data yang telah diupdate pada “tabel objek wisata”

4.1.3.4 Hapus Objek Wisata

Use Case Name :	Hapus Objek Wisata	
Use Case ID :	SR-04	
Priority :	<i>Medium</i>	
Primary Business Actor :	Admin	
Description :	Use case ini mendeskripsikan admin yang akan menggunakan fitur menghapus data objek wisata.	
Precondition :	Admin berada pada halaman “Tambah Objek Wisata”.	
	Actor Action	System Response
Typical Course of Event :	Step 1 : Menekan tombol “hapus” pada “tabel objek wisata”	
	Step 2 : Menampilkan konfirmasi menghapus objek wisata	
	Step 3 : Menekan tombol “oke” pada menu konfirmasi	
	Step 4 : Menghapus data objek wisata yang dipilih di	

		database
		Step 5 : Menampilkan data objek wisata pada “tabel objek wisata”
Alternative :	Jika data gagal dihapus di database maka akan ada pemberitahuan hapus data objek wisata gagal.	
Postcondition :	Admin dapat mengecek objek wisata sudah terhapus pada “tabel objek wisata”	

4.1.3.5 Tambah User

Use Case Name :	Tambah User							
Use Case ID :	SR-05							
Priority :	<i>High</i>							
Primary Business Actor :	Admin dan User							
Description :	Use case ini mendeskripsikan admin yang akan menggunakan fitur menambah user dan user yang akan membuat akun.							
Precondition :	Admin berada pada halaman “Tambah User” dan user berada pada halaman “signup”							
Typical Course of Event :	<table border="1"> <thead> <tr> <th>Actor Action</th> <th>System Response</th> </tr> </thead> <tbody> <tr> <td>Step 1 : Admin dan User memasukkan semua data pada form input data.</td> <td></td> </tr> <tr> <td>Step 2 : Admin menekan tombol “submit” dan user menekan tombol “sign”</td> <td></td> </tr> </tbody> </table>	Actor Action	System Response	Step 1 : Admin dan User memasukkan semua data pada form input data.		Step 2 : Admin menekan tombol “submit” dan user menekan tombol “sign”		
Actor Action	System Response							
Step 1 : Admin dan User memasukkan semua data pada form input data.								
Step 2 : Admin menekan tombol “submit” dan user menekan tombol “sign”								

	up”	
	Step 3 : Menampilkan konfirmasi input data user	
	Step 4 : Menekan tombol “oke” pada menu konfirmasi	
	Step 4 : Menginputkan data user ke database.	
	Step 5 : Mengalihkan admin dan user ke halaman “Rating”	
	Step 6 : Menampilkan pemberitahuan telah berhasil menambah data user.	
Alternative :	Jika data gagal ditambahkan ke database maka akan ada pemberitahuan input data ke database gagal. User akan dikembalikan ke halaman “signup” dan admin dikembalikan ke halaman “Tambah User”	
Postcondition :	Admin dan user berada pada halaman memberi rating objek wisata.	

4.1.3.6 Edit User

Use Case Name :	Edit User	
Use Case ID :	SR-06	

Priority :	<i>Medium</i>	
Primary Business Actor :	Admin dan User	
Description :	Use case ini mendeskripsikan admin yang akan menggunakan fitur mengedit data user pada halaman “edit user” dan user yang akan mengedit datanya pada halaman “user profile”.	
Precondition :	Admin berada pada halaman “User”.	
	Actor Action	System Response
	Step 1 : Menekan tombol “edit” pada “tabel user”	
		Step 2 : Menampilkan halaman edit user
		Step 3 : Menampilkan data user yang akan diedit di form input
Typical Course of Event :	Step 4 : Mengedit data user di form input	
	Step 5 : Menekan tombol “edit”	
		Step 6 : Menampilkan konfirmasi simpan perubahan data
	Step 7 : Menekan tombol “oke” pada menu konfirmasi	
		Step 8 : Mengupdate data user di database

		Step 9 : Menampilkan data user pada “tabel user”
Alternative :	Jika data gagal diupdate di database maka akan ada pemberitahuan update data user gagal.	
Postcondition :	Admin dapat melihat data yang telah diupdate pada “tabel user” di halaman “Edit User”	

