PENGEMBANGAN CHAT BOT PADA COMA UNTUK MEMBERIKAN MOTIVASI KEPADA PENGGUNA MENGGUNAKAN AIML

SKRIPSI

Untuk memenuhi sebagian persyaratan
Memperoleh gelar Sarjana Komputer

Disusun oleh:
Bestralaga Rusmarasy
NIM: 145150200111134

PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2019

PENGESAHAN

PENGEMBANGAN CHAT BOT PADA COMA UNTUK MEMBERIKAN MOTIVASI KEPADA PENGGUNA MENGGUNAKAN AIML

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Bestralaga Rusmarasy
NIM: 145150200111134

Skripsi ini telah diuji dan dinyatakan lulus pada
13 Maret 2019
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
Bayu Priyambadha, S.Kom, M.Kom	Fajar Pradana, S.ST, M.Eng.
NIP: 19820909 200812 1 004	NIP: 19871121 201504 1 004

Mengetahui
Ketua Jurusan Teknik Informatika

Tri Astoto Kurniawan, S.T, M.T, Ph.D
NIP: 19710518 200312 1 001

PERNYATAAN ORISINALITAS

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, didalam naskah skripsi ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di perguruan tinggi dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis disitasi dalam naskah ini dan disebutkan dalam daftar pustaka.

Apabila ternyata didalam naskah skripsi ini dapat dibuktikan terdapat unsur plagiasi, saya bersedia skripsi ini digugurkan dan gelar akademik yang telah saya peroleh (sarjana) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan pasal 70).

Malang, 13 Maret 2019

Bestralaga Rusmarasy
NIM: 145150200111134

KATA PENGANTAR

Segala puji bagi Allah SWT atas limpahan segala rizki-Nya sehingga penulis dapat menyelesaikan skripsi berjudul "Pengembangan Chat Bot pada CoMa untuk memberikan motivasi kepada pengguna menggunakan AIML".

Untuk kesempatan ini penulis juga menyampaikan rasa terima kasih kepada pihak-pihak yang telah membantu penulis selama penyusunan skripsi, diantaranya:

1. Allah SWT yang telah memberi kemudahan dan jalan dalam semua proses penulisan skripsi ini.
2. Orang Tua penulis, yaitu Bapak Herlan Rumpoko dan Ibu Irna Sekarmastuti yang selalu mengingatkan dan memotivasi penulis.
3. Bapak Bayu Priyambadha, S.Kom, M.Kom selaku dosen pembimbing I yang telah meluangkan waktu untuk memberi masukan, motivasi, ilmu serta saran yang sangat bermanfaat bagi proses penyelesaian skripsi ini.
4. Bapak Fajar Pradana, S.ST, M.Eng selaku dosen pembimbing II yang telah meluangkan waktu untuk memberi masukan, motivasi, ilmu serta saran yang sangat bermanfaat bagi proses penyelesaian skripsi ini.
5. Bapak Wayan Firdaus Mahmudy, S.Si, M.T, Ph.D., Bapak Ir. Heru Nurwasito, M.Kom., Bapak Marji, Drs., M.T, dan Bapak Edy Santoso, S.Si, M.Kom. selaku Dekan, Wakil Dekan 1, Wakil Dekan 2, dan Wakil Dekan 3 Fakultas Ilmu Komputer, Universitas Brawijaya.
6. Bapak Tri Astoto Kurniawan, S.T, M.T, Ph.D selaku Ketua Jurusan Teknik Informatika.
7. Seluruh Dosen Fakultas Ilmu Komputer, Universitas Brawijaya atas ketersediaannya dalam mengajarkan dan membagikan ilmu yang bermanfaat bagi penulis.
8. Teman-teman Teknik Informatika 2014 yang membantu memberikan ilmu dalam implementasi dari penelitian ini.
9. Dhanuari Indra Bastari selaku pengembang CodeManiac pertama yang telah memberikan pengarahan terhadap sistem yang telah dibuat.

10. Afif Nandya Saputra, Fathony Teguh Irawan, dan Miftahul Rizki Purwonegoro selaku teman penulis yang turut berperan dalam pengembangan lanjut sistem CodeManiac.
11. Semua pihak yang tidak bisa dituliskan disini yang terlibat secara langsung maupun tidak langsung dalam proses pengerjaan skripsi maupun sebagai pemberi semangat dan motivasi.

Penulis menyadari masih banyak kekurangan dalam penyusunan skripsi ini baik dalam teknik penyajian materi, pembahasan maupun penarikan kesimpulan. Demi kesempurnaan penelitian skripsi ini, saran dan kritik yang sifatnya membangun sangat penulis harapkan. Semoga karya tulis ini bermanfaat dan dapat memberikan pengetahuan yang berarti bagi pihak yang membutuhkan.

Malang, 13 Maret 2019

Penulis

bestralagarumpoko@gmail.com

ABSTRAK

Bestralaga Rusmarasy, Pengembangan Chat Bot pada CoMa untuk memberikan motivasi kepada pengguna menggunakan AIML

Pembimbing: Bayu Priyambadha, S.Kom, M.Kom dan Fajar Pradana, S.ST, M.Eng.

Kemudahan akses dan kegiatan belajar mengajar pada *e-Learning* memunculkan masalah baru, dimana terdapat beberapa mahasiswa yang merasa kesulitan mengerjakan tugas di *e-Learning* dan kurangnya motivasi dalam mengerjakan tugasnya tersebut. Dengan adanya website *e-Learning* Code Maniac, mahasiswa merasa terbantu dalam proses pembelajaran, namun belum mampu membangun motivasi belajar mahasiswa. Maka pada penelitian ini akan mengembangkan lebih lanjut website *e-Learning* CodeManiac dengan penambahan *Chatbot* untuk memberikan motivasi terhadap pengguna serta aktif menangkap perilaku pengguna dan menampilkan pesan berdasarkan perilaku pengguna. pada penelitian ini peneliti menggunakan AIML. AIML sendiri adalah salah satu turunan dari XML. Sama seperti XML, AIML berisikan elemen yang memuat tag-tag yang mendefinisikan sebuah data objek. AIML disini sebagai bahasa yang digunakan untuk menyusun logika *chatbot* yang berisikan kumpulan pola dan respon yang dapat digunakan oleh *chatbot* untuk penelusuran jawaban untuk setiap kalimat yang diberikan oleh pengguna. Dengan implementasi ini telah berhasil memotivasi pengguna *e-Learning* CodeManiac dengan adanya *chatbot* yang dapat berinteraksi dengan pengguna dan aktif menangkap perilaku pengguna dan pada tahap akhir pengembangan lanjut pada Code Maniac ini telah diuji dengan pengujian *black-box*, *whitebox* dan usabilitas dimana nilai masing-masing adalah 100% valid, 100% valid, dan 88,8%.

.**Kata kunci**: *e-Learning, Chatbot, AIML, CodeManiac*

ABSTRACT

Bestralaga Rusmarasy, Development of Chat Bots on CoMa to provide motivation to user using AIML

Mentor: Bayu Priyambadha, S.Kom, M.Kom and Fajar Pradana, S.ST, M.Eng.

Ease of access and teaching-learning activities in e-Learning raises a new problem, where there are some students who found a difficulties to do a task in e-Learning and lack of motivation in doing the task. With the Code Maniac e-Learning website was help students in the learning process, but haven't been able built the students motivation. So in this study will enhance CodeManiac e-Learning website with the addition of *chatbot* to provide motivation to users and actively capture user behavior and give a messages based on user behavior. In this study writer used AIML. AIML itself is subset of XML. Like XML, AIML here as the language used to construct *chatbot* logic which contains a set of patterns and responses that can be used by *chatbot* to track answers every sentence given. With this implementation has been motivated CodeManiac users feels by the *chatbot* who can interact with users and actively captures user behavior. In final step on the enhancement of codemaniac has been tested with whitebox testing, blackbox testing and usability testing with each score 100% valid, 100% valid and 88%.

Keywords: *e-Learning, Chatbot, AIML, CodeManiac*

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR LAMPIRAN

BAB 1 PENDAHULUAN

1.1 Latar belakang

Saat ini mahasiswa dan dosen menuntut suatu teknologi informasi yang dapat diakses dimana saja memunculkan adanya *e-Learning*. Dimana *e-Learning* dapat mempermudah pekerjaan para pengajar dalam memberikan tugas serta materi, membantu mahasiswa dalam pembelajaran dengan mudahnya akses serta tidak perlu bertatap muka dengan pengajar. Banyak mahasiswa yang kesulitan dalam mengerjakan tugas di *e-Learning* dan kurangnya motivasi dalam mengerjakan tugasnya tersebut menandakan bahwa kurang efektifnya penggunaan elearning tersebut. Penelitian yang dilakukan oleh Alinier Guillaume mengenai faktor berulang yang berkontribusi terhadap efektivitas *e-Learning* adalah 'latihan'. Pelajar diberi kesempatan atau diperlukan untuk mempraktekkan materi pendidikan yang disajikan melalui solusi *e-Learning* dalam studi kasus, simulasi atau situasi kerja yang sebenarnya. Tujuannya adalah untuk mendukung retensi pembelajaran dan transfer untuk berlatih, karena praktik simulasi menyediakan lingkungan belajar yang aman sebelum keterampilan diterapkan dalam situasi kerja yang penting: 'Simulasi *Intermediate-fidelity* adalah teknik pelatihan yang bermanfaat. Ini memungkinkan kelompok kecil siswa untuk berlatih dalam lingkungan yang aman dan terkendali bagaimana bereaksi secara memadai dalam situasi perawatan pasien yang kritis. Jenis pelatihan ini sangat berharga untuk membekali siswa dengan keterampilan teknis dan non-teknis minimum sebelum mereka menggunakannya dalam pengaturan praktik '(Alinier et al, 2006).

Dengan adanya website *e-Learning* Code Maniac mahasiswa merasa terbantu dalam proses pembelajaran yang interaktif (Bastari, 2017), namun e-Learning belum mampu membangun motivasi belajar mahasiswa dimana motivasi menempati urutan ketiga dalam faktor efektif tidaknya sebuah eLearning yang interaktif berdasarkan studi dari Gamage (2014) dan upaya membangun motivasi tersebut dengan memberikan suatu fitur seperti *Chatbot. Chatbot* merupakan sebuah perangkat lunak yang dibangun untuk mengGambarkan suatu proses percakapan atau komunikasi timbal-balik kepada pengguna dengan bentuk teks. Tanggapan dihasilkan berdasarkan filtrasi kata yang telah diinput oleh pengguna dan menghasilkan respon balasan berdasarkan pengetahuan *bot* tersebut, sehingga percakapan atau komunikasi yang terjadi seolah dilakukan oleh dua manusia yang saling berinteraksi (Wallace, 2003).

Pengembangan *Chatbot* juga memerlukan suatu basis sistem pengetahuan yang mempermudah interaksi antara komputer dengan manusia. Penelitian yang dilakukan oleh Kristanto mengenai *Artificial Intelligence* (AI) atau disebut juga dengan kecerdasan buatan merupakan salah satu cabang dari ilmu pengetahuan komputer yang dikhususkan pada pengolahan sebuah tingkah laku pada sistem kecerdasan komputer. Berdasarkan studi oleh Kristanto (2003) fundamental dari kecerdasan buatan adalah Pengetahuan dari komputer itu sendiri yang didapat

dari proses seperti pelatihan. Penelitian oleh Wallace mengenai *Artificial Intelligence Markup Language* (AIML). Dimana *AIML* merupakan *script markup* yang digunakan untuk menyusun pola atau pattern pengenalan bahasa manusia dalam perangkat lunak agar dapat berinteraksi dengan manusia menggunakan bahasa alami (*Natural Language*). Konsep *AIML* ini berupa *template matching* yaitu dengan mencocokan inputan pengguna dengan *pattern* (pola-pola) atau contoh percakapan yang telah ditentukan sebelumnya (Wallace, 2003).

Berdasarkan permasalahan dan solusi yang telah dipaparkan sebelumnya, yakni mengenai *e-Learning* dan pemaparaan mengenai *chatbot*, maka dilaksanakan sebuah penelitian mengenai pemanfaatan *Chatbot pada Code Maniac* sebagai pemberi motivasi terhadap pengguna. Aplikasi ini tetap menggunakan pendekatan OO (pendekatan berorientasi objek) seperti pengembangan sebelumnya dimana sebagai cara pemodelan untuk mendeskripsikan sistem perangkat lunak. Pengembanganan *Chatbot* memanfaatkan AIML sebagai penentu jawaban ataupun balasan yang tepat berdasarkan pertanyaan/inputan dari pengguna. Teknologi lain yang digunakan dalam pengembangan aplikasi ini adalah PHP, dimana PHP merupakan bahasa pemrograman untuk pengembangan aplikasi berbasis website.

1.2 Rumusan masalah

Berdasarkan latar belakang tersebut maka dapat dirumuskan:

1. Bagaimana hasil analisis *Chat Bot* terhadap proses pembelajaran coding java pada CoMa?
2. Bagaimana rancangan *Chat Bot* pada CoMa?
3. Bagaimana implementasi *Chat Bot* pada CoMa?
4. Bagaimana pengujian *Chat Bot* pada CoMa?

1.3 Tujuan

Adapun tujuan dari dilakukannya penelitian ini yaitu:

1. Meningkatkan motivasi mahasiswa pada pembelajaran coding java pada CoMa.
2. Membantu mahasiswa dalam mengerjakan tugas.
3. Mengetahui penerapan *Chat Bot* pada aplikasi CoMa.

1.4 Manfaat

Adapun manfaat dari penelitian ini adalah:

1. Membantu mahasiswa dalam pengerjaan tugas yang telah diberikan di *e-Learning.*
2. Memotivasi mahasiswa pada saat berlangsungnya proses pembelajaran coding java pada CoMa.

1.5 Batasan masalah

1. Memotivasi mahasiswa pada saat berlangsungnya proses pembelajaran.
2. Tampilan *Chatbot* muncul ketika diakses halaman *live coding*.
3. *Chatbot* melayani percakapan dalam bahasa indonesia.
4. Implementasi *Chatbot* pada CoMa berbasis web.
5. *Chatbot* tidak dapat melakukan penambahan pengetahuan secara otomatis, dibutuhkan seorang admin untuk menambahkan pengetahuan pada *Chatbot*.
6. Dokumen pengetahuan *Chatbot* disimpan dalam dalam bentuk format AIML.

1.6 Sistematika pembahasan

Bagian ini berisi struktur skripsi ini mulai Bab Pendahuluan sampai Bab Penutup.

BAB 1 PENDAHULUAN

Bab ini menjelaskan secara singkat mengenai latar belakang, rumusan masalah, manfaat, batasan masalah, sistematika pembahasan serta jadwal penelitian.

BAB 2 LANDASAN KEPUSTAKAAN

Pada bab ini berisi tentang dasar teori yang sesuai dengan pengembangan sistem.

BAB 3 METODOLOGI PELAKSANAAN

Pada bab ini menjelaskan mengenai metode penulis dalam melaksanakan penelitian yang dimulai dari studi literatur/kepustakaan.

BAB 4 ANALISIS PERANCANGAN SISTEM

Pada bab ini menjelaskan analisis yang dilakukan terhadap cara kerja aplikasi dan perancangan *Chatbot* dalam tahapan yang sistematis.

BAB 5 IMPLEMENTASI

Pada bab ini berisi implementasi dari analisis dan perancangan sistem *Chatbot* yang me yang telah dirancang sebelumnya.

BAB 6 PENGUJIAN

Pada bab ini menjelaskan apakah sistem telah dibangun sesuai rancangan dan memvalidasi serta memverifikasi sistem yang telah dibangun.

BAB 7 PENUTUP

Pada bab ini penulis menyimpulkan hasil penelitian dan saran untuk peneliti selanjutnya.

BAB 2 LANDASAN KEPUSTAKAAN

2.1 Kajian Pustaka

Melalui kajian pustaka, peneliti akan mengkaji beberapa hasil penelitian lain yang relevan dengan pengembangan lanjut CodeManiac. Pembahasan ini bertujuan guna mendukung penelitian yang akan berlangsung. Beberapa penelitian tersebut merupakan yang berkaitan dengan *e-Learning* dan *Chatbot*.

Penelitian mengenai *e-Learning* sebelumnya dilakukan oleh Eli B. Cohen dan Malgorzata Nycz yang berjudul "Learning Objects and *E-Learning*: an Informing Science Perspective". Penelitian ini meneliti bagaimana Gambaran umum mengenai *e-Learning* dari dasar-dasarnya melalui bagaimana *e-Learning* diimplementasikan menggunakan Sistem Manajemen Konten Pembelajaran yang luas diluar kampus. Mereka menjelaskan bahwa *e-Learning* terbagi menjadi empat bagian terpisah dan dimensi yang saling berkaitan, pertama teori pembelajaran, kedua dimensi psikologis dan tuntutan para pembelajar, ketiga teknologi, dan keempat konten yang dipelajari. Dimana sistem *e-Learning* menggantikan peran guru sebagai pengajar, namun sumber utama pengetahuan bukanlah guru melainkan *knowledge-based* yang dikumpulkan, dirakit dan diurutkan oleh guru, serta masukan dari pelajar dan guru tersebut saat berlangsungnya belajar-mengajar. *Learning Content Management System* (LCMS) lebih baik daripada *Learning Object Repository* yang sederhana dimana yang berisi komponen yang mendukung penulisan, dikombinasikan dengan repositori objek pembelajaran dan alat untuk mengirimkan objek kepada siswa (Cohen, 2006).

Penelitian mengenai *e-Learning* selanjutnya dilakukan oleh Kusuma Ayu Laksitowening, Amarilis Putri Yanuarifiani dan Yanuar Firdaus Arie Wibowo yang berjudul "*Enhancing E-Learning System to Support Learning Style Based Personalization*". Penelitian ini meneliti tentang bagaimana cara meningkatkan sistem *e-Learning* untuk mendukung pembelajaran berdasarkan personalisasi. Penelitian ini dilakukan berdasarkan bahwa setiap siswa unik dan memiliki kebutuhan tersendiri terkait dengan metodologi pembelajaran. Penelitian ini mengusulkan peningkatan pada sistem *e-Learning* yang mengakomodasi personalisasi berdasarkan *Felder-Silverman Learning Style Model.* Penelitian ini memanfaatkan ontologi dalam pemodelan objek dan relasi yang diperlukan dalam sistem *e-Learning* yang diusulkan. Pendekatan Felder-Silverman mengidentifikasi gaya belajar siswa berdasarkan empat dimensi: persepsi, masukan, pemrosesan, dan pemahaman. Setiap dimensi mengkategorikan siswa menjadi dua. Dengan demikian, ada 16 jenis gaya belajar berdasarkan kombinasi dua kategori dalam empat dimensi. Presepsi dibedakan menjadi presepsi indrawi dan intuitif. Masukan dibedakan menjadi visual dan verbal. Pemrosesan dibagi menjadi aktif dan reflektif. Pemahaman dibagi menjadi sekuensial dan global (Laksitowening, 2016). "

Penelitian selanjutnya dilakukan oleh Richard Wallace tentang "A.L.I.C.E. (Artificial Linguistic Internet Computer Entity)" pada tahun 1995. Penelitian ini meneliti tentang metode bagaimana *chatterbot* dapat menggunakan bahasa natural (*Natural Language Processing*) dimana percakapan manusia terhadap komputer dengan inputan berupa bahasa alami manusia. Dimana ALICE menggunakan metode AIML untuk membentuk respons terhadap pertanyaan. Basis pengetahuan utama ALICE disimpan pada file AIML yang berbeda. AIML adalah bahasa *scripting interpreter* yang merupakan subset dari XML (*Extensible Markup Language)* namum dengan fungsi membuat sistem inputan-balasan berbasis pengetahuan. Dokumen AIML terdiri dari objek-objek yang dipisahkan oleh tag-tag tertentu seperti layaknya dokumen XML atau HTML (Wallace, 2003).