4.1.3.7 Hapus User

Use Case Name :	Hapus User	
Use Case ID :	SR-07	
Priority :	Medium	
Primary Business Actor :	Admin	
Description :	Use case ini mendeskripsikan admin yang akan menggunakan fitur menghapus data user.	
Precondition :	Admin berada pada halaman “User”.	
Typical Course of Event :	Actor Action	System Response
	Step 1 : Menekan tombol “hapus” pada “tabel user”	
		Step 2 : Menampilkan konfirmasi menghapus user
	Step 3 : Menekan tombol “oke” pada menu konfirmasi	
		Step 4 : Menghapus data user yang dipilih

		di database
		Step 5 : Menampilkan data user pada “tabel user”
Alternative :	Jika data gagal dihapus di database maka akan ada pemberitahuan hapus data user gagal.	
Postcondition :	Admin dapat mengecek user sudah terhapus pada “tabel user”	

4.1.3.8 Tambah Admin

Use Case Name :	Tambah Admin	
Use Case ID :	SR-08	
Priority :	Medium	
Primary Business Actor :	Admin	
Description :	Use case ini mendeskripsikan admin yang akan menggunakan fitur menambah admin.	
Precondition :	Admin berada pada halaman “Admin”.	
	Actor Action	System Response
	Step 1 : Admin memasukkan semua data pada form input data.	
Typical Course of Event :	Step 2 : Admin menekan tombol “submit”	
		Step 3 : Menampilkan konfirmasi input data admin

	Step 4 : Menekan tombol “oke” pada menu konfirmasi	
	Step 5 : Menginputkan data admin ke database.	
	Step 6 : Menampilkan pemberitahuan telah berhasil menambah data admin.	
Alternative :	Jika data gagal ditambahkan ke database maka akan ada pemberitahuan input data ke database gagal, admin dikembalikan ke halaman “Admin”	
Postcondition :	Admin dapat melihat data admin yang ditambahkan pada tabel “admin”	

4.1.3.9 Edit Admin

Use Case Name :	Edit Admin	
Use Case ID :	SR-09	
Priority :	Medium	
Primary Business Actor :	Admin	
Description :	Use case ini mendeskripsikan admin yang akan menggunakan fitur mengedit data admin.	
Precondition :	Admin berada pada halaman “Tambah	

	Admin”.	
	Actor Action	System Response
Typical Course of Event :	Step 1 : Menekan tombol “edit” pada “tabel admin”	
		Step 2 : Menampilkan halaman edit admin
		Step 3 : Menampilkan data admin yang akan diedit di form input
	Step 4 : Mengedit data admin di form input	
	Step 5 : Menekan tombol “submit”	
		Step 6 : Menampilkan konfirmasi simpan perubahan data
	Step 7 : Menekan tombol “oke” pada menu konfirmasi	
		Step 8 : Mengupdate data admin di database
		Step 9 : Menampilkan data admin pada “tabel admin”
Alternative :	Jika data gagal diupdate di database maka akan ada pemberitahuan update data admin gagal.	

Postcondition :	Admin dapat melihat data yang telah diupdate pada “tabel admin” di halaman edit admin
------------------------	---

4.1.3.10 Hapus Admin

Use Case Name :	Hapus Admin	
Use Case ID :	SR-10	
Priority :	<i>Medium</i>	
Primary Business Actor :	Admin	
Description :	Use case ini mendeskripsikan admin yang akan menggunakan fitur menghapus data admin.	
Precondition :	Admin berada pada halaman “Tambah Admin”.	
	Actor Action	System Response
Typical Course of Event :	Step 1 : Menekan tombol “hapus” pada “tabel admin”	
		Step 2 : Menampilkan konfirmasi menghapus admin
	Step 3 : Menekan tombol “oke” pada menu konfirmasi	
		Step 4 : Menghapus data admin yang dipilih di database
		Step 5 : Menampilkan data admin pada “tabel admin”

Alternative :	Jika data gagal dihapus di database maka akan ada pemberitahuan hapus data admin gagal. Admin yang sedang login tidak bisa menghapus data dirinya di “tabel admin”
Postcondition :	Admin dapat mengecek admin sudah terhapus pada “tabel admin”