2.2 Landasan Teori

2.2.1 Chatbot

Chatbot merupakan sebuah sistem berbasiskan *Natural Language*(bahasa alami). *Chatbot* menurut studi dari Utdirartatmo (2001) merupakan salah satu pengembangan sistem percakapan antara manusia dengan mesin/komputer. Percakapan yang terjadi terbatas atas *knowledge base chatbot* itu sendiri. *Grammar* (aturan berbahasa yang benar) dan struktur bahasa tersebut menjadi sebuah batasan untuk menjadi sebuah kata kunci dalam filtrasi untuk merespon inputan oleh pengguna. Berdasarkan studi dari Wallace (2010) *Chatbot* adalah karakter bahasa alami yang berkomunikasi dengan penggunanya, atau orang-orang yang sedang chatting di internet bahkan melalui komunikasi suara seperti telepon. Setiap *chatbot* diatur oleh seorang botmaster, yaitu orang dibelakang layar yang berandil besar dalam membentuk suatu kepribadian *bot* serta pengetahuan *chatbot*.

2.2.2 AIML

AIML merupakan subset dari *Extensible Markup Language* (XML) namun dengan fungsi yang mendetail. Salah satu fungsinya adalah membuat sistem input pertanyaan-balasan berbasis pengetahuan, pada AIML tag dapat diartikan berbeda sesuai dengan fungsi masing-masing (Wallace, 2003).

Bagian-bagian penting dari AIML adalah sebagai berikut

a. *Category*

Category adalah inti dari pengetahuan, disetiap *category* wajib terdapat *pattern* dan *template*. Dibawah ini adalah Gambaran penggunaan *category* (Wallace, 2003):

```
<category>
    <pattern>siapa orang tuamu </pattern>
    <template>kamu tidak perlu tahu orang tuaku</template>
</category>
```

Pada saat *category* tersebut dipanggil maka bot akan membalas inputan user "siapa orang tuamu" dengan "kamu tidak perlu tahu orang tuaku".

b. *Pattern*

Pattern merupakan sebuah pertanyaan/inputan, pattern dapat direspon dengan satu atau lebih Balasan. Suatu pattern yang dianggap sama disebut *wildcard*, dibawah ini adalah cara penggunaan *wildcard*:

```
Siapa nama *
```

sepadan dengan inputan "siapa nama kamu", "siapa nama dosen kamu", dan sebagainya (Wallace, 2003).

c. *Template*

Template merupakan jawaban dari satu *pattern* atau lebih. Dibawah adalah contoh penggunaan *Template*:

```
Selamat tinggal.
```

Variabel atau sebuah inputan yang berarti sama seperti nama bot dan disisipkan ke dalam kalimat. *Template* juga memungkinkan untuk meneruskan ke *pattern* lain dengan menggunakan elemen AIML bernama *srai.* Elemen *srai* dapat digunakan untuk mengimplementasikan persamaan arti seperti pada contoh berikut.

```
<category>
<pattern>dadah</pattern>
<template>Selamat tinggal/>.</template>
</category>
<category>
<pattern>sampai jumpa</pattern>
<template><srai>dadah</srai></template>
</category>
```

Pada *Category* yang berisikan *Pattern* "dadah" yang berarti inputan selamat tinggal yang akan dibalas "selamat tinggal". *Category* selanjutnya menjawab inputan "sampai jumpa" sama seperti inputan "dadah", dengan kata lain sebuah inputan akan diteruskan ke *category* yang dianggap sama dengan di sintakskan <srai> yang menunjukkan bahwa dia satu *category* inputan, maka dapat diartikan bahwa kedua inputan kata diatas bernilai sama(Wallace, 2003).

d. *That*

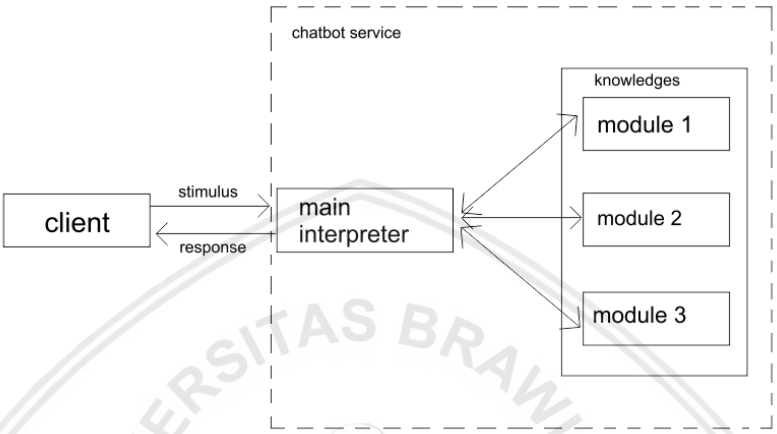
That adalah element AIML yang merujuk pada balasan, tanggapan sebelumnya. That digunakan pada category ketika balasan/tanggapan masih bersangkutan dengan inputan atau balasan sebelumnya.

Dalam perancangan template Wallace (2003) berpendapat bahwa *pattern* diartikan sebagai inputan bisa berarti sebuah pertanyaan, *template* diartikan sebagai sebuah balasan, *template* sendiri bisa berarti sebuah jawaban tunggal maupun jawaban random.

2.2.3 Program O

Program O adalah implementasi baru bot AIML yang ditulis dalam PHP. Masih di fase awal pembangunan. Fitur yang paling menonjol adalah algoritme pencocokan pola. Pola untuk tiga konteks dasar disimpan secara keseluruhan dalam tabel database, dan alih-alih dengan mencocokkan pola dengan input, ekspresi reguler dibuat untuk setiap masukan dan database kemudian dicari

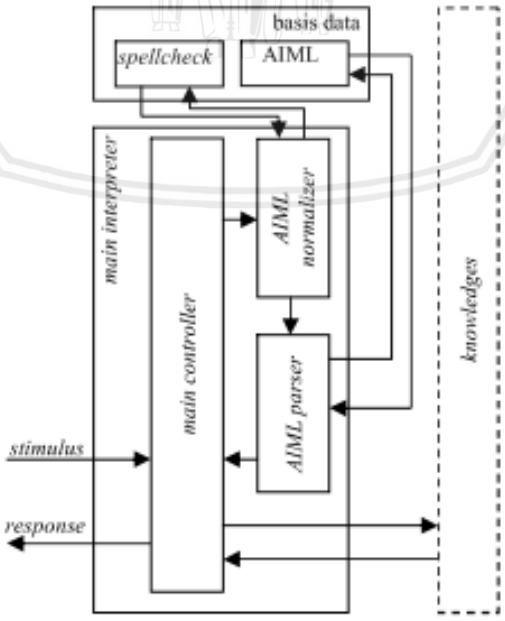
string pola yang sesuai dengan ekspresi reguler ini. Memilih pola terbaik tidak dilakukan dengan menggunakan algoritme default. Sebaliknya, setiap kata dan wildcard. Dari pola tersebut memberikan kontribusi terhadap skor akhir dengan bobot yang bervariasi (Sullivan, 2009). Berdasarkan studi dari Setiaji (2013) *Program O* merupakan penerjemah *chatbot* menggunakan PHP dan AIML, sistem penerjemah ini mampu melakukan CRUD (Create Read Update Delete) pada pengetahuan *Chatbot*.



Gambar 2.1 Gambaran Umum *Chatbot*

Sumber: Setiaji (2013)

Pada Gambar 2.1 merupakan Gambaran umum mengenai cara kerja *Chatbot* dimana *client* memberikan *stimulus* lalu *main interpreter* mencocokan dengan *knowledges* yang modul-modul, modul ini berisikan topik yang spesifik dan berbeda setiap modulnya.



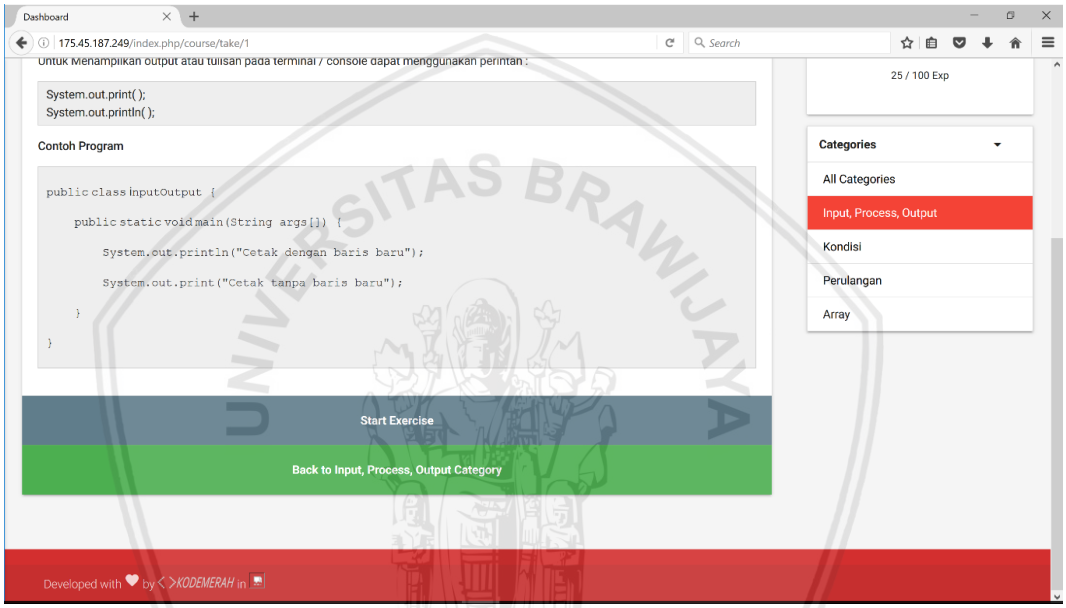
Gambar 2.2 Detail Perancangan *main interpreter*

Sumber: Setiaji (2013)

Pada Gambar 2.2 merupakan detail pada rancangan *main interpreter* dimana berisikan *main controller* yang menampung inputan atau *stimulus* dari pengguna, menyiapkan output atau *response*dan mengirimkan kembali ke pengguna.

2.2.4 CoMa (Code Maniac)

Code Maniac merupakan sebuah aplikasi yang menggunakan metode gamification dan *IOE-Behavior*, dimana pada metode *gamification* ini menerapkan fitur *level*, penghargaan, fitur untuk menantang *member* lain, Pada Code Maniac ini sistem memeriksa barisan kode java yang dimasukan oleh *member* dibandingkan dengan kunci kode java yang terdaftar (Bastari, 2017).



Gambar 2.3 Tampilan Exercise pada Code Maniac

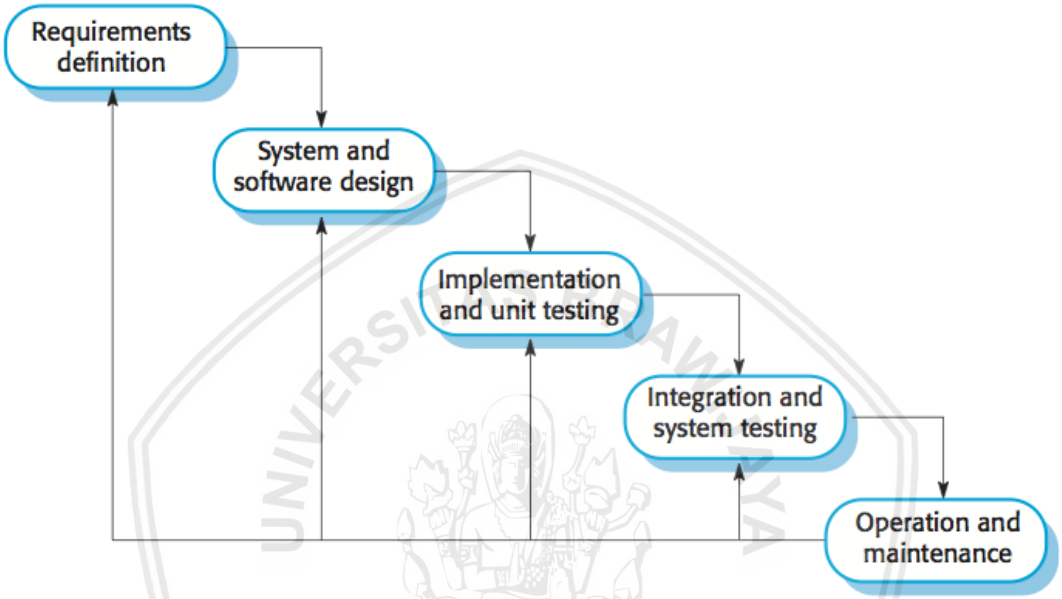
Pada Gambar 2.3 menjelaskan salah satu fitur utama pada Code Maniac adalah *exercise* dimana pada fitur ini *member* atau pengguna dituntut untuk menyelesaikan suatu pertanyaan pada setiap topik, ketika *member* telah menyelesaikan pertanyaan tersebut maka *member* akan mendapatkan exp (Bastari, 2017).

2.3 RPL (Rekayasa Perangkat Lunak)

Berdasarkan studi oleh Pressman (2001) RPL merupakan pengembangan dan pengaplikasian disiplin dan prinsip keahlian teknik dalam pengembangan perangkat lunak yang efisien, efektif, handal, terjangkau pada lingkup yang sesungguhnya. Rekayasa Perangkat Lunak berdasarkan studi oleh Sommerville (2001) merupakan penerapan sebuah prinsip perekayasaan perangkat lunak yang mencakup seluruh tahapan atau fase, dari tahap spesifikasi sistem perangkat lunak hingga perawatan perangkat lunak yang telah digunakan oleh masyarakat.

2.3.1 Pemodelan

Royce (1970) dalam studinya menyatakan sebuah pemodelan pengembangan perangkat lunak, dimana sebuah pemodelan tersebut mengalir dari satu tahapan paling atas ke tahapan dibawahnya dan dinamakan sebagai *Waterfall Model*. Menurut studi dari Sommerville(2011) Pemodelan *Waterfall* ini merupakan sebuah proses rekayasa perangkat lunak yang terencana dimana tidak ada perubahan ketika proses pengembangan perangkat lunak berjalan. Gambar 2.4 merepresentasikan tahapan pemodelan *waterfall*.



Gambar 2.4 Model Waterfall

Sumber: Sommerville (2010)

Pada Gambar 2.4 merupakan model *waterfall* semua tahapan dilakukan secara runtut. Pada penelitian ini menggunakan UML dalam melakukan pemodelan sistem. Berdasarkan studi dari Hermawan (2004), "*Unified Modelling Language* merupakan bahasa untuk memodelkan sebuah sistem sehingga mampi mengambil keputusan dan memahami mengenai sistem yang akan dibangun". UML merupakan sebuah cara memodelkan suatu perancangan sistem berorientasi objek, dimana interaksi sistem dengan subsistem serta sistem lain. Rekayasa perangkat lunak terpusat pada desain dan pengembangan perangkat lunak dimana memahami sistem secara keseluruhan, interaksi antar objek didalam sistem, pengujian perangkat lunak yang telah di rekayasa apakah sesuai dengan perencanaan dan dokumentasi perangkat lunak tersebut.

2.3.2 Analisis dan Perancangan

2.3.2.1 Analisis Berorientasi Objek (*Object-Oriented Analysis/OOA*)

Analisis berorientasi objek adalah teknik pemodelan yang menggabungkan data dan proses menjadi sebuah konsep yang disebut objek. OOA menurut Whitten (2007) adalah sebuah representasi objek sistem (struktur, fungsi dan interaksi antar objek).

2.3.2.2 Perancangan Berorientasi Objek (*Object-Oriented Design / OOD*)

Perancangan berorientasi objek menurut Whitten (2007) merupakan sebuah proses perancangan antar objek yang berhubungan dalam perangkat lunak, dimana tentang keterkaitannya dengan objek, atribut dan fungsinya secara bersamaan (Whitten & Bentley, 2007).

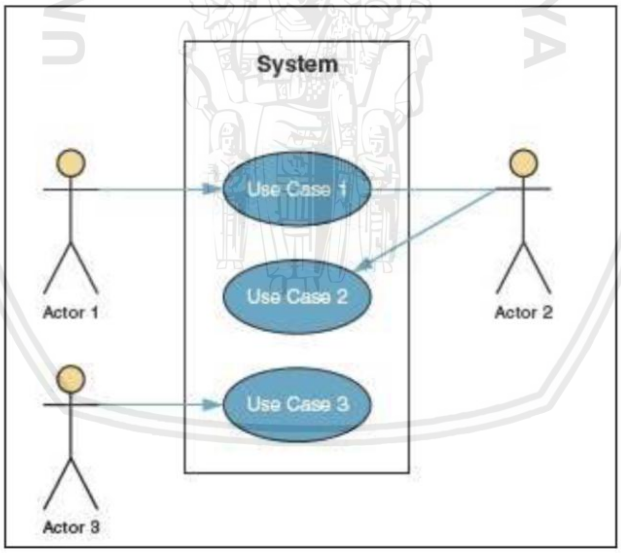
2.3.2.3 UML (Unified Modifying Language)

UML menurut Whitten (2007) adalah merupakan kesepakatan pemodelan yang dipergunakan untuk memaparkan atau mendeskripsikan sebuah perangkat lunak yang terkait dengan objek. Berikut berupakan tipe diagram dalam UML:

a. *Usecase* Diagram

Use-case diagram menurut Whitten(2017) merupakan diagram dimana merepresentasikan interaksi/hubungan antara aktor dengan sistem. Diagram ini mendeskripsikan siapa saja yang akan menggunakan sistem, upaya apa saja untuk mencapai suatu proses tersebut dan bagaimana hubungan pengguna dengan sistem.

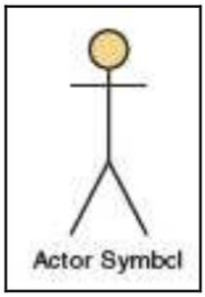
1. *Usecase* adalah urutan tindakan saling terkait dalam melengkapi suatu perilaku tunggal yang ditunjukkan pada Gambar 2.5.



Gambar 2.5 Contoh Diagram *Usecase*

(Sumber: Whitten & Bentley, 2007)

2. Aktor adalah seseorang yang terlibat atau berinteraksi dengan system dapat disimbolkan seperti pada Gambar 2.6.

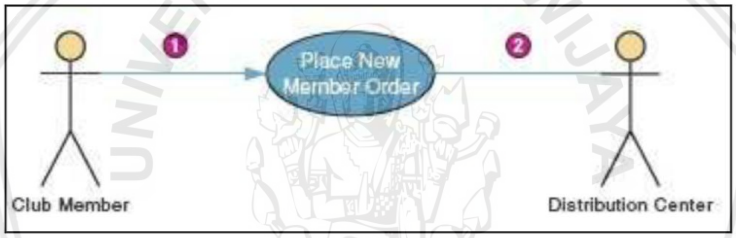


Gambar 2.6 Simbol Aktor

(Sumber: Whitten & Bentley, 2007)

3. Hubungan (*Relationship)* pada diagram *usecase* diGambarkan dengan garis antara dua simbol. Dalam hubungan terbagi menjadi *association, generalization, extend,* dan *include*.

   1) Gabungan (*Association*): sebuah interaksi antara aktor dengan *usecase*, satu *usecase* dapat berinteraksi dengan satu aktor atau lebih.

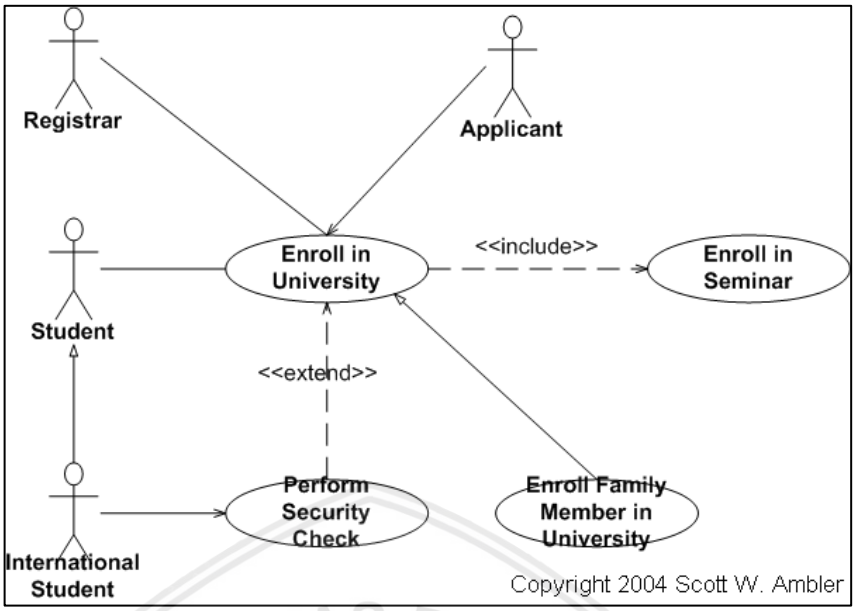


Gambar 2.7 *Association*

(Sumber: Whitten & Bentley, 2007)

Pada Gambar 2.7 merupakan suatu hubungan assosiasi aktor club member dengan fungsi *place new member order* serta *distribution center* dengan *place new member order*.

   2) *Extends:* dimana menjelaskan opsional *usecase* yang dapat dikerjakan jika mengakses/menjalankan *usecase* sebelumnya dan dapat berdiri sendiri tanpa *usecase* yang lain, direpresentasikan dengan diagram pada Gambar 2.8.
   3) *Include:* dimana menjelaskan *usecase* tambahan yang dapat dikerjakan jika mengakses/menjalankan *usecase* sebelumnya dan tidak dapat berdiri sendiri tanpa *usecase* yang lain, direpresentasikan dengan diagram pada Gambar 2.8.



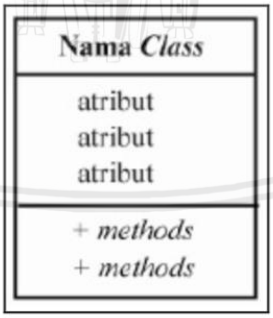
Gambar 2.8 *Extend Include*

(Sumber: Ambler W. S, 2004)

b. *Class* Diagram

Pada studi Whitten *(2007) Class* diagram merepresentasikan suatu struktur objek pada sistem, menunjukkan objek serta relasi antar objek- objek tersebut.

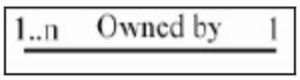
1) *Class* dapat direpresentasikan sesuai dengan Gambar 2.9 dimana sebuah persegi yang terurai atas tiga bagian. Bagian teratas adalah bagian nama dari *class*. Bagian sentral mendefinisikan atribut *class*. Bagian terakhir mendeskripsikan method dari sebuah *class*.



Gambar 2.9 *Class*

(Sumber: Whitten & Bentley, 2007)

2) *Association* merupakan sebuah hubungan yang paling umum antara dua *class* dan dilambangkan oleh sebuah garis yang menghubungkan antara dua *class*. Garis ini bisa melambangkan tipe-tipe hubungan. Contoh diGambarkan pada Gambar 2.10.



Gambar 2.10 *Association*

(Sumber: Whitten & Bentley, 2007)

3) *Composition* adalah sebuah *class* tidak bisa berdiri sendiri karena sebuah bagian dari *class* yang lain. Sebuah hubungan composition disimbolkan dengan garis yang berujungkan jajaran genjang berisi/solid dapat dilihat pada Gambar 2.11.

Gambar 2.11 *Composition*

(Sumber: Whitten & Bentley, 2007)

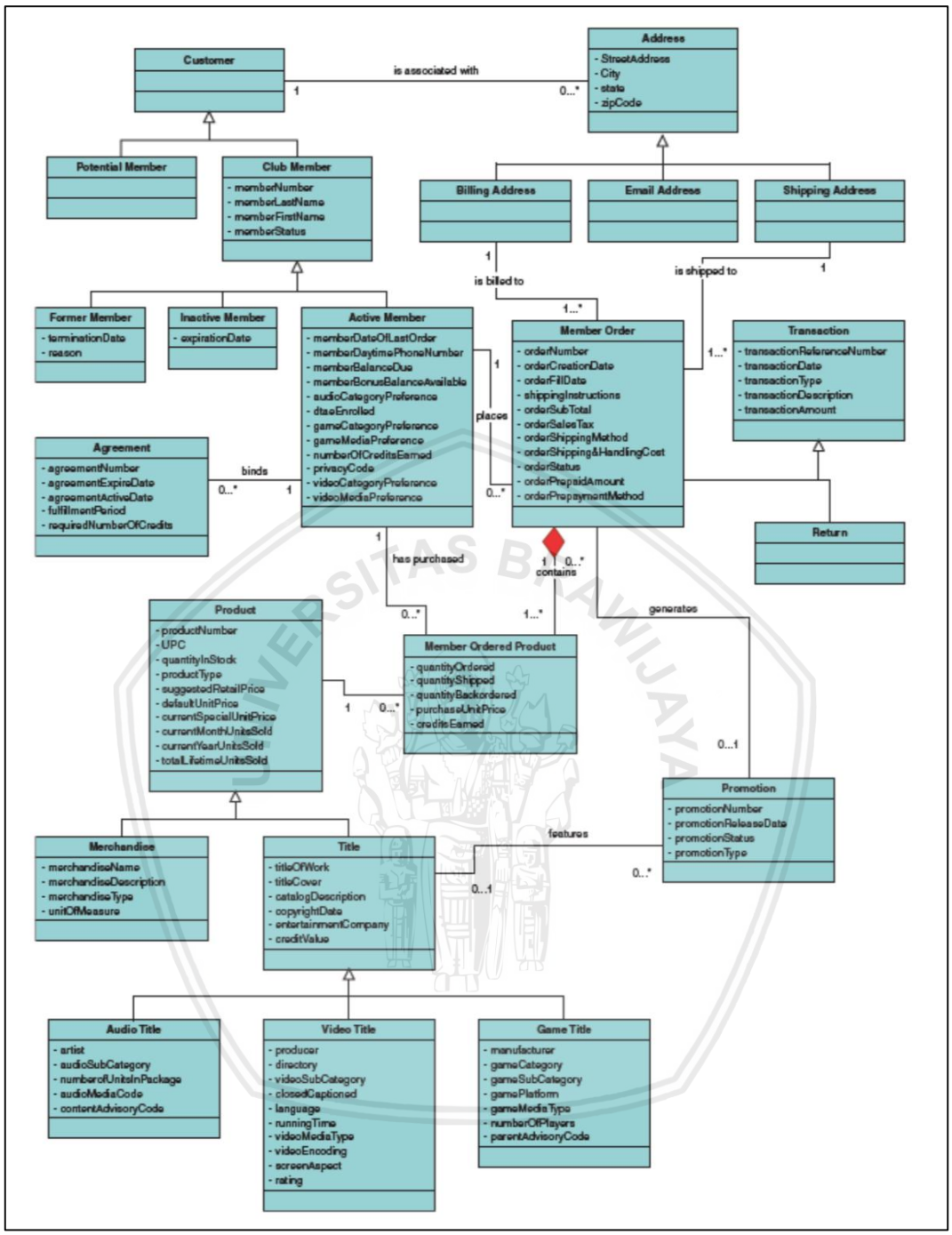
4) *Agregation* adalah sebuah *class* yang dapat berdiri sendiri meskipun merupakan bagian dari *class* lainnya dan dapat diGambarkan seperti Gambar 2.12.

Gambar 2.12 *Agregation*

(Sumber: Whitten & Bentley, 2007)

5) Pada Gambar 2.13 adalah contoh *class* diagram dan cara mengakses *visibility* UML menyediakan tiga tingkat dari *visibility*, yaitu.

   1. Public : ditandai dengan simbol “+”
   2. Protected : ditandai dengan simbol “#”
   3. Private : ditandai dengan simbol “-‘



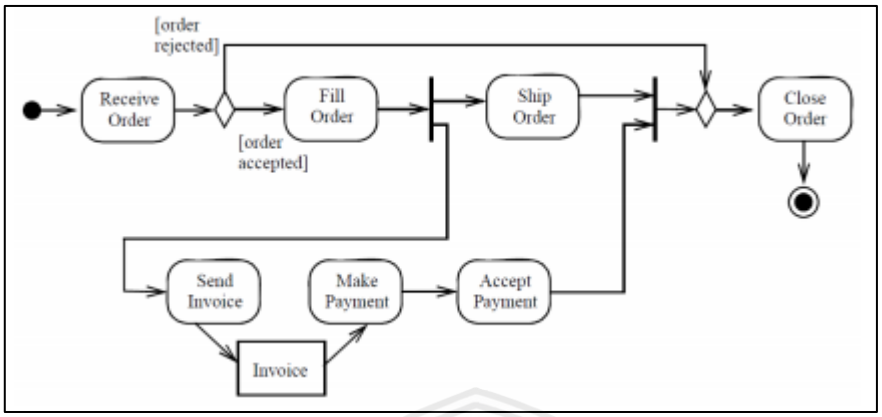
Gambar 2.13 Contoh *Class* Diagram

(Sumber: Whitten & Bentley, 2007)

c. *Activity* Diagram

Activity diagram menurut Whitten (2017) adalah sebuah diagram yang merepresentasikan sebuah alir kerja *usecase* dimana ada proses penyeleksian, perulangan maupun konkurensi (sumber daya yang diakses dalam waktu bersamaan) didalamnya. *Activity* diagram merepresentasikan kumpulan aktivitas

dengan menerangkan logika dari aktifitas yang dilakukan. Diagram ini juga mampu menerangkan aktifitas yang bersifat kondisional.



Gambar 2.14 Contoh Activity Diagram

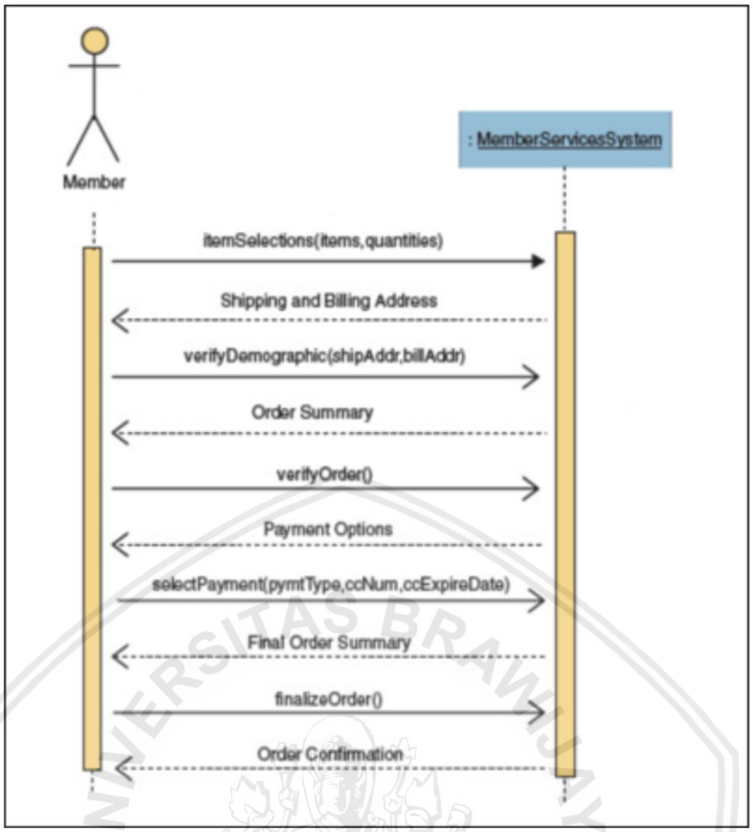
(Sumber UML v2.5.1)

Pada Gambar 2.14 adalah sekumpulan pelambangan pada *Activity diagram.*

1. *Initial Node*, sebuah lingkaran hitam yang menunjukkan awal dari sebuah proses.
2. *Flow*, sebuah anak panah yang menunjukkan ketahapan selanjutnya
3. *Action*, sebuah persegi dengan ujung tumpul yang menunjukkan aktifitas yang terjadi.
4. *Decision*, sebuah jajar genjang dengan satu buah *flow* masuk dan dua atau lebih *flow* keluar yang menunjukkan bahwa ada satu atau lebih kemungkinan yang akan terjadi.
5. *Merge*, sebuah jajargenjang dengan 2 atau lebih *flow* masuk yang sebelumnya terpisah karena *decision.*
6. *Fork*, sebuah garis hitam dengan satu *flow* masuk dan 2 atau lebih *flow* keluar yang menunjukkan proses paralel yang dapat terjadi secara bersamaan.
7. *Join*, sebuah garis hitam dengan dua atau lebih *flow* masuk dan satu *flow* keluar yang menunjukkan bahwa aksi yang masuk ke join harus telah diselesaikan sebelum proses berlanjut.
8. *Final Activity*, sebuah dua lingkaran dengan lingkaran hitam didalamnya yang menunjukkan akhir dari sebuah proses.

d. Sequence Diagram

Sequence diagram menurut Whitten (2007) adalah merepresentasikan cara kerja objek berinteraksi satu dengan yang lainnya melalui pesan dalam eksekusi dari sebuah operasi. Sebuah sistem *sequence diagram* menunjang untuk mengidentifikasi pesan yang masuk dan keluar dari sebuah sistem.



Gambar 2.15 Contoh *Sequence* Diagram

(Sumber: Whitten & Bentley, 2007)

Pada Gambar 2.15 adalah sekumpulan pelambangan didalam sequence diagram, yaitu.

4. *Actor*, berupa aktor yang memulai pada *usecase* ditunjukkan dengan simbol *usecase* aktor.
    1. *System*, sebuah kotak yang menunjukkan sistem yang sedang berjalan.
    2. *Lifelines*, sebuah garis putus-putus menurun dari simbol aktor dan sistem.
    3. *Activation bars*, sebuah balok yang terletak di atas *lifelines*, periode waktu interkasi ditunjukkan oleh balok ini.
    4. *Input messages*, sebuah pesan masuk yang ditunjukkan oleh Gambar anak panah mendatar dari aktor ke sistem.
    5. *Output messages*, sebuah pesan keluar yang ditunjukkan oleh Gambar anak panah mendatar dengan garis putus-putus dari sistem ke aktor.
    6. Receiver *Actor*, berupa aktor lain yang menerima pesan dari sistem.
    7. *Frame*, sebuah kotak yang menambahkan pesan terpisah untuk menunjukkan perulangan, alternative, atau pilihan.

2.3.2.4 Implementasi

Implementasi hasil dari perancangan ke dalam barisan kode (*coding*) sesuai dengan sintaks dari bahasa pemrograman yang dipilih. Implementasi diartikan sebagai interaksi antara penyusunan sebuah tujuan dengan sarana-sarana tindakan dalam mencapai tujuan tersebut, atau kemampuan untuk menghubungkan dalam hubungan sebab-akibat antara yang dimaksud, dengan cara untuk mencapainya (Pressman, 2010). Terdapat dua metode dalam pemrograman, yaitu pemrograman terstruktur (*Structured Programming - SP*) dan pemrograman berorientasi objek (*Object Oriented Programming - OOP*).

2.3.2.5 Pengujian

2.3.2.6 Blackbox Testing

Blackbox testing menurut Pressman (2010) merupakan pengujian yang berfokus terhadap persyaratan dari perangkat lunak yang memungkinkan penguji untuk mendapatkan set, kondisi, dan input yang secara keseluruhan melakukan persyaratan fungsional untuk sebuah program. *Blackbox testing* dilakukan dengan anggapan dasar penguji tidak mengetahui sistem didalam aplikasi yang sedang diuji, yang diuji hanya berdasarkan inputan penguji dan output apa yang dikeluarkan oleh sistem.

2.3.2.7 Whitebox Testing

Whitebox testing menurut Pressman (2010) adalah sebuah pendekatan dalam pengujian dimana tes berdasarkan pada pengetahuan penguji mengenai struktur program serta komponen didalamnya. Pada *white-box* testing pengujian dianggap sudah mengetahui sistem dari general hingga terkecil (Unit). Jalur independen kode program dapat diketahui dari *flow graph*. *Flow graph* merupakan diagram yang menjelaskan alur kode program.