4.1.3.11 Memberi rating objek wisata

Use Case Name :	Memberi Rating Objek Wisata	
Use Case ID :	SR-11	
Priority :	<i>High</i>	
Primary Business Actor :	Admin dan User	
Description :	Use case ini mendeskripsikan admin dan user yang akan menggunakan fitur rating setelah menambahkan data user.	
Precondition :	Admin dan user berada pada halaman “Rating”.	
	Actor Action	System Response
Step 1 : Memilih rating objek wisata		
Step 2 : Menekan tombol “simpan”		
Typical Course of Event :	Step 3 : Menampilkan konfirmasi menambahkan rating	
Step 4 : Menekan tombol “oke” pada menu konfirmasi		

		Step 5 : Menambahkan rating objek wisata ke database
		Step 6 : Mengalihkan admin ke halaman “Tambah User” dan mengalihkan user ke halaman “user dashboard”
		Step 7 : Menampilkan pemberitahuan sukses menambahkan rating
Alternative :	-	
Postcondition :	Admin dialihkan ke halaman “Tambah User” dan user dialihkan ke halaman “user dashboard”	

4.1.3.12 Menghitung Similaritas Objek Wisata

Use Case Name :	Menghitung Similaritas Objek Wisata	
Use Case ID :	SR-12	
Priority :	<i>High</i>	
Primary Business Actor :	Admin	
Description :	Use case ini mendeskripsikan admin yang akan menggunakan fitur hitung similarity.	
Precondition :	Admin berada pada halaman “Hitung Similarity”.	

	Actor Action	System Response
	Step 1 : Menekan tombol “hitung similarity”	
Typical Course of Event :		Step 2 : Menampilkan konfirmasi menghitung similarity
	Step 3 : Menekan tombol “oke” pada menu konfirmasi	
		Step 4 : Menghitung similarity seluruh objek wisata di database
		Step 5 : Menampilkan similarity objek wisata pada “tabel similarity”
Alternative :	Jika gagal menghitung similarity, akan ditampilkan pemberitahuan gagal menghitung similarity pada halaman “Hitung Similarity”	
Postcondition :	Admin dapat melihat similarity antar objek wisata pada “tabel similarity”	

4.1.3.13 Melihat Similaritas Objek Wisata

Use Case Name :	Melihat Similaritas Objek Wisata	
Use Case ID :	SR-13	
Priority :	<i>Medium</i>	
Primary Business Actor :	Admin	
Description :	Use case ini mendeskripsikan admin yang akan melihat similaritas objek wisata.	
Precondition :	Admin berada pada halaman “Home”.	
Typical Course of Event :	Actor Action	System Response
	Step 1 : Menekan menu “Hitung Similarity”	Step 2 : Menampilkan data similarity objek wisata pada “tabel similarity”
Alternative :	Jika similarity belum dihitung maka admin dapat menghitung similarity dengan menekan tombol “hitung similarity”	
Postcondition :	Admin dapat melihat similarity antar objek wisata pada “tabel similarity”	

4.1.3.14 Menghitung Prediksi

Use Case Name :	Menghitung Prediksi	
Use Case ID :	SR-14	
Priority :	<i>High</i>	
Primary Business	Admin	

Actor :		
Description :	Use case ini mendeskripsikan admin yang akan menggunakan fitur hitung prediksi.	
Precondition :	Admin berada pada halaman “Hitung Prediksi”.	
	Actor Action	System Response
	Step 1 : Menekan tombol “hitung prediksi”	
	Step 2 : Menampilkan menu popup untuk memilih jumlah maksimal neighbor	
	Step 3 : Memilih maksimal neighbor di menu select pada menu popup	
	Step 4 : Menekan tombol “submit” pada menu popup	
	Step 5 : Menampilkan konfirmasi hitung prediksi	
	Step 6 : Menekan tombol “oke” pada menu konfirmasi	
	Step 7 : Menghitung prediksi rating seluruh objek wisata yang kosong pada	

		tabel rating_training di database
		Step 8 : Menampilkan prediksi rating objek wisata pada “tabel prediksi”
		Step 9 : Menampilkan pemberitahuan sukses menghitung prediksi
Alternative :	Jika gagal menghitung prediksi, akan ditampilkan pemberitahuan gagal menghitung prediksi pada halaman “Hitung Prediksi”	
Postcondition :	Admin dapat melihat prediksi rating objek wisata pada “tabel prediksi”	