2.3.2.8 Usability Testing

Usability testing dilakukan untuk menguji sebuah sistem dengan cara mengujikan langsung kepada sejumlah responden untuk mengetahui tingkat kemudahan penggunaan sebuah sistem. Dalam pengujian usabilitas ini menggunakan *System Usability Scale* (SUS). Penelitian yang dilakukan oleh Brooke (1986) SUS digunakan untuk mengevaluasi berbagai jenis produk, dari perangkat lunak, perangkat keras, website hingga aplikasi.

2.4 Motivasi

Motivasi berasal dari kata "*MOVERE*" yang berarti dorongan atau daya penggerak. Sebuah motivasi dipengaruhi oleh faktor eksternal dan internal. Michael J. Jucius mengatakan motivasi adalah kegiatan yang memberikan dorongan luar maupun dalam kepada orang lain maupun diri sendiri untuk mengambil suatu tindakan.

Pada proses pembelajaran terdapat beberapa elemen kunci yang mampu mendorong dan mempertahankan motivasi peserta didik yaitu perhatian, relevansi, percaya diri dan kepuasan. keempat elemen ini merupakan parameter motivasi para peserta didik apakah proses pembelajaran dilakukan secara tepat atau tidak (Keller, 2008). Keller mengemukakan model ARCS (*Attention, Relevance, Confidence, Satisfaction*) pada model ini peniliti memakai salah satu model pada ARCS yaitu *Attention*.

1. *Attention* atau perhatian.

   Perhatian dapat diperoleh baik oleh gairah perseptual ataupun gairah penyelidikan Dalam kasus gairah perseptif, perhatian para pembelajar akan diperoleh dengan kejutan, keraguan atau ketidakpercayaan. Untuk pertanyaan, rasa ingin tahu peserta didik akan dirangsang oleh tantangan masalah yang perlu dipecahkan. Untuk meraih dan menarik perhatian peserta, berbagai metode dapat digunakan, termasuk: partisipasi aktif, penggunaan humor, konflik, variasi serta contoh-contoh masalah dunia nyata(Keller, 2008).

2.5 RESTful Web Service

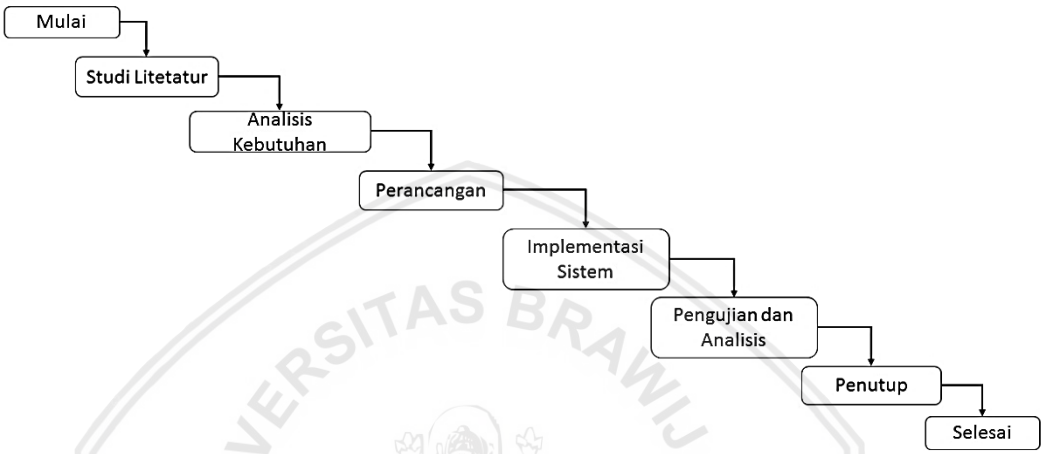
Representational State Transfer (REST) merupakan bagian dari HTTP (*Hypertext Transfer Protocol*) dimana menyediakan antarmuka yang dapat membuat, mengambil, memperbarui, menghapus dan memanipulasi sumber daya dengan pertukaran representasi (Abeysinghe, 2008).

Dimana *RESTful Web Service* menerapkan metode HTTP, diantaranya:

1. GET, mendapatkan sebuah sumber daya yang diidentifikasi dengan URI (Uniform Resource Identifier).
2. POST, mengirimkan sumber daya ke server. Digunakan untuk membuat sumber daya baru.
3. PUT, mengirimkan sumber daya ke server. Digunakan untuk memasukkan atau memperbarui sumber daya yang telah ada.
4. DELETE, menghapus sumber daya yang diidentifikasi dengan URI.

BAB 3 METODOLOGI

Metode penelitian merupakan tahapan peneliti dalam memecahkan suatu permasalahan pada penilitian dengan sistematis. Dimana metodologi penelitian terdiri dari tahapan studi literatur, analisis kebutuhan, perancangan sistem, implementasi, pengujian dan analisis. Pada Gambar 3.1 adalah diagram alir dari metodologi penelitian.



Gambar 3.1 Diagram Metode Penelitian

Pada bab ini dijelaskan perihal tahapan yang akan dilakukan untuk menanggulangi masalah yang dibahas. Tahapan awal dari studi literatur yaitu merumusan masalah. Kemudian melaksanakan analisis kebutuhan dimana melaksanakan pengumpulan data. Langkah selanjutnya adalah perancangan sistem meliputi desain antarmuka, lalu melakukan implementasi dari perancangan sistem yang dilakukan sebelumnya, setelah itu dilakukan pengujian serta evaluasi terhadap implementasi tersebut. Selepas mendapatkan hasil yang sesuai langkah selanjutnya ialah penarikan kesimpulan dan saran.

3.1 Studi Literatur

Studi Literatur dilakukan dengan mempelajari ilmu mengenai dasar-dasar pengembangan *Chatbot* dan hal - hal yang lain yang mendukung. Referensi yang dibutuhkan berhubungan dengan AIML. Dasar teori pendukung diperoleh dari sumber seperti jurnal, artikel, buku, dan penelitian sebelumnya yang berkaitan dengan topik penelitian ini. Akan halnya teori-teori yang berkaitan dengan penelitian ini antara lain :

1. *Chatbot*
2. *Artificial Intelligence Markup Language* (AIML)
3. Program O
4. RESTful Web Service

3.2 Analisis Kebutuhan

Analisis kebutuhan dilakukan untuk menggali mengetahui kebutuhan yang akan digunakan pada proses perancangan yang nantinya menjadi fungsional pengguna. Metode dalam analisis kebutuhan pada penelitian ini menggunakan Object Oriented Analysis (OOA) dengan memodelkan kebutuhan menggunakan bahasa Unified Modelling Language (UML). Terdapat proses yang perlu dilakukan dalam melakukan analisis kebutuhan yaitu:

1. Elisitasi kebutuhan

   Penggalian kebutuhan pada penelitian lanjut ini dilakukan dengan wawancara serta studi dokumen. Wawancara dan studi dokumen dilakukan dengan tujuan menggali kebutuhan dan memahami masalah dengan baik. Wawancara dilakukan kepada mahasiswa Teknik Informatika Fakultas Ilmu Komputer Universitas Brawijaya pada angkatan 2013 dan 2014 yang berjumlah 20 orang. Wawancara dilakukan dengan memberikan set pertanyaan tentang permasalahan yang ada dan solusi yang diharapakan, jawaban dari pertanyaan wawancara akan membantu peneliti dalam menentukan solusi dan kebutuhan.

2. Spesifikasi kebutuhan

   Kebutuhan tambahan yang telah didapatkan dari fase elisitasi kebutuhan dijelaskan secara mendetail. Kebutuhan yang telah dispesifkasi akan dimodelkan menggunakan usecase diagram dan usecase scenario.

3. Validasi dan verifikasi kebutuhan

   Tahap validasi dan verifikasi dilakukan agar kebutuhan yang telah didapatkan dan didefinisikan dengan benar, akurat dan lengkap. Proses validasi dan verifikasi dilakukan dengan membuat prototype agar membantu peneliti dalam memahami kebutuhan dan memastikan bahwa kebutuhan yang telah didefinisikan dan dispesifikasikan telah benar.

4. Manajemen Kebutuhan

   Dalam menjaga konsistensi dari setiap kebutuhan yang telah didapatkan maka setiap kebutuhan diberikan kode unik yang membedakan antas satu kebutuhan dengan kebutuhan lainnya, Setiap Kebutuhan akan diberikan kode SRS-CM-F-Z-X untuk kebutuhan fungsional dan SRS-CM-NF-X untuk kebutuhan non fungsional.

3.3 Perancangan Sistem

Pada tahap perancangan, sistem akan dikerjakan sesuai dari analisis kebutuhan yang telah didefinisikan dan akan dirancang dengan metode OOD dan dimodelkan menggunakan UML . Pada perancangan ini akan menghasilkan:

1. Perancangan Arsitektur

   Pada perancangan arsitektur dilakukan pemodelan dengan diagram UML seperti sequence diagram dan class diagram.

2. Perancangan Komponen

   Pada perancangan komponen dituliskan algoritme dari tiga fungsionalitas utama dari sistem yang dibangun. Penulisan algoritme dari tiga fungsionalitas tersebut dituliskan dalam *pseudocode*.

3. Perancangan Basis Data

   Pada perancangan basis data digunakan *Conceptual Data Model* (CDM) untuk memodelkan relasi data pada database yang dibuat. CDM ini nantinya akan diimplementasikan menjadi Physical Data Model (PDM) pada tahap implementasi.

4. Perancangan Pengetahuan *Chatbot*

   Pada perancangan pengetahuan *Chatbot* dibentuk sebuah pola pertanyaan berdasarkan topik. Dimana pada suatu topik akan terbentuk pola pertanyaan namun dengan maksud tujuan yang sama,.

5. Perancangan Antarmuka

   Pada perancangan antarmuka, rancangan antarmuka sistem akan dibuat berupa *wireframe* dengan komponen-komponen pada antarmuka sistem. Pada setiap komponen *wireframe* akan diberi penjelasan untuk menghindari kesalahan pada tahap implementasi.

hasil informasi dari analisis kebutuhan untuk pengetahuan *Chatbot* akan dibuat daftar sesuai dengan tema maupun subjek nya, lalu menentukan bentuk-bentuk pertanyaan atau input berdasarkan suatu topik pembelajaran dan menyesuaikan pola pertanyaan dengan bentuk *pattern* AIML.

3.4 Implementasi Sistem

Pada tahap implementasi, perancangan template dan desain sistem yang telah dibuat diimplementasikan ke dalam kode program. Implementasi *Chatbot* menggunakan platform Website dan berbahasa pemrograman PHP, HTML, Javascript dan AIML. Implementasi dari data yang telah diolah yaitu basis pengetahuan (*knowledge based*) pada bot dapat membantu pengguna.

3.5 Pengujian dan Analisis

Pada tahap pengujian analisis, pengujian dilakukan dengan beberapa tahap yaitu:

1. Pengujian Unit.

Pengujian unit dilakukan dengan tujuan untuk menguji suatu komponen dari suatu sistem yang telah dibuat. Komponen yang diuji berupa *class* atau objek berdasarkan perancangan komponen. Metode pengujian *basis path testing* digunakan untuk melakukan teknik pengujian *whitebox testing*.

2. Pengujian Integrasi.

Pengujian Integrasi menggunakan strategi *top-down integration* dimana pengujian dilakukan dari modul tertinggi dan turun ke modul-modul dibawahnya. Pengujian Integrasi menggunakan teknik *blackbox testing*.

3. Pengujian Validasi.

Pengujian Validasi menggunakan teknik *blackbox testing* dimana pengujian ini bertujuan untuk menguji kesesuaian kebutuhan yang telah didefinisikan pada sistem yang dibangun. Pengujian ini hanya melihat kondisi masukan dan keluaran yang dihasilkan sistem.

4. Pengujian Pengetahuan *Chatbot*

Pengujian Pengetahuan *Chatbot* menggunakan teknik *blackbox testing* dimana pengujian ini memastikan bahwa setiap pertanyaan yang diberikan oleh pengguna dapat direspon oleh sistem berdasarkan pengetahuan yang ada. Pengujian ini dibantu dengan aplikasi *POSTMAN* untuk mengetahui setiap *key, value* diberikan telah direspon dengan benar oleh sistem.

5. Pengujian *Usability*

Pengujian ini dilakukan untuk memastikan bahwa sistem telah dibangun sesuai dan dapat digunakan dengan mudah, mudah dipahami serta digunakan oleh pengguna. Pengujian ini dilakukan ketika sistem telah siap digunakan oleh pengguna sehingga pengguna dapat merasakan kemudahan menggunakan sistem ini.

Pada tahap terakhir merupakan analisis keseluruhan aplikasi yang rekayasa sudah sesuai dengan perancangan yang telah dikemukakan sebelumnya.

3.6 Penutup

Penutupan merupakan penarikan kesimpulan dari analisis dan saran dari penulis setelah semua tahapan penelitian yang telah dilakukan. Kesimpulan dibuat berlandaskan hasil pengembangan aplikasi dan berisi mengenai saran jika penelitian ini dikembangkan lebih lanjut.

BAB 4 ANALISIS KEBUTUHAN

4.1 Gambaran Umum Aplikasi

Aplikasi Code Maniac merupakan aplikasi yang dibangun dengan tujuan untuk membantu mahasiswa dalam pembelajaran koding java. Pada pengembangan lanjut ini terfokus pada pengembangan chatterbot dimana pengguna dapat bertanya mengenai topik-topik koding java, peringatan terhadap pengguna ketika pengguna melakukan kesalahan ketika mengerjakan *exercise* yang diberikan. Inputan pertanyaan oleh pengguna akan diproses oleh sistem dengan mengimplementasikan AIML

4.2 Identifikasi Aktor

Aktor melambangkan seseorang yang mampu berinteraksi dengan sistem. Aktor dalam aplikasi ini dijelaskan pada Tabel 4.1.

Tabel 4.1 Identifikasi Aktor

Aktor	Deskripsi
Pengguna	Merupakan seseorang yang menggunakan aplikasi Code Maniac (CoMa)
Admin	Merupakan sesorang yang dapat mengelola data Code Maniac serta Pengetahuan *Chatbot*

4.3 Kebutuhan Fungsional Sistem

Spesifikasi beserta Kebutuhan fungsional sistem ditunjukkan pada Tabel 4.2. Setiap Kebutuhan akan diberikan kode SRS-CM-F-Z-X untuk kebutuhan fungsional dan SRS-CM-NF-X untuk kebutuhan non fungsional. SRS merupakan *System Requirement Spesification*, CM merupakan singkatan dari *CodeManiac*, F untuk kebutuhan fungsional, NF untuk kebutuhan non fungsional , Z menunjukkan nomor dari aktor dan X menunjukkan nomor dari definisi kebutuhan utama.

Tabel 4.2 Kebutuhan Fungsional Pengguna

No.	Kode Kebutuhan Sistem	Deskripsi Kebutuhan
1	SRS-CM-F-1-1	Sistem dapat menerima masukan berupa pertanyaan dari pengguna.
2	SRS-CM-F-1-2	Sistem dapat memberikan keluaran informasi berdasarkan pertanyaan pengguna.
3	SRS-CM-F-1-3	Sistem dapat memperingatkan kesalahan coding pengguna saat melakukan *live coding*.

4	SRS-CM-F-1-4	Sistem dapat memperingatkan saat pengguna tidak mengerjakan bab *course* secara runtut.

Tabel 4.3 Kebutuhan Fungsional Admin

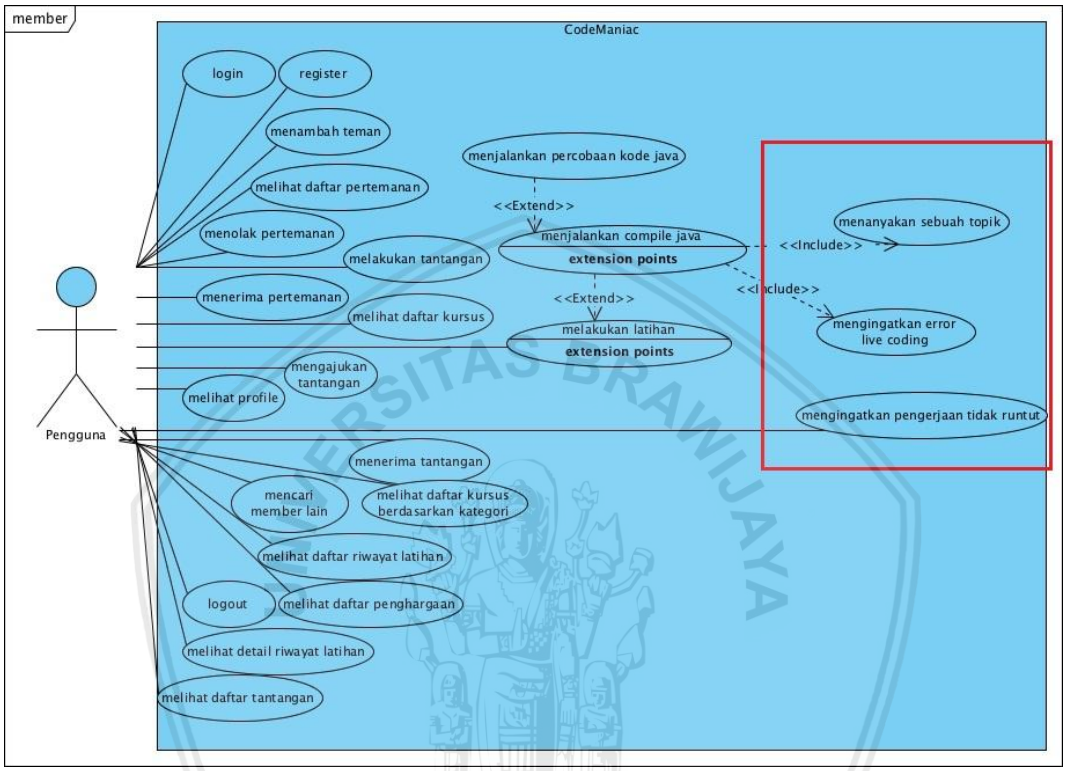
No.	Kode Kebutuhan Sistem	Deskripsi Kebutuhan
1	SRS-CM-F-2-1	Sistem dapat menampilkan log hasil percakapan pengguna dengan *chatbot*.
2	SRS-CM-F-2-2	Sistem dapat menambah pengetahuan *chatbot*.
3	SRS-CM-F-2-3	Sistem dapat Mengubah pengetahuan *chatbot*.
4	SRS-CM-F-2-4	Sistem dapat menghapus pengetahuan *chatbot*.
5	SRS-CM-F-2-5	Sistem dapat menambah pengetahuan berdasarkan konten *course* yang telah ditambahkan.

Tabel 4.4 Kebutuhan Non Fungsional Sistem

No.	Kode Kebutuhan Sistem	Deskripsi Kebutuhan
1	SRS-CM-NF-1	*Usability* yaitu kemudahan penggunaan dan memahami aplikasi yang digunakan oleh pengguna

4.3.1 *Diagram Usecase*

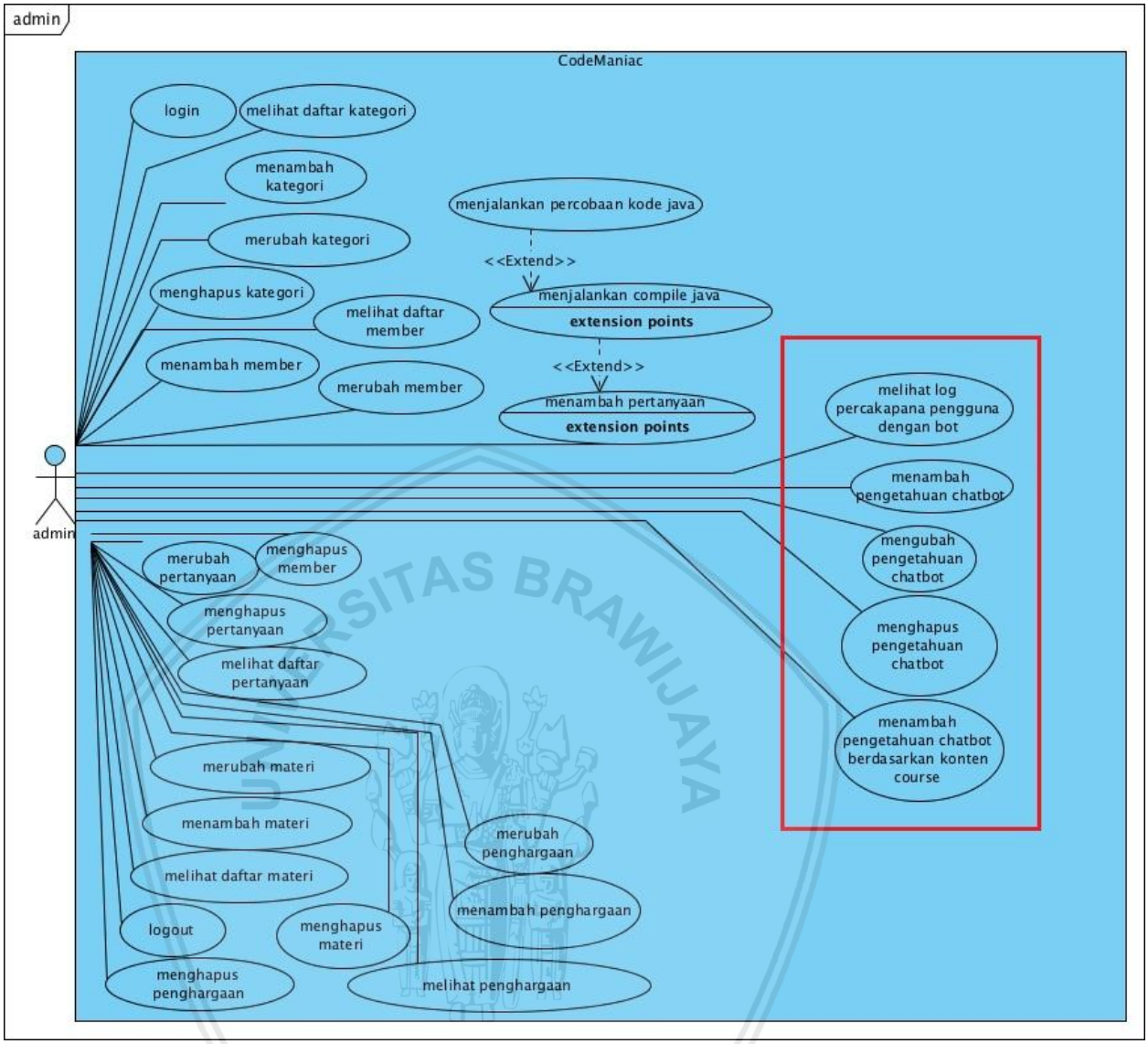
Diagram *usecase* berisi kumpulan aksi yang dilakukan oleh aktor kepada sistem untuk memperoleh suatu tujuan. Aktor pada diagram *usecase* dapat berupa manusia atau sistem eksternal yang berhubungan dengan sistem CodeManiac. Diagram *usecase* direpresentasikan dalam Gambar 4.1 dan Gambar 4.2.



Gambar 4.1 Diagram *Usecase* Sistem Code Maniac Aktor Pengguna

Berikut penjelasan dari diagram *use-case* pada Gambar 4.1:

Pada *use-case* peringatan error pada *live coding* aktor dapat menerima peringatan ketika melakukan *live coding* pada *exercise*. Kemudian pada *usecase* menanyakan topik aktor dapat menanyakan topik tertentu kepada *bot* pada *chatbox*. Pada *use-case* peringatan pengerjaan tidak runtut aktor dapat menerima peringatan ketika melakukan pengambilan suatu materi namun tidak runtut sesuai urutan materi.



Gambar 4.2 Diagram Usecase Sistem Code Maniac Aktor Admin

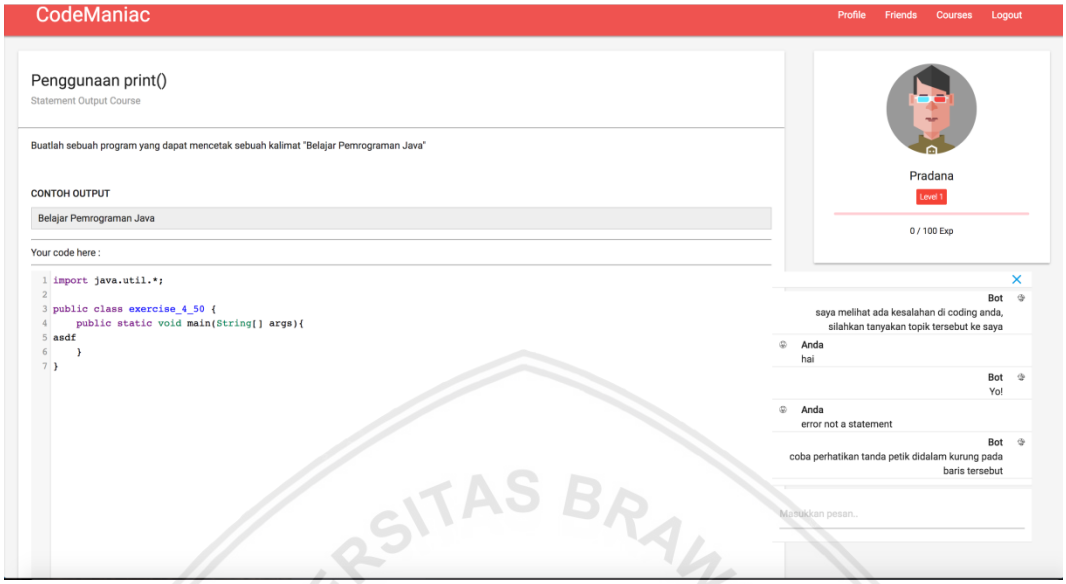
Berikut penjelasan dari diagram *use-case* pada Gambar 4.2:

Pada *use-case* melihat log percakapan pengguna dengan chatbot admin dapat melihat percakapan antara pengguna dengan chatbot. Pada *use-case* menambah, mengubah, dan menghapus pengetahuan chatbot admin dapat menambah, mengubah dan menghapus pengetahuan pada chatbot. Pada *use-case* menambah pengetahuan chatbot berdasarkan konten course admin dapat menambah pengetahuan chatbot berdasarkan konten course yang telah ada pada sistem.

4.4 Validasi dan Verifikasi

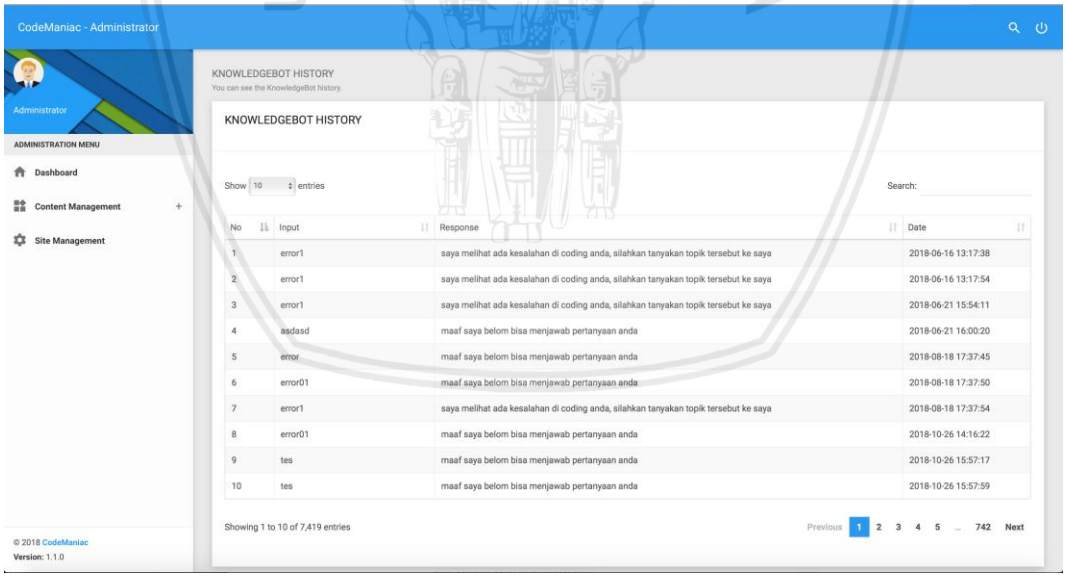
Validasi dan verifikasi bermaksud untuk mengetahui apakah sistem yang dibangun adalah benar. Proses validasi dan verifikasi menggunakan teknik *prototyping* dimana dilakukan pembuatan *prototype* dari kebautuhan yang didapatkan. Teknik ini mempermudah *stakeholder* untuk mendapatkan

Gambaran sistem yang akan dibuat serta untuk meminimalisir kesalahan yang terjadi. *Prototype* yang dibuat untuk proses validasi dan verifikasi dapat dilihat pada Gambar 4.2 – Gambar 4.4.



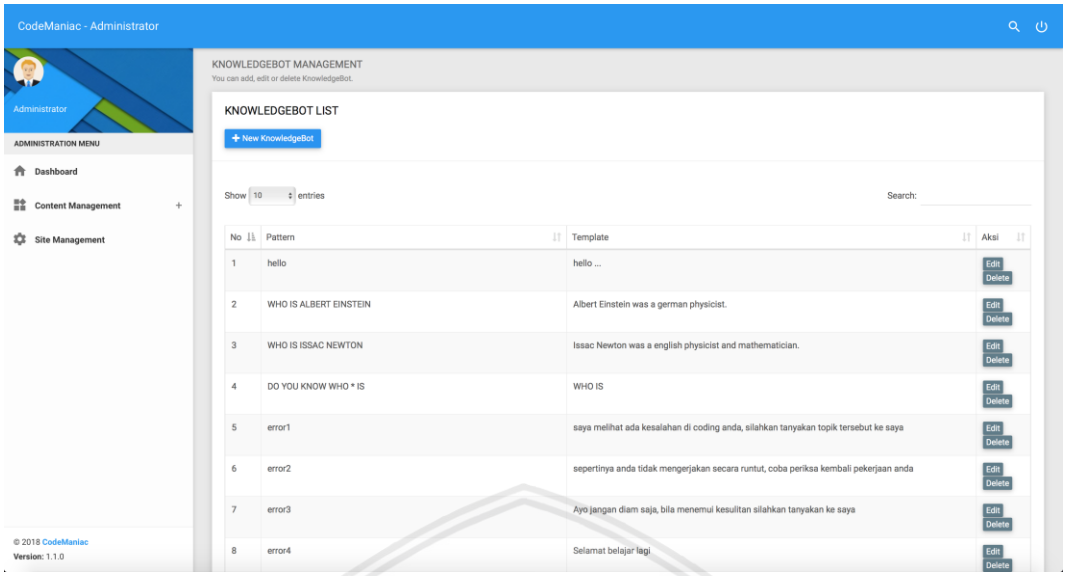
Gambar 4.3 Mockup *Chatbox*

Gambar 4.2 menunjukkan mockup dari kebutuhan *stakeholder* untuk *chatbox*



Gambar 4.4 Mockup Knowledgebot History

Gambar 4.3 menunjukkan mockup dari kebutuhan *stakeholder* untuk Knowledgebothistory.



Gambar 4.5 Mockup Knowledgebot

Gambar 4.4 menunjukkan mockup dari kebutuhan *stakeholder* untuk Knowledgebot.

4.5 Analisis Data

Analisis data bermaksud untuk memperoleh struktur data penyimpanan yang nantinya diimplementasikan pada aplikasi CodeManiac, data yang digunakan dalam aplikasi CodeManiac berdasarkan analisis sebagai berikut:

a. *Log* merupakan data yang mencatat aktivitas dari user, yang berisikan *user, bot, input, response, date.*

b. *Property* merupakan data yang menyimpan semua data dari *userInfo* dan *botInfo* yang berisikan *chatbox, name, type, unique, value.*

c. Data merupakan data yang menyimpan kata yang diucapkan *user* dan *bot, Parser s*ebagai *interpreter (*pengurai kata, mencari persamaan kata, dst), *Data* berisikan *unique ,data.*

Pada *AIML* menggunakan struktur data *Tree*, dimana AIML akan berisikan tentang pengetahuan *Chatbot* sebagai berikut:

a. Category merupakan sebuah inti dari pengetahuan yang berisikan pattern dan Template

b. Pattern merupakan sebuah pertanyaan atau sebuah pertanyaan yang dimasukan oleh pengguna.

c. Template merupakan sebuah respon atau jawaban dari pertanyaan yang dimasukan oleh pengguna.

4.6 Skenario *Usecase*

Skenario *usecase* merupakan penjabaran dari tiap *usecase* yang direpresentasikan pada diagram *usecase* sebelumnya. Pada skenario *usecase* terbagi dalam beberapa segmen, dari nama *usecase,* kode kebutuhan, aktor yang berinteraksi, tujuan *usecase, pre condition* tentang kondisi apa saja yg harus dipenuhi sebelum menjalankan *mainflow* pada *usecase, main flow* yaitu alur utama untuk mencapai tujuan sebuah *usecase, alternative flow* ketika alur utama tidak terpenuhi & *post condition* dimana kondisi terakhir aktor setelah aktor menjalankan *mainflow.* Diperoleh 8 *usecase scenario* yaitu Menanyakan sebuah topik, Mengingatkan error pada *live coding*, Mengingatkan pengerjaan tidak runtut, Melihat log percakapan pengguna dengan *Chatbot,* Menambah pengetahuan *Chatbot*, Mengedit pengetahuan *Chatbot,* dan Menghapus pengetahuan *Chatbot*. Skenario *usecase* akan ditunjukkan oleh TabeI 4.5 hingga TabeI 4.11.

Tabel 4.5 Skenario *Usecase* Menanyakan sebuah topik

Nama *Usecase*	Menanyakan sebuah topik
Kode Kebutuhan Terkait	SRS-CM-F-1-1, SRS-CM-F-1-2
Aktor	Pengguna
Tujuan	Menanyakan sebuah topik
Pre Condition	Aktor telah berada di halaman exercise dan telah menekan Gambar bot
Main flow	1. Pengguna memberikan pertanyaan terhadap bot 2. Sistem menerima pertanyaan lalu sistem menampilkan jawaban berdasarkan pengetahuan yang ada
Post Condition	Sistem akan menampilkan jawaban berdasarkan pertanyaan pengguna dan berdasarkan pengetahuan yang ada
Alternative Flow	Sistem akan menampilkan jawaban ‘maaf saya belum bisa menjawab pertanyaan anda’

Tabel 4.6 Skenario *Usecase* Mengingatkan error pada *live coding*

Nama *Usecase*	Mengingatkan error pada *live coding*
Kode Kebutuhan Terkait	SRS-CM-F-1-3
Aktor	Pengguna
Tujuan	Memperingatkan pengguna saat terjadi kesalahan pada *live coding*
Pre Condition	Pengguna telah berada di halaman exercise dan sedang melakukan *live coding*
Main flow	1. Pengguna melakukan *live coding* 2. Sistem meng-*compile source code* yang ditulis oleh pengguna di belakang layar tanpa sepengetahuan pengguna
Post Condition	Sistem akan menampilkan peringatan kesalahan coding pada chatbox jika terjadi kesalahan pada coding
Alternative Flow	-

Tabel 4.7 kenario *Usecase* Mengingatkan pengerjaan tidak runtut

Nama *Usecase*	Mengingatkan pengerjaan tidak runtut
Kode Kebutuhan Terkait	SRS-CM-F-1-4
Aktor	Pengguna
Tujuan	Memperingatkan pengguna saat pengambilan materi secara tidak runtut
Pre Condition	Pengguna telah berada di halaman course
Main flow	1. Pengguna memilih suatu materi 2. Sistem menampilkan materi yang dipilih oleh aktor
Post Condition	Sistem akan menampilkan peringatan pengambilan topik course yang tidak runtut pada chatbox jika terjadi pengambilan topik *course* yang tidak runtut
Alternative Flow	-

Tabel 4.8 Skenario *Usecase* melihat log percakapan pengguna dengan *chatbot*

Kode Kebutuhan Terkait	SRS-CM-F-2-1
Aktor	Admin
Tujuan	Admin melihat log percakapan antara pengguna dengan *chatbot*
Pre Condition	Admin telah masuk ke dalam sistem dan berada di halaman administrator
Main flow	1. Admin memilih menu "Content Management" 2. Sistem menampilkan daftar menu 3. Admin memilih menu "*Chatbot*History" 4. Sistem menampilkan seluruh riwayat percakapan antar pengguna dengan *chatbot*
Post Condition	Admin berhasil melihat riwayat percakapan antar pengguna dengan *chatbot*
Alternative Flow	-

Tabel 4.9 Skenario *Usecase* Menambah pengetahuan *chatbot*

Kode Kebutuhan Terkait	SRS-CM-F-2-2
Aktor	Admin
Tujuan	Admin dapat menambah pengetahuan *chatbot*.
Pre Condition	Admin telah masuk ke dalam sistem dan berada di halaman administrator
Main flow	1. Admin memilih menu "Content Management" 2. Sistem menampilkan daftar menu 3. Admin memilih menu "Knowledgebot" 4. Sistem menampilkan seluruh pengetahuan *chatbot* 5. Admin menekan tombol "New

	Knowledgebot” 6. Sistem menampilkan form knowledgebot list 7. Admin mengisi form knowledgebot list 8. Aktor menekan tombol save 9. Sistem menyimpan knowledgebot yang baru
Post Condition	Data knowledgebot bertambah di dalam sistem
Alternative Flow	1. Admin mengisi form tidak lengkap 2. Admin menekan tombol “save” 3. Sistem menampilkan informasi bahwa form harus lengapi

Tabel 4.10 Skenario *Usecase* mengubah pengetahuan *chatbot*

Kode Kebutuhan Terkait	SRS-CM-F-2-3
Aktor	Admin
Tujuan	Admin dapat mengubah pengetahuan *chatbot*.
Pre Condition	Admin telah masuk ke dalam sistem dan berada di halaman administrator
Main flow	1. Admin memilih menu “Content Management” 2. Sistem menampilkan daftar menu 3. Admin memilih menu “Knowledgebot” 4. Sistem menampilkan seluruh pengetahuan *chatbot* 5. Admin memilih salah satu pengetahuan bot lalu menekan tombol edit 6. Sistem menampilkan form knowledgebot list yang dipilih 7. Admin mengedit form knowledgebot list

	8. Aktor menekan tombol save 9. Sistem menyimpan knowledgebot yang baru
Post Condition	Data knowledgebot terbaru tersimpan di dalam sistem
Alternative Flow	1. Admin mengisi form tidak lengkap 2. Admin menekan tombol “save” 3. Sistem menampilkan informasi bahwa form harus dilengkapi

Tabel 4.11 Skenario *Usecase* menghapus pengetahuan *chatbot*

Kode Kebutuhan Terkait	SRS-CM-F-2-4
Aktor	Admin
Tujuan	Admin dapat menghapus pengetahuan *chatbot*.
Pre Condition	Admin telah masuk ke dalam sistem dan berada di halaman administrator
Main flow	1. Admin memilih menu “Content Management” 2. Sistem menampilkan daftar menu 3. Admin memilih menu “Knowledgebot” 4. Sistem menampilkan seluruh pengetahuan *chatbot* 5. Admin memilih salah satu pengetahuan bot lalu menekan tombol edit 6. Sistem menampilkan form knowledgebot list yang dipilih 7. Admin melihat form knowledgebot list 8. Aktor menekan tombol delete 9. Sistem menghapus knowledgebot yang dipilih admin
Post Condition	Data knowledgebot yang dipilih admin terhapus

Alternative Flow	-

Tabel 4.12 Skenario *Usecase* menambah pengetahuan berdasarkan konten *course* yang telah ditambahkan.