4.1.3.15 Melihat Hasil Prediksi Objek Wisata

Use Case Name :	Melihat Hasil Prediksi Objek Wisata	
Use Case ID :	SR-14	
Priority :	Medium	
Primary Business Actor :	Admin dan User	
Description :	Use case ini mendeskripsikan admin dan user yang akan melihat hasil prediksi rating objek wisata.	
Precondition :	Admin berada pada halaman “Home” dan user berada pada halaman “user dashboard”.	
Typical Course of	Actor Action	System Response

Event :	Step 1 : Admin menekan menu “hitung prediksi” dan user menekan menu “recommendation”. Untuk melihat data rating prediksi, pada aktor admin dapat menekan tombol “hitung prediksi” dan mengulangi proses pada usecase “Hitung Prediksi”.	
	Step 2 : Pada aktor user, sistem mengalihkan user ke halaman “recommendation”	
	Step 3 : Pada aktor user, sistem menampilkan data rating objek wisata yang telah dirating dan data rating prediksi objek wisata	
Alternative :	-	
Postcondition :	Admin dapat melihat prediksi rating objek wisata pada “tabel prediksi” dan user dapat melihat rating prediksi pada tabel “Rekomendasi Objek Wisata”	

LAMPIRAN 2 : Uji Validasi Program

5.6.2 Hasil Uji Validasi Program

Uji validasi digunakan penulis untuk membandingkan perhitungan manual dengan perhitungan pada program. Berikut ini beberapa perhitungan untuk uji validasi.

5.6.2.1 Uji Validasi Perhitungan *Similarity*

Langkah pertama yaitu melakukan perhitungan *similarity item-based collaborative filtering* dengan rumus 2.1. Penulis mengambil contoh perhitungan *similarity* antara objek wisata dengan id 1 dan objek wisata dengan id 8. Untuk mengetahui nama objek wisata dengan id 1 dan objek wisata dengan id 8 dapat dilihat pada gambar 5.56 dan gambar 5.57 berikut ini.

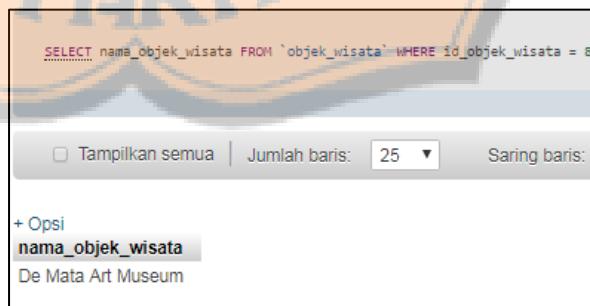


```
SELECT nama_objek_wisata from `objek_wisata` where id_objek_wisata = 1
```

Tampilkan semua | Jumlah baris: 25 Saring baris:

+ Opsi
nama_objek_wisata
Museum TNI AU Dirgantara Mandala

Gambar 5.56 Nama objek wisata dengan id 1



```
SELECT nama_objek_wisata FROM `objek_wisata` WHERE id_objek_wisata = 8
```

Tampilkan semua | Jumlah baris: 25 Saring baris:

+ Opsi
nama_objek_wisata
De Mata Art Museum

Gambar 5.57 Nama objek wisata dengan id 8

- Diketahui bahwa objek wisata dengan id 1 adalah Museum TNI AU Dirgantara Mandala dan objek wisata dengan id 8 adalah De Mata Art Museum. Dengan menggunakan perhitungan *similarity pearson correlation*, langkah pertama yang dilakukan adalah mengeksekusi *query* untuk menghitung rata-rata *rating* dari kedua objek wisata tersebut. Berikut ini adalah *query* tersebut yang dapat dilihat pada *query 5.1*.

```
SELECT avg(rating) FROM `rating` WHERE id_objek_wisata = 1 and data_type = 0
SELECT avg(rating) FROM `rating` WHERE id_objek_wisata = 8 and data_type = 0
```

Query 5.1 Mengambil nilai rata-rata objek wisata

Dari hasil eksekusi *query*, didapatkan rata-rata *rating* objek wisata 1 adalah **3.546296** dan rata-rata *rating* objek wisata 8 adalah **3.634615**. Hasil *running query* dapat dilihat pada gambar 5.58 dan gambar 5.59 berikut ini.

<pre>SELECT avg(rating) FROM `rating` WHERE id_objek_wisata = 1 and data_type = 0</pre>
<input type="checkbox"/> Tampilkan semua Jumlah baris: 25 ▾ Saring baris: Cari di tabel ini + Opsi avg(rating) 3.546296

Gambar 5.58 Hasil running query objek wisata 1

<pre>SELECT avg(rating) FROM `rating` WHERE id_objek_wisata = 8 and data_type = 0</pre>
<input type="checkbox"/> Tampilkan semua Jumlah baris: 25 ▾ Saring baris: Cari di tabel ini + Opsi avg(rating) 3.634615

Gambar 5.59 Hasil running query objek wisata 8

2. Langkah selanjutnya adalah mencari interseksi *rating* dari objek wisata 1 dengan objek wisata 8 dengan menggunakan *query*, seperti pada *query* 5.2 berikut ini.

```
SELECT id_user FROM `rating` WHERE id_objek_wisata = 1 AND data_type = 0 AND id_user IN(SELECT id_user FROM `rating` WHERE id_objek_wisata = 8 AND data_type = 0) ORDER BY id_user
```

Query 5.2 Mengambil interseksi *rating* objek wisata

Dari hasil eksekusi *query*, diketahui interseksi *rating* objek wisata 1 dengan objek wisata 8 adalah sebanyak 19 *user* yang me-*rating* kedua objek wisata yang sama. Hasil eksekusi *query* dapat dilihat pada gambar 5.55 dan ditampilkan pada tabel 5.3 berikut ini.

Opsi	id_user
Tambahkan semua	1
Jumlah baris:	25
Starting baris:	Cari di tabel ini
Urut berdasarkan kunci:	Tidak ada
+ Opsi	
	1
	2
	8
	9
	18
	23
	43
	45
	57
	59
	61
	65
	66
	73
	77
	84
	93
	94
	97

Gambar 5.60 Hasil eksekusi *query* mencari interseksi *rating*

Tabel 5.3 Hasil eksekusi *query* mencari interseksi *rating*

No	Id user	Objek wisata 1	Objek Wisata 8
1	1	3	3
2	2	4	4
3	8	4	3
4	9	3	4
5	18	3	1
6	23	3	2
7	43	4	5
8	45	3	3
9	57	3	5
10	59	3	4
11	61	2	4.5
12	65	1	2
13	66	3.5	3.5
14	73	2	3
15	77	4	1
16	84	5	5
17	93	3	4
18	94	5	4
19	97	4	4

Pada tabel 5.2 diketahui data *user* yang me-*rating* objek wisata 1 dan juga objek wisata 8. *Rating* inilah yang akan digunakan untuk menghitung *similarity* dengan rumus 2.1 berikut.

Nilai *Similarity* PC(1,8) =

$$\frac{(3 - 3.546296)(3 - 3.634615) + (4 - 3.546296)(4 - 3.634615) + (4 - 3.546296)(3 - 3.634615) +}{\sqrt{(3 - 3.546296)^2 + (4 - 3.546296)^2 + (4 - 3.546296)^2 + (3 - 3.546296)^2 + (3 - 3.546296)^2 + (3 - 3.546296)^2 +}}$$

$$\frac{(3 - 3.546296)(4 - 3.634615) + (3 - 3.546296)(1 - 3.634615) + (3 - 3.546296)(2 - 3.634615) +}{\sqrt{(3 - 3.546296)^2 + (4 - 3.546296)^2 + (3 - 3.546296)^2 + (3 - 3.546296)^2 + (3 - 3.546296)^2 + (2 - 3.546296)^2 +}}$$