Kode Kebutuhan Terkait	SRS-CM-F-2-5
Aktor	Admin
Tujuan	Admin dapat menambah pengetahuan berdasarkan konten *course* yang telah ditambahkan.
Pre Condition	Admin telah masuk ke dalam sistem dan berada di halaman administrator
Main flow	1. Admin memilih menu "Content Management" 2. Sistem menampilkan daftar menu 3. Admin memilih menu "Courses" 4. Sistem menampilkan seluruh Course 5. Admin memilih salah satu Course lalu menekan tombol Generate Knowledge 6. Sistem memproses course dan menambahkannya pada knowledgebot
Post Condition	Sistem akan menambah knowledgebot berdasarkan course yang dipilih
Alternative Flow	-

BAB 5 PERANCANGAN DAN IMPLEMENTASI

5.1 Perancangan

Perancangan diIaksanakan berdasarkan hasiI dari analisis kebutuhan yang telah dilakukan sebelumnya. Proses perancangan CodeManiac ini dibagi menjadi enam tahap dimulai dari perancangan arsitektur, perancangan sequence diagram, perancangan *class*, perancangan komponen, perancangan basis data, perancangan pengetahuan *Chatbot* dan perancangan antarmuka.

5.1.1 Perancangan Arsitektur Sistem

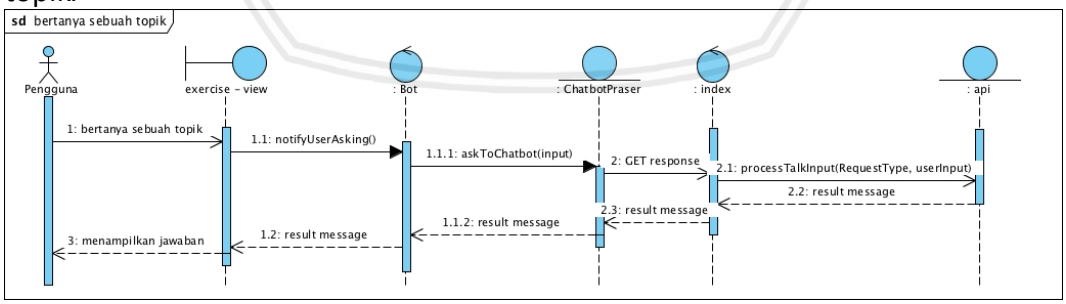
Aplikasi CodeManiac ini dibangun menggunakan framework CodeIgniter dimana CI sendiri menerapkan konsep MVC atau *Model View Controller* dimana *Model* merupakan *class* yang berisi kumpulan data dari aplikasi, *View* merupakan *class* yang dibangun sebagai antarmuka dari aplikasi dan *Controller* sebagai *class* yang mengatur proses kerja dari *class View*.

Ketika berinteraksi dengan CI, browser akan mengirimkan *request* kepada *web server* kemudian akan diteruskan ke sistem routing CI. Router CI akan memproses *request* kemudian mengalihkannya ke masing-masing *class* dan *method* sesuai dengan request url yang telah didefinisikan sebelumnya. Oleh *controller*, terjadi komunikasi dengan model jika diperlukan data yang berhubungan dengan *database*. Dalam beberapa kasus, *controller* akan melakukan render *view* yang nantinya akan dikonversi menjadi HTML dan dikirim kembali ke browser.

5.1.2 Sequence Diagram

1. *Diagram Sequence* menanyakan sebuah topik

Sequence diagram ini digunakan ketika pengguna menanyakan sebuah topik.

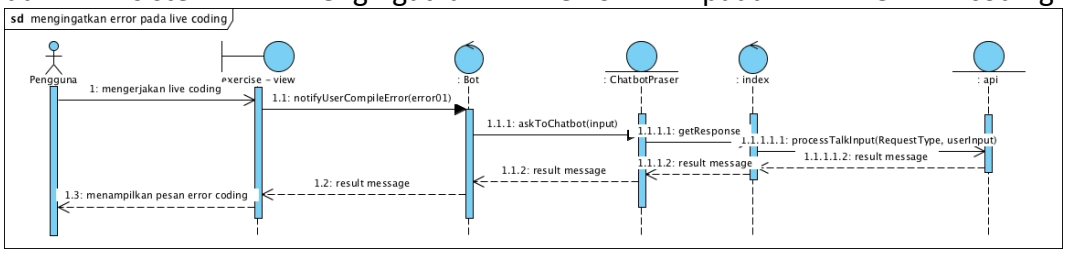


Gambar 5.1 Sequence diagram menanyakan sebuah topik di sistem *CodeManiac*

Gambar 5.1 Menjelaskan ketika pengguna menanyakan sebuah topik, maka sistem akan menjalankan fungsi notifyUserAsking lalu ChatbotParser menjalankan fungsi askToChatbot dan sistem menampilkan jawaban sesuai dengan pengetahuan yang ada.

2. *Diagram Sequence* mengingatkan error pada live coding

Sequence diagram ini digunakan ketika pengguna melakukan live coding dan sistem mengingatkan error pada live coding.

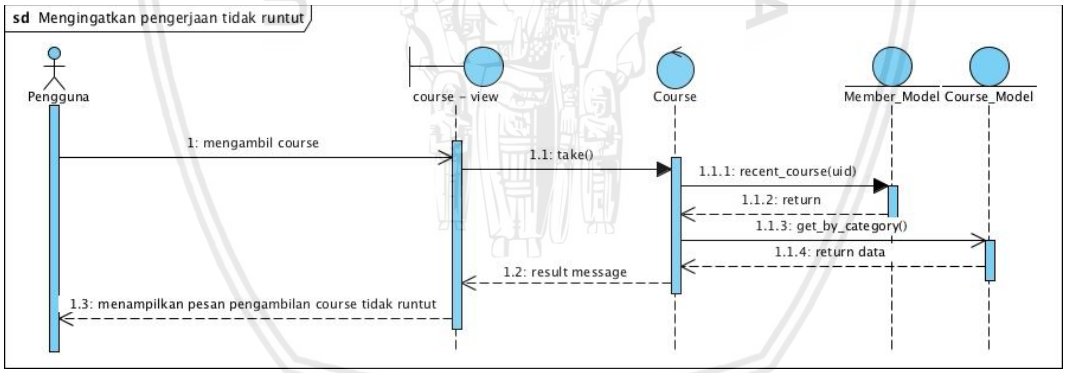


Gambar 5.2 Sequence diagram mengingatkan error pada live coding di sistem *CodeManiac*

Gambar 5.2 Menjelaskan ketika pengguna melakukan live coding dan melakukan kesalahan pada live coding, maka sistem akan menjalankan fungsi notifyUserCompileError lalu pada ChatbotParser akan menjalankan fungsi askToChatbot lalu sistem menampilkan peringatan error pada live coding.

3. *Diagram Sequence* mengingatkan pengerjaan tidak runtut

Sequence diagram ini digunakan ketika pengguna mengerjakan bab course secara tidak runtut.

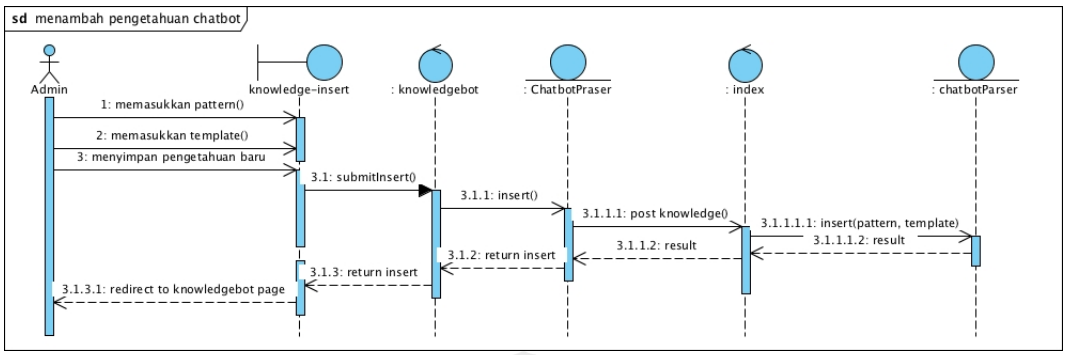


Gambar 5.3 Sequence diagram mengingatkan pengerjaan tidak runtut pada sistem *CodeManiac*

Gambar 5.3 Menjelaskan ketika pengguna mengambil suatu course namun tidak secara runtut, maka sistem akan menjalankan fungsi take lalu sistem akan menjalankan fungsi recent_course dan get_by_category untuk melihat apakah pilihan course yang diambil oleh pengguna berurutan atau tidak, jika pengguna tidak mengambil secara berurutan maka sistem akan menampilkan pesan pengambilan course tidak runtut.

4. *Diagram Sequence* Menambah Pengetahuan Bot

Sequence diagram ini digunakan ketika admin menambah pengetahuan *chatbot*

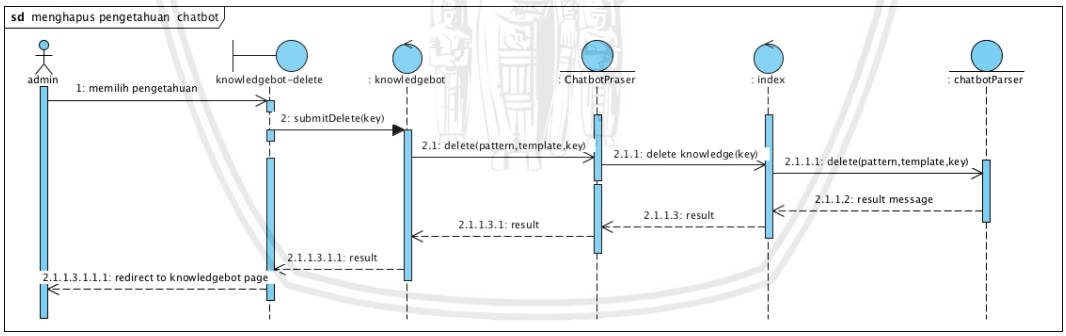


Gambar 5.4 Sequence diagram menambah pengetahuan *chatbot pada CodeManiac*

Gambar 5.4 menjelaskan ketika admin menambahkan pengetahuan *chatbot*, maka sistem akan menjalankan fungsi submitInsert dan menambahkan pengetahuan kedalam sistem lalu akan menampilkan kembali halaman knowledge bot.

5. *Diagram Sequence* Menghapus Pengetahuan Bot

Sequence diagram ini digunakan ketika admin menghapus pengetahuan *chatbot*

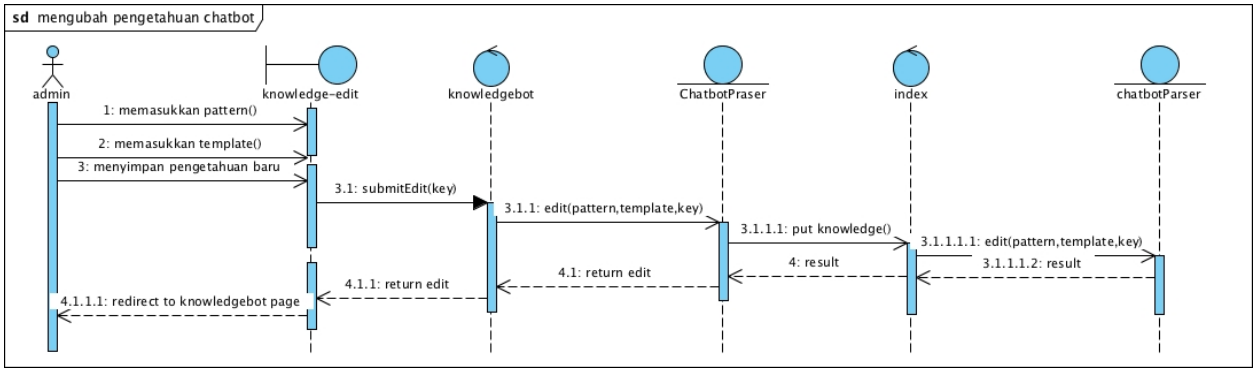


Gambar 5.5 Sequence diagram menghapus pengetahuan *chatbot* di sistem *CodeManiac*

Gambar 5.5 menjelaskan ketika admin menghapus pengetahuan *chatbot*, maka sistem akan menjalankan fungsi submitDelete dan menghapus pengetahuan kedalam sistem lalu akan menampilkan kembali halaman knowledge bot.

6. *Diagram Usecase* Mengubah Pengetahuan Bot

Sequence diagram ini digunakan ketika admin mengubah pengetahuan *chatbot*.

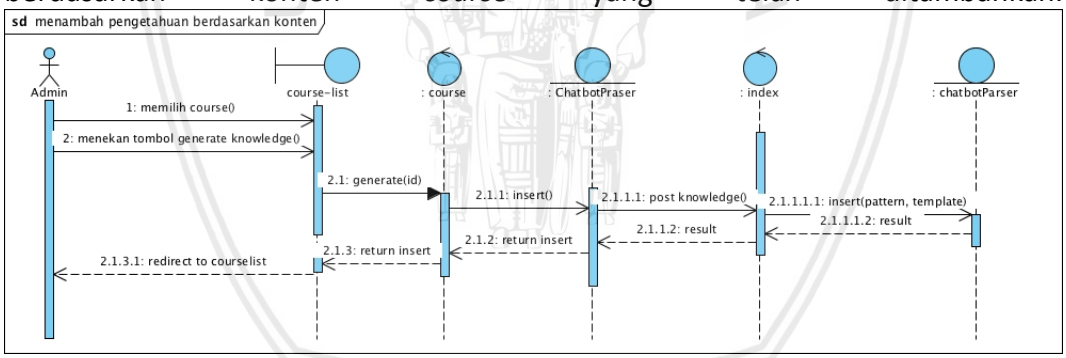


Gambar 5.6 Sequence diagram mengubah pengetahuan *chatbot pada CodeManiac*

Gambar 5.6 menjelaskan ketika admin mengubah pengetahuan *chatbot*, maka sistem akan menjalankan fungsi submitEdit dan mengubah pengetahuan kedalam sistem lalu akan menampilkan kembali halaman knowledge bot.

7. *Diagram Usecase* Menambah Pengetahuan berdasarkan konten *course*

Sequence diagram ini digunakan ketika menambah pengetahuan berdasarkan konten *course* yang telah ditambahkan.