$$\frac{(4 - 3.546296)(5 - 3.634615) + (3 - 3.546296)(3 - 3.634615) + (3 - 3.546296)(5 - 3.634615) +}{\sqrt{(1 - 3.546296)^2 + (3.5 - 3.546296)^2 + (2 - 3.546296)^2 + (4 - 3.546296)^2 + (5 - 3.546296)^2 + (3 - 3.546296)^2 +}}$$

$$\frac{(3 - 3.546296)(4 - 3.634615) + (2 - 3.546296)(4.5 - 3.634615) + (1 - 3.546296)(2 - 3.634615) + (3.5 - 3.546296) +}{\sqrt{(5 - 3.546296)^2 + (4 - 3.546296)^2} \cdot \sqrt{(3 - 3.634615)^2 + (4 - 3.634615)^2 + (3 - 3.634615)^2 + (4 - 3.634615)^2 +}}$$

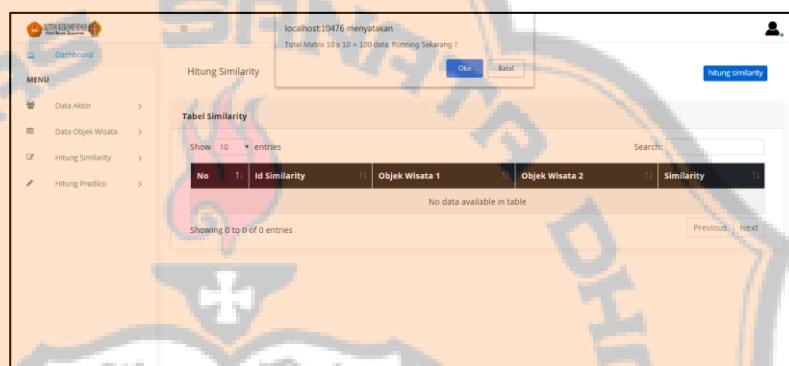
$$\frac{(2 - 3.546296)(3 - 3.634615) + (4 - 3.546296)(1 - 3.634615) + (5 - 3.546296)(5 - 3.634615) + (3 - 3.546296)}{\sqrt{(1 - 3.634615)^2 + (2 - 3.634615)^2 + (5 - 3.634615)^2 + (3 - 3.634615)^2 + (5 - 3.634615)^2 + (4 - 3.634615)^2 +}}$$

$$\frac{(4 - 3.634615) + (5 - 3.546296)(4 - 3.634615) +}{\sqrt{(4.5 - 3.634615)^2 + (2 - 3.634615)^2 + (3.5 - 3.634615)^2 + (3 - 3.634615)^2 + (1 - 3.634615)^2 + (5 - 3.634615)^2 +}}$$

$$\frac{(4 - 3.546296)(4 - 3.634615)}{\sqrt{(4 - 3.634615)^2 + (4 - 3.634615)^2 + (4 - 3.634615)^2}}$$

Nilai similarity PC = $\frac{7.476315}{23.01034} = 0.324911$ dibulatkan menjadi **0.32**

Perhitungan manual yang telah dilakukan memperoleh hasil *similarity* objek wisata 1 dan objek wisata 8 sebesar 0.32. Penulis kemudian mencocokan hasil perhitungan manual dengan perhitungan pada sistem website. Skenario pencocokan yaitu dengan mencari nilai *similarity* pada tabel *similarity* di menu hitung *similarity* pada sistem website, seperti yang disajikan pada gambar 5.61 dan gambar 5.62 berikut ini.



Gambar 5.61 Menghitung *similarity* pada menu hitung *similarity*

No	Id Similarity	Objek Wisata 1	Objek Wisata 2	Similarity
1	1	Museum TNI AU Dirgantara Mandala	Museum TNI AU Dirgantara Mandala	1.0
2	2	Museum TNI AU Dirgantara Mandala	Monumen Jogja Kembali	0.22
3	3	Museum TNI AU Dirgantara Mandala	Tebing Breksi	0.13
4	4	Museum TNI AU Dirgantara Mandala	Keraton Ratu Boko	0.11
5	5	Museum TNI AU Dirgantara Mandala	Museum Bronteng Yogyakarta	0.16
6	6	Museum TNI AU Dirgantara Mandala	Taman sari	0.12
7	7	Museum TNI AU Dirgantara Mandala	Keraton Yogyakarta	0.27
8	8	Museum TNI AU Dirgantara Mandala	De Mata Art Museum	0.32
9	9	Museum TNI AU Dirgantara Mandala	Taman Pintar	0.23
10	10	Museum TNI AU Dirgantara Mandala	Candi Prambanan	-0.07

Gambar 5.62 Hasil perhitungan *similarity* pada tabel *similarity*

Dapat dilihat pada gambar 5.62, kotak merah adalah data objek wisata dan nilai *similarity* objek wisata tersebut. Nilai *similarity* dengan perhitungan pada sistem website menghasilkan nilai 0.32, nilai tersebut sama dengan perhitungan manual. Hal ini membuktikan bahwa perhitungan manual dan perhitungan pada sistem sudah tervalidasi dengan benar.

5.6.2.2 Uji Validasi Perhitungan Prediksi

Pada skenario perhitungan prediksi, penulis mengambil sampel data *testing* dengan id *user* 5 dan id objek wisata 4, seperti pada gambar 5.63.

		id_testing	id_user	id_objek_wisata	rating
<input type="checkbox"/>	Ubah	1	1	10	5.00
<input type="checkbox"/>	Ubah	2	1	7	4.00
<input type="checkbox"/>	Ubah	3	2	9	4.00
<input type="checkbox"/>	Ubah	4	2	10	3.00
<input type="checkbox"/>	Ubah	5	5	10	4.00
<input checked="" type="checkbox"/>	Ubah	6	5	4	3.50
<input type="checkbox"/>	Ubah	7	6	6	4.00
<input type="checkbox"/>	Ubah	8	6	8	3.00
<input type="checkbox"/>	Ubah	9	7	4	4.00
<input type="checkbox"/>	Ubah	10	7	8	3.00

Gambar 5.63 Sampel data *testing* untuk prediksi

Diketahui pada gambar 5.63, *user* 5 telah me-*rating* objek wisata 4 dengan *rating* sebesar 3.5. Untuk mengetahui nama *user* 5 dan objek wisata 4 maka dilakukan eksekusi *query* pada database seperti pada gambar 5.64 berikut ini.

```
SELECT dw.nama_objek_wisata AS 'OBJEK WISATA 4', us.nama_user AS 'USER 5' FROM objek_wisata dw, user us WHERE id_objek_wisata = 4 AND id_user = 5
```

Tampilkan semua | Jumlah baris: 25 | Saring baris: Cari di tabel ini | Edit di kolom

+ Opsi
OBJEK WISATA 4 | USER 5
Keraton Ratu Boko | angelia sekartati p

Gambar 5.64 Hasil eksekusi *query* untuk mengetahui nama *user* 5 dan objek wisata 5

Diketahui pada gambar 5.64 bahwa nama *user* 5 adalah angelia sekarati p dan nama objek wisata 4 adalah keraton ratu boko. Dengan menggunakan data testing yang telah dipilih, maka dilakukan perhitungan dengan proses sebagai berikut:

1. Jumlah *neighbor* yang dipilih untuk proses perhitungan prediksi adalah sebanyak 6 maksimum *neighbor*. 6 *neighbor* tersebut adalah 6 objek wisata selain objek wisata 4 yang telah di-rating oleh *user* 5, serta nilai *similarity neighbor* diatas 0.
2. Melakukan eksekusi *query* pada basis data untuk mendapatkan nilai *similarity* objek wisata lain yang memiliki nilai paling tinggi dan diatas 0. Berikut adalah *query* tersebut pada *query* 5.3.

```
SELECT ows.id_objek_wisata_1 AS 'Objek Wisata 1', ows.id_objek_wisata_2 AS 'Objek Wisata 2',
r.rating AS 'Rating Objek Wisata 2', ows.nilai_similarity AS 'Nilai Similarity' FROM similarity ows
JOIN rating r ON (ows.id_objek_wisata_2=r.id_objek_wisata) WHERE id_objek_wisata_1=4 AND
nilai_similarity> 0 AND id_objek_wisata_2 IN (SELECT id_objek_wisata FROM rating WHERE
id_user = 5 AND data_type = 0) AND id_user = 5 ORDER BY nilai_similarity DESC
```

Query 5.3 *Query* untuk mendapat nilai *similarity* objek wisata 4 dengan objek wisata lainnya

Hasil eksekusi *query* 5.3 dapat dilihat pada gambar 5.65 sebagai berikut.