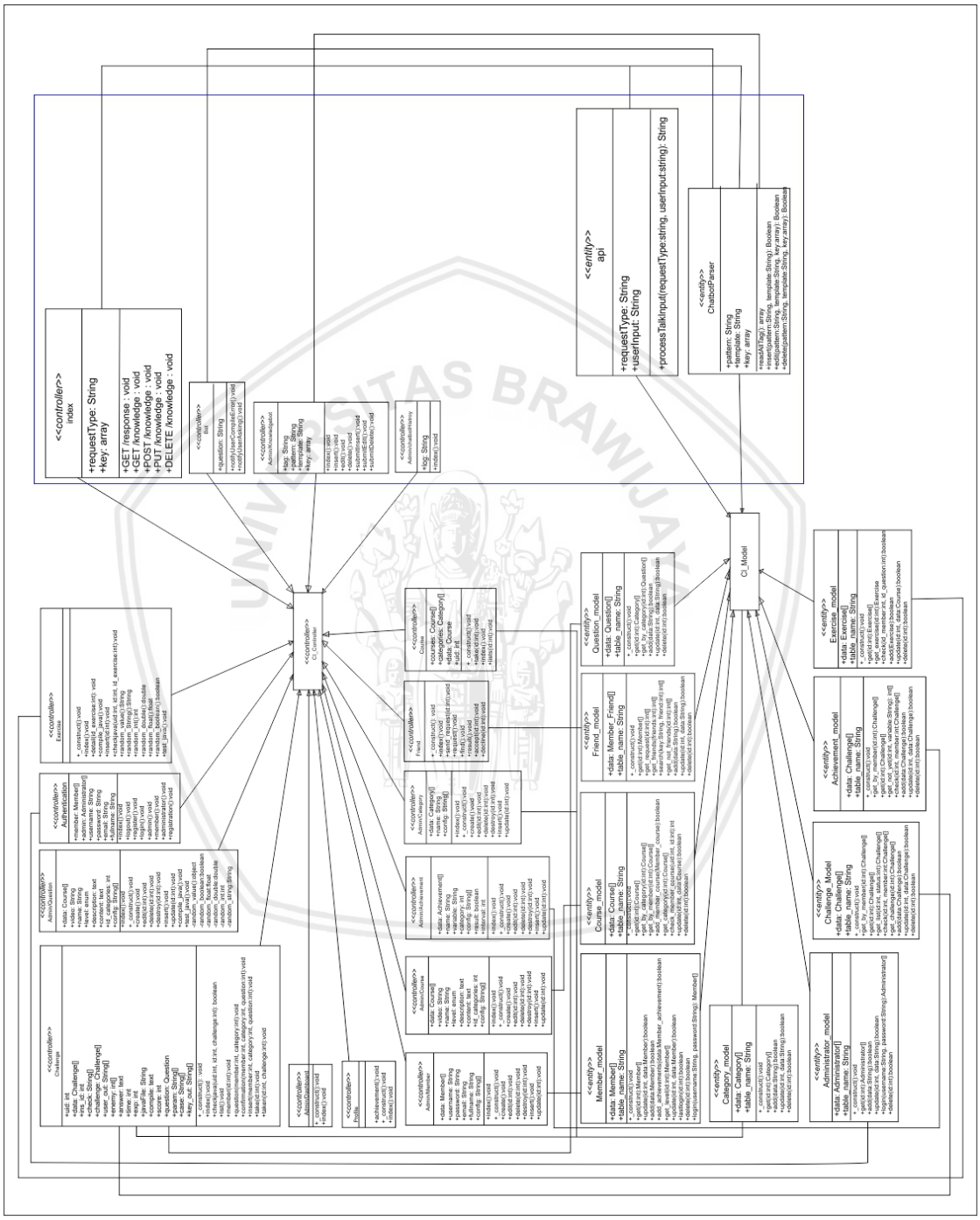


Gambar 5.7 Sequence diagram menambah pengetahuan berdasarkan konten *course* pada *CodeManiac*.

Gambar 5.7 menjelaskan ketika admin mengubah pengetahuan *chatbot*, maka sistem akan menjalankan fungsi submitEdit dan mengubah pengetahuan kedalam sistem lalu akan menampilkan kembali halaman courselist.

5.1.3 Perancangan *Class* Diagram

Class diagram merupakan diagram yang digunakan untuk merepresentasikan struktur dari *class* yang diimplementasikan ke dalam sistem CodeManiac.



Gambar 5.8 *Class* diagram pengembangan sistem CodeManiac

Pada Gambar 5.8 penulis menambahkan 4 controller dan 2 Model pada sistem Codemaniac.

5.1.4 Perancangan Komponen

Perancangan komponen ialah penyederhanaan sub-sistem menjadi komponen detail. Perancangan komponen memaparkan atribut dan algoritme *method* dalam sebuah *cIass* yang sudah dimodelkan sebelumnya pada pemodelan *cIass* diagram. Pada perancangan komponen mengambil method processTalkInput dari controller "api" ,method *index* dari "Chatbothistory", dan method *POST Knowledge* dari controller "index".

5.1.4.1 Algoritme Method processTalkInput

Nama *Class* : api

Nama Method : processTalkInput()

Algoritme : (ALGO_CM_1_01)

Tabel 5.1 Perancangan Komponen Method Process Talk Input

```
1   public function processTalkInput()
1     $config = new Config();
1     define("LOG", $config->log);
1     header(LOG ? "Content-Type: text/plain; charset=utf-8" :
"Content-Type: application/json; charset=utf-8");
      error_reporting(LOG ? E_ALL : JSON_ERROR_NONE);
      $result = array(
1           'status' => 'error',
1           'type' => 'empty',
1           'message' => 'empty message ...',
1           'data' => 'empty',
1                 );
2     if          (!isset($_REQUEST['requestType'])          ||
!isset($_REQUEST['userInput'])) {
            $result['status'] = 'error';
3           $result['type'] = $_REQUEST['requestType'];
3           $result['message'] = 'requestType and userInput is
3   required ...';
            $result['data'] = 'empty';
3     } else {
4           $userId      =      isset($_REQUEST['userId'])      ?
$_REQUEST['userId'] : $_SERVER['REMOTE_ADDR'];
5           LOG && print "userId : " . $userId . "\n";
            $chatbot = new Chatbot($config, $userId);
5           if ($_REQUEST['requestType'] == 'talk') {
5             $res = $chatbot->talk($_REQUEST['userInput']);
6             $data = $chatbot->getData();
7             $result['status'] = 'success';
7             $result['type'] = 'talk';
```

```
7         $result['message'] = trim(preg_replace("/\s+/", " ", $res));
7         $result['data'] = $data;
7     } elseif ($_REQUEST['requestType'] == 'forget') {
7         $chatbot->forget();
8         $result['status'] = 'success';
9         $result['type'] = 'forget';
          $result['message'] = 'forgetting completed ...';
9         $result['data'] = 'empty';
9     } else {
9         $result['status'] = 'error';
10        $result['type'] = $_REQUEST['requestType'];
11        $result['message'] = 'invalid request type ...';
11        $result['data'] = 'empty';
11     }
11 }
11     if (LOG) {
11        print "\n";
12        print_r($result);
13        print "\n";
13     } else {
13        echo json_encode($result);
14            }
16    }
17
```

5.1.4.2 Algoritme Method Index

Nama *Class* : *Chatbot*history

Nama Method : index()

Algoritme : (ALGO_CM_2_1)

Tabel 5.2 Perancangan Komponen Method Index

1	Begin index()
2	DB get *chatbot*master database
3	Select all column and rom from the LOG table from database
4	Result data.log as new array
5	Load view admin/header
6	Load view admin/*chatbot*history with data
7	Load view admin/footer

5.1.4.3 Algoritme Method POST knowledge

Nama *Class* : index

Nama Method : POST knowledge()

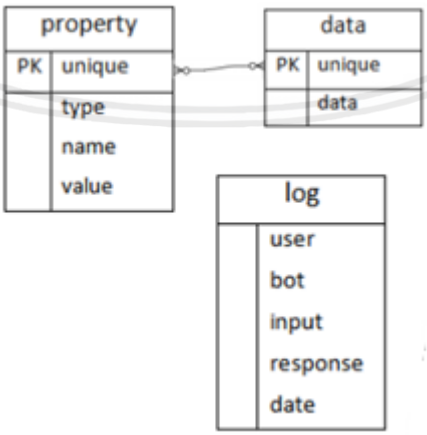
Algoritme : (ALGO_CM_2_02)

Tabel 5.3 Perancangan Komponen Method POST Knowledge

1	Begin GET response
2	Body runs as validate server input function
3	IF body http response code 400 echo 400 bad request
4	Return
5	Require chatbotparser.php
6	Parser as new chatbotparser (aiml file)
7	Parser insert pattern and template in body
8	Read all tag as result
9	Header content type
10	Echo json encode as result

5.1.5 Perancangan Basis Data

Dalam perancangan basis data ini akan ditunjukkan bagaimana data direkam dalam sistem berdasarkan diagram kelas yang telah dimodelkan sebelumnya, maka didapatkan rancangan basis data yang diGambarkan dalam bentuk *entity relational diagram* seperti yang ditunjukkan pada Gambar 5.9.



Gambar 5.9 *Entity Relational Diagram*

Entity Relational Diagram (ERD) diatas bahwa tabel data berisikan unique berhubungan many to many dengan unique di tabel property, dan pada tabel log

berisikan interaksi yang dilakukan pengguna terhadap chatbot beserta waktu saat interaksi tersebut.

5.1.6 Perancangan Pengetahuan *Chatbot*

Dalam perancangan pengetahuan *Chatbot* dibentuk sebuah pola pertanyaan berdasarkan topik. Tabel 5.4 merupakan contoh pada salah satu topik dalam pengetahuan *Chatbot.*

Tabel 5.4 Pattern Pola Pertanyaan Pengetahuan *Chatbot*

No	Topik Pertanyaan	Pola Pertanyaan	Pattern
1	Apa itu println	a. Apa itu println b. Apa println itu c. println adalah d. println	a. apa itu println b. * println * c. println * d. println

Pada tabel 5.4 terdapat *wildcard* (*) dimana *wildcard* disini bukan sebuah variabel acuan namun diartikan sebagai inputan bebas. Pada tabel 5.5 merupakan contoh *Pattern* dan *Template* pada suatu topik.

Tabel 5.5 Pattern Template Pengetahuan Chabot

No	Topik Pertanyaan	Pattern	Template
1	Apa itu println	a. apa itu println b. * println * c. println * d. println	a. println akan mencetak dengan adanya enter atau penambahan baris pada kalimat berikutnya b. <srai>apa itu println </srai> c. <srai>apa itu println </srai> d. <srai>apa itu println </srai>

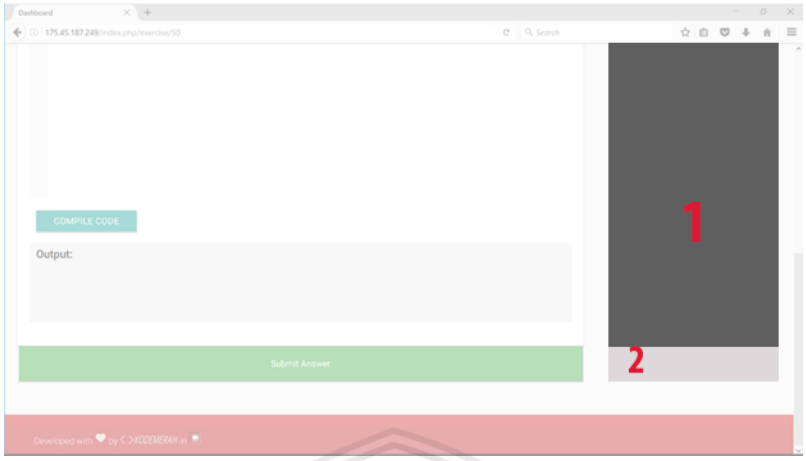
Pada tabel 5.5 terdapat *srai* dimana tag ini berarti ketika pengguna memasukkan pertanyaan println dan akan direspon sama seperti saat pengguna memasukkan pertanyaan apa itu println.

5.1.7 Perancangan Antarmuka

Dalam perancangan antarmuka akan ditunjukkan sejumlah antarmuka seperti antarmuka *chatbox* sesuai dengan kebutuhan dari pengguna yang akan menggunakan aplikasi CodeManiac ini.

5.1.7.1 Perancangan Antarmuka *chatbox*

Perancangan antarmuka *chatbox* pada Gambar 5.10 dan Penjelasan Detail dari perancangan antarmuka tersebut dijelaskan pada Tabel 5.3.

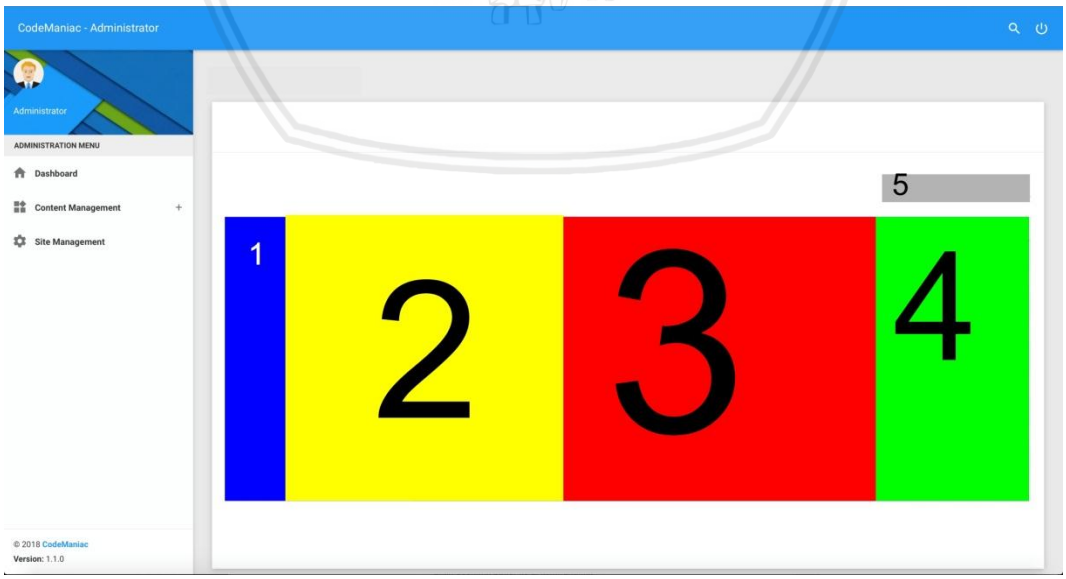


Gambar 5.10 Perancangan antarmuka chatbox

Tabel 5.6 Penjelasan Antarmuka chatbox

No	Nama Objek	Tipe	Keterangan
1	Riwayat percakapan pengguna	Text	Kolom untuk menampilkan riwayat pertanyaan pengguna, jawaban dari *chatbot*, serta peringatansaat pengguna melakukan kesalahan pada *live coding*.
2	Kolom input pertanyaan	Text	Kolom untuk menginputkan pertanyaan terhadap *chatbot*

5.1.7.2 Perancangan Antarmuka *Knowledgebothistory*

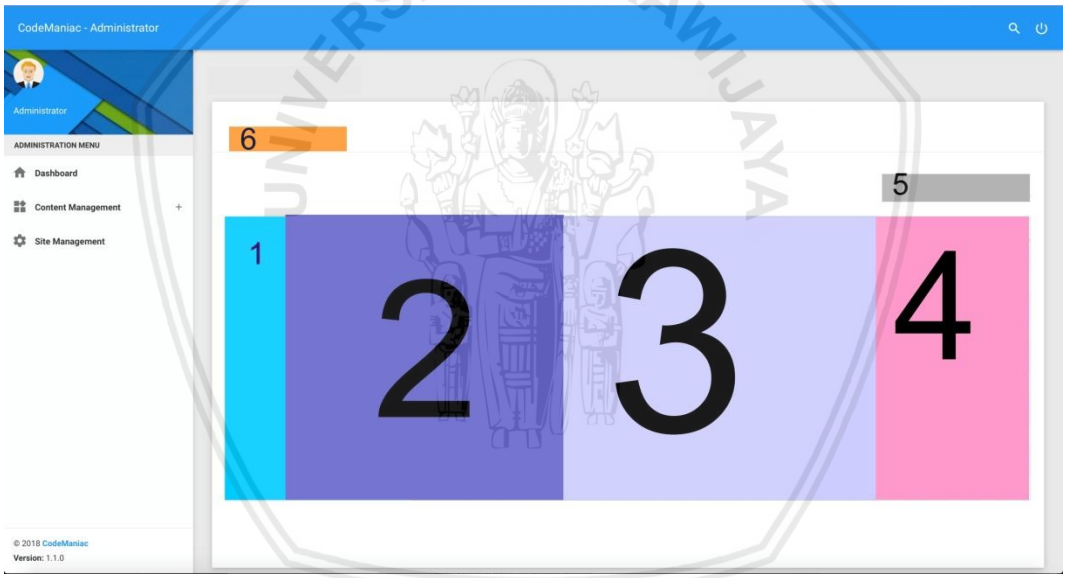


Gambar 5.11 Perancangan Antarmuka Knowledgebothistory

Tabel 5.7 Penjelasan Antarmuka Knowledgebothistory

No	Nama Objek	Tipe	Keterangan
1	Kolom nomor	text	Kolom untuk menampilkan nomor urut riwayat percakapan
2	Kolom Inputan pengguna	Text	Kolom untuk menampilkan riwayat pertanyaan pengguna,
3	Kolom response	Text	Kolom untuk menampilkan respon dari *chatbot*.
4	Kolom tanggal	Text	Kolom untuk menampilkan riwayat tanggal percakapan terjadi
5	Kolom search	Text	Kolom untuk mencari riwayat percakapan

5.1.7.3 Perancangan Antarmuka *Knowledge*



Gambar 5.12 Perancangan antarmuka Knowledge

Tabel 5.8 Antarmuka Knowledgebot

No	Nama Objek	Tipe	Keterangan
1	Kolom nomor	text	Kolom untuk menampilkan nomor urut riwayat percakapan
2	Kolom Inputan pengguna	Text	Kolom untuk menampilkan riwayat pertanyaan pengguna,
3	Kolom response	Text	Kolom untuk menampilkan respon dari *chatbot*.

4	Tombol Edit dan Delete	button	Kolom untuk mengedit atau menghapus knowledgebot
5	Kolom search	Text	Kolom untuk mencari riwayat percakapan
6	Tombol knowledge baru	button	Tombol untuk menambah knowledgebot

5.2 Implementasi Sistem

Tahapan implementasi yang dilakukan terdiri dari penjelasan tentang spesifikasi lingkungan pengembangan sistem, batasan implementasi, implementasi algoritme, implementasi basis data dan implementasi antarmuka pengguna.

5.2.1 Spesifikasi Sistem

Spesifikasi sistem merupakan lingkungan implementasi dalam pengembangan sistem yang terdiri dari perangkat keras dan perangkat lunak.

5.2.1.1 Spesifikasi Perangkat Keras

Spesifikasi perangkat keras yang digunakan penulis dalam pengembangan lanjut sistem CodeManiac dapat dilihat pada tabel 5.9.

Tabel 5.9 Spesifikasi Perangkat Keras

Nama Komponen	Spesifikasi
Laptop	1. MacBook Pro Retina Display (13inch, 2015) 2. Processor 2.9 G.Hz Intel Core i5 3. Memory 8 GB 1867 MHz DDR3 4. Graphic Intel Iris Graphics 6100 1536 MB

5.2.1.2 Spesifikasi Perangkat Lunak

Spesifikasi perangkat lunak yang digunakan dalam pengembangan lanjut sistem CodeManiac dapat dilihat pada tabel 5.10.

Tabel 5.10 Spesifikasi Perangkat Lunak

Nama Komponen	Spesifikasi
*Edit*or Dokumentasi	Microsoft Word 2015

*Edit*or Perancangan	Visual Paradigm
*Edit*or Pemrograman	Visual Studio Code – 3
Bahasa Pemrograman	PHP
Framework	CodeIgniter
Web Service	Xampp v3.2.1
Database	MySQL
Browser	Safari

5.2.1.3 Spesifikasi Sistem Operasi

Tabel 5.11 Penjelasan Spesifikasi Sistem Operasi

Nama Komponen	Spesifikasi
Sistem Operasi	macOS High Sierra 10.12.6

5.2.2 Implementasi Kelas

Setiap kelas pada tahap perancangan sistem diimplementasikan pada sebuah file sistem dengan ekstensi *.php. Penjelasan terkait dengan nama kelas dan file sistem yang ditambahkan pada sistem akan dijelaskan pada tabel 5.12.