Objek Wisata 1	Objek Wisata 2	Rating Objek Wisata 2	Nilai Similarity
4	3	3.50	0.48
4	9	5.00	0.39
4	5	4.00	0.32
4	7	4.00	0.23
4	2	3.50	0.21
4	6	4.00	0.18

Gambar 5.65 Rating dan nilai *similarity* 6 *neighbor*

Dari gambar 5.65 diketahui 6 *neighbor* yang memiliki *similarity* tertinggi terhadap objek wisata 4 dan telah di-*rating* oleh user 5.

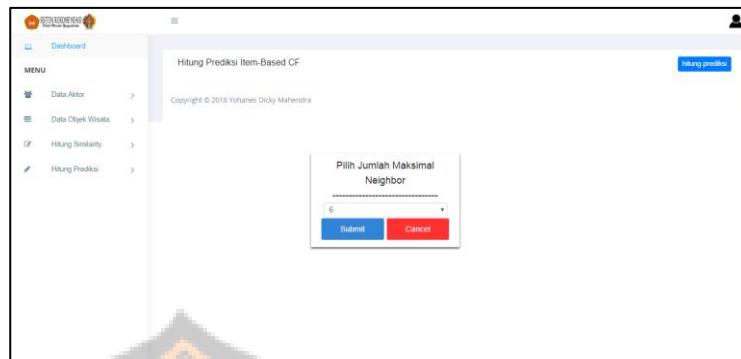
3. Langkah selanjutnya adalah melakukan perhitungan prediksi dengan menggunakan rumus 3.2.

Prediksi *rating* ID *testing* 6 =

$$\frac{(3.5 * 0.48) + (5 * 0.39) + (4 * 0.32) + (4 * 0.23) +}{|0.48| + |0.39| + |0.32| + |0.23| +}$$
$$\frac{(3.50 * 0.21) + (4 * 0.18)}{|0.21| + |0.18|}$$
$$= \frac{7.285}{1.810} = 4.024861878 \text{ dibulatkan menjadi } 4.02$$

Setelah dilakukan perhitungan, didapatkan nilai prediksi *user* 5 terhadap objek wisata 4 dengan nilai prediksi sebesar 4.02.

4. Langkah selanjutnya yang penulis lakukan adalah membandingkan perhitungan manual dengan perhitungan pada sistem, seperti pada gambar 5.66 dan gambar 5.67 berikut ini.



Gambar 5.66 Menghitung prediksi dengan maksimum *neighbor* 6 pada sistem website

No	id Prediksi	Nama Objek Wisata	Nama User	Rating Prediksi
15	395	Keraton Ratu Boko	angela sekarati p	4.02
16	399	De Mata Art Museum	angela sekarati p	3.93
17	400	Candi Prambanan	angela sekarati p	3.97

Gambar 5.67 Prediksi *rating* untuk objek wisata 4 dan user 5

Pada gambar 5.67, diketahui bahwa jumlah maksimal *neighbor* adalah 6. Dengan menggunakan data *testing* dengan id = 6 menunjukkan nilai prediksi yang sama dengan perhitungan manual, yaitu 4.02. Hal ini membuktikan bahwa perhitungan prediksi manual dengan perhitungan prediksi di sistem web sudah tervalidasi dengan benar.

5.6.2.3 Perhitungan MAE

Pada uji perhitungan prediksi sebelumnya, didapatkan hasil prediksi untuk data *testing id* = 6 adalah 4.02 dan *rating* aslinya adalah 3.50.

Nilai MAE sesuai dengan rumus 3.1 dari sampel tersebut adalah:

$$\frac{|4.02 - 3.50|}{1} = 0.52$$