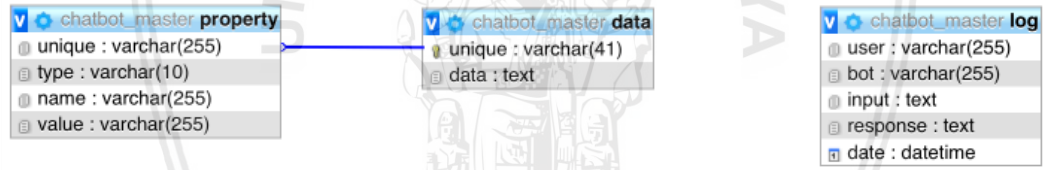
Tabel 5.12 Implementasi Kelas

No	Package	Nama Kelas	Nama File Sistem
1	Controller	Course	Course.php
2	Controller	Chatbothistory	Admin/*chatbot*history.php
3	Controller	Bot	Chatbot-master/bot.php
4	Controller	Knowledgebot	Admin/knowledgebot.php
5	Model	ChatbotParser	ChatbotParser.php
6	Model	Knowledgebot_Model	Knowledgebot-model.php
7	Model	Api	Chatbot-master/api.php
8	Model	ChatbotParser	Chatbot-master/chatbotParser.php
9	Model	Course_Model	Course_model.php
10	Model	Member_model	Member_model.php
11	Model	Challenge_model	Challenge_model.php
12	Model	ChatbotCategory	ChatbotCategory.php

13	View	File HTML	Admin/knowledgebot-insert.php
14	View	File HTML	Admin/knowledgebot-delete.php
15	View	File HTML	Admin/knowledgebot-edit.php
16	View	File HTML	Member-edit.php
17	View	File HTML	Admin/header.php
18	View	File HTML	Admin/chatbothistory.php
19	VIew	File HTML	Admin/knowledgebot.php
20	View	File HTML	Course-list.php
21	View	File HTML	Exercise-view.php
22	View	dashboard	dashboard.php

5.2.3 Implementasi Basis Data

Implementasi data dilakukan berdasarkan analisis data yang dilakukan pada analisis data sebelumnya yang telah dilakukan sesuai dengan kebutuhan sistem.



Gambar 5.13 Implementasi Basis Data

Gambar 5.13 menunjukkan hasil implementasi dari rancangan basis data.

5.2.4 Implementasi Kode Program

Berikut adalah hasil implementasi algoritme dari *class* controller index, dan *chatbot*history dalam bahasa pemrograman php yang mengimplementasikan framework CodeIgniter

5.2.4.1 Implementasi ALG_CM_1_1

Tabel 5.13 Implementasi Komponen Method Process Talk Input

```
1  $router->map('GET', '/response', function(){
2  Require './api.php';
3  $requestType;'';
4  if(isset($_GET['request_type'])){}}
5  $userInput;'';
6  if(isset($_GET['user_input'])){
7  $userInput= $_GET['user_input'];}
```

```
8  Header('Content-Type; application/json');
9  processTalkInput($requestType, $userInput);});
```

5.2.4.2 Implementasi ALG_CM_2_2

Tabel 5.14 Implementasi Komponen Method Index

```
1  $router->map('POST', '/knowledge', function(){
2    $body = validateServerInput();
3    If (!body){http_response_code(400);
4        Echo '400 Bad Request';
5        Return;}
6  Require './ChatbotParse.php';
7  $parser = new ChatbotParser('./aiml/chabot.aiml');
8  $parser->insert($body->pattern, $body->template);
9  $result = $parser->readAllTag();
10 Header('Content-Type: application/json');
11 Echo json_encode($result);
12 });
```

5.2.4.3 Implementasi ALG_CM_2_1

Tabel 5.15 Implementasi Komponen Method POST Knowledge

```
1  public function index()
2  { $db2 = $this->load->database("chatbotmaster",True);
3     $query = $db2->query("select * from log ");
4     $data['log'] = $query->result_array();
5     $this->load->view('admin/header');
6     $this->load->view('admin/Chatbothistory',$data);
7     $this->load->view('admin/footer');}
```

5.2.5 Implementasi Pengetahuan *Chatbot*

Implementasi pengetahuan *Chatbot* dilakukan berdasarkan analisis data yang dilakukan berdasarkan perancangan sebelumnya dan diterapkan kedalam *AIML.* Berikut adalah struktur data dari salah satu topik pada pengetahuan chatbot yang telah diimplementasi ke dalam AIML.

Tabel 5.16 Implementasi Pengetahuan Chatbot

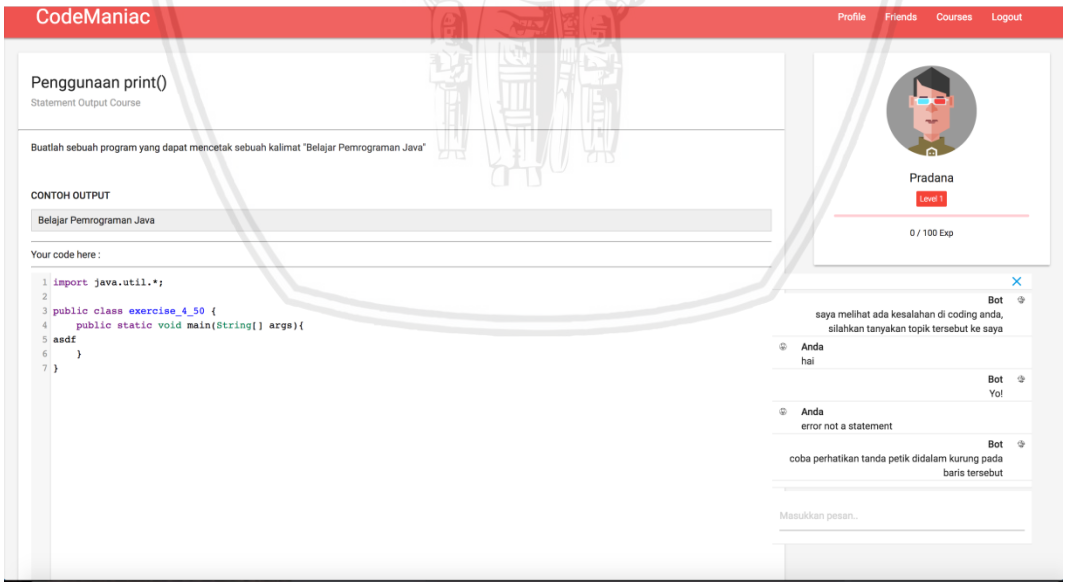
```
1  <category>
2  <pattern>println</pattern>
3  <template> println akan mencetak dengan adanya enter atau penambahan
4  baris pada kalimat berikutnya </template>
   </category>
```

```
5   <category>
6   <pattern>* println</pattern>
7   <template><srai>println</srai></template>
8   </category>
9   <category>
10  <pattern>println *</pattern>
11  <template><srai>println</srai></template>
12  </category>
13  <category>
14  <pattern>* println *</pattern>
15  <template><srai>println</srai></template>
16  </category>
```

5.2.6 Implementasi Antarmuka

Implementasi antarmuka dilakukan berdasarkan analisis data yang dilakukan berdasarkan perancangan sebelumnya dan diterapkan ke dalam bahasa PHP sebagai antarmuka yang mengimplementasikan *Framework CodeIgniter*. Berikut adalah hasil implementasi antarmuka pengguna.

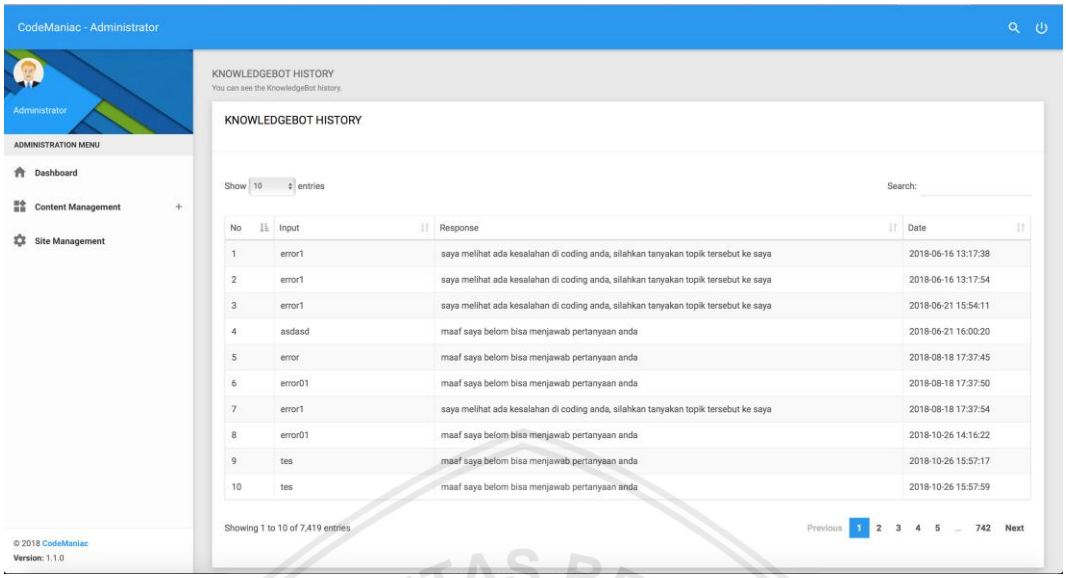
5.2.6.1 Implementasi Antarmuka Chatbox



Gambar 5.14 Implementasi Antarmuka *Chatbox*

Gambar 5.14 menunjukkan hasil implementasi dari rancangan antarmuka chatbox pada halaman mengerjakan materi *course* 1. Halaman ini ditampilkan penjelasan materi, pertanyaan sesuai course yang diambil, *form source code* dan *chatbox* untuk berinteraksi dengan *bot*.

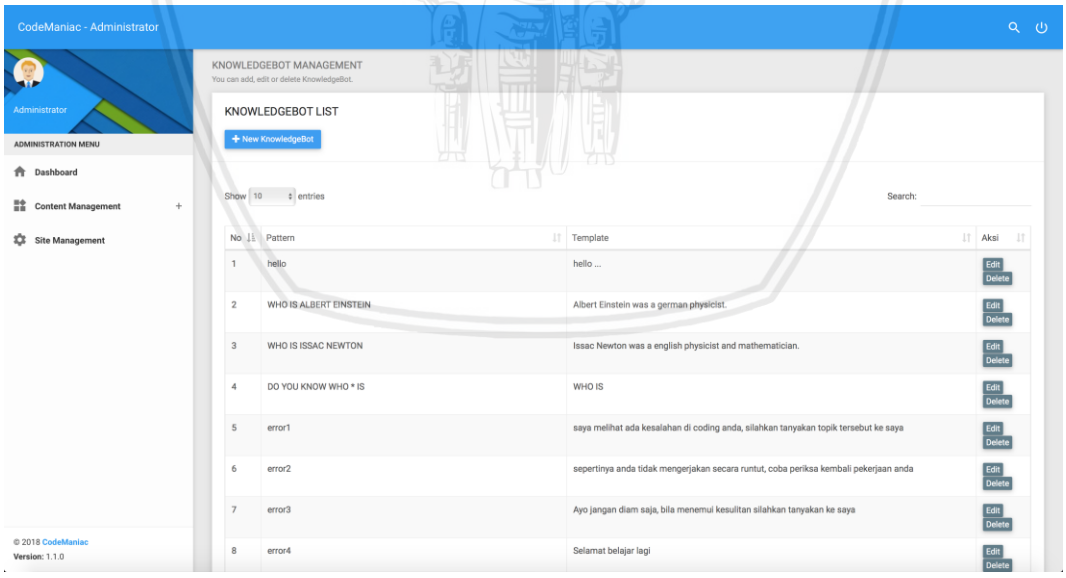
5.2.6.2 Implementasi antarmuka Knowledgebot History



Gambar 5.15 Implementasi Antarmuka Knowledgebot History

Gambar 5.15 menunjukkan hasil implementasi dari rancangan antarmuka Knowledgebothistory. Halaman ini menjelaskan tentang pertanyaan yang diinputkan pengguna terhadap *chatbot* berserta waktu saat pengguna menanyakan pertanyaan tersebut.

5.2.6.3 Implementasi antarmuka Knowledgebot



Gambar 5.16 Implementasi Antarmuka Knowledgebot

Gambar 5.16 menunjukkan hasil implementasi dari rancangan antarmuka Knowledgebot. Halaman ini berisikan pengetahuan *chatbot* dimana admin bisa menambah, mengedit dan menghapus pengetahuan *chatbot*.

BAB 6 PENGUJIAN SISTEM

6.1 Pengujian Unit

Proses pengujian *unit* menggunakan suatu pengujian *whitebox* berdasarkan *basis path* dimana identifikasi metode ini berdasarkan jalur atau struktur yang ada dari sistem. Menggunakan *pseudocode* yang digunakan oleh sistem dalam pengembangan *Chatbot*, *pseudocode* tersebut nantinya dimodelkan ke dalam suatu *flow graph* yang menentukan berapa jumlah *cyclomatic complexity* dan jumlah jalur independen. Jumlah kompleksitas siklomatis diperoleh dengan tiga persamaan berikut:

V(G) = E-N+2 (6.1)

V(G) = P+1 (6.2)

V(G) = R (6.3)

Dimana,

V(G) : Jumlah kompleksitas siklomatis

E : Sisi atau edge (garis penghubung antar node)

N : Jumlah simpul (node)

P : Predicate node pada grafik alir

R : Jumlah region pada flow graph.

Tujuan dilakukannya pengujian unit adalah untuk memastikan algoritme yang memiliki sudah diimplementasikan sesuai dengan dirancang sebelumnya.

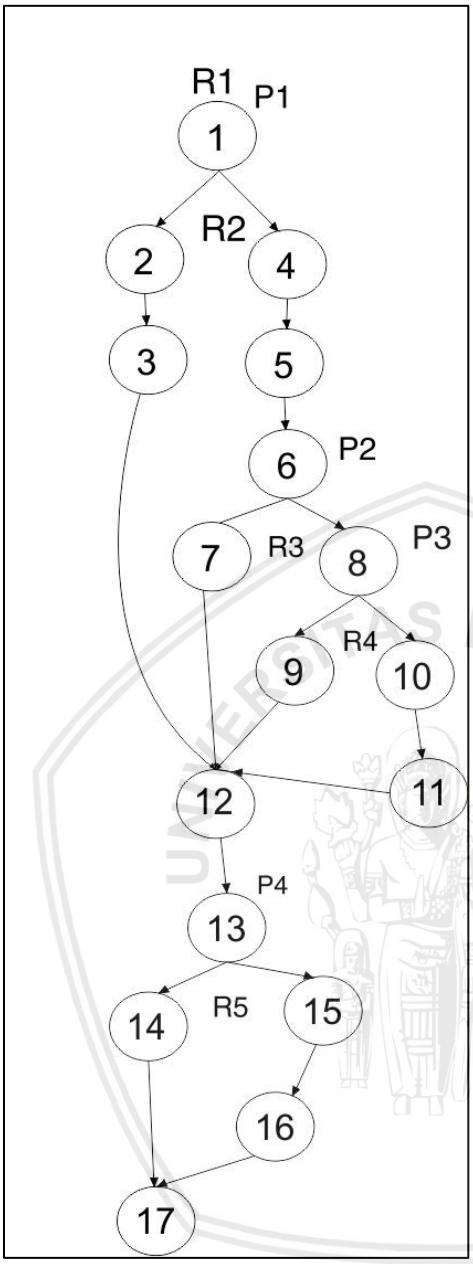
6.1.1 Pengujian Unit Klas api algoritme processTalkInput

1. Pseudocode

```
1   function processTalkInput(RequestType, sserInput)
1     $config = new Config();
1     define("LOG", $config->log);
1     header(LOG ? "Content-Type: text/plain; charset=utf-8" : "Content-
Type: application/json; charset=utf-8");
      error_reporting(LOG ? E_ALL : JSON_ERROR_NONE);
      $result = array(
1          'status' => 'error',
1          'type' => 'empty',
1          'message' => 'empty message ...',
1          'data' => 'empty',
1                );
2     if              (!isset($_REQUEST['requestType'])              ||
!isset($_REQUEST['userInput'])) {
          $result['status'] = 'error';
```

```
3       $result['type'] = $_REQUEST['requestType'];
3       $result['message'] = 'requestType and userInput is required
...';
3       $result['data'] = 'empty';
    } else {
3       $userId = isset($_REQUEST['userId']) ? $_REQUEST['userId'] :
4   $_SERVER['REMOTE_ADDR'];
5       LOG && print "userId : " . $userId . "\n";
        $chatbot = new Chatbot($config, $userId);
5       if ($_REQUEST['requestType'] == 'talk') {
5         $res = $chatbot->talk($_REQUEST['userInput']);
6         $data = $chatbot->getData();
7         $result['status'] = 'success';
7         $result['type'] = 'talk';
7         $result['message']   =   trim(preg_replace("/\s+/",   "   ",
7   $res));
7         $result['data'] = $data;
      } elseif ($_REQUEST['requestType'] == 'forget') {
7         $chatbot->forget();
8         $result['status'] = 'success';
9         $result['type'] = 'forget';
          $result['message'] = 'forgetting completed ...';
9         $result['data'] = 'empty';
9       } else {
9         $result['status'] = 'error';
10        $result['type'] = $_REQUEST['requestType'];
11        $result['message'] = 'invalid request type ...';
11        $result['data'] = 'empty';
11      }
11  }
11      if (LOG) {
11         print "\n";
12         print_r($result);
13         print "\n";
13      } else {
13         echo json_encode($result);
14            }
16    }
17
```

2. *Basis Path Testing*
   a. *Flow Graph*



Gambar 6.1 Flow graph algoritme processTalkInput klas api

b. Cyclomatic Complexity

- V(G) = 5, ada 5 *region* R1, R2, R3, R4, R5
- V(G) = 20 *edges* – 17 *nodes* + 2 = 5
- V(G) = 4 *predicate nodes* + 1 = 5

c. Independent Path

- Jalur 1 = 1 – 2 – 3 – 4 – 12 – 13 – 14 – 17
- Jalur 2 = 1 – 2 – 3 – 4 – 12 – 13 – 15 – 16 – 17
- Jalur 3 = 1 – 4 – 5 – 6 – 7 – 12 – 13 – 14 – 17

- Jalur 4 = 1 – 4 – 5 – 6 – 8 – 9 – 12 – 13 – 15 – 16 – 17
- Jalur 5 = 1 – 4 – 5 – 6 – 8 – 10 – 11 – 12 – 13 – 15 – 16 – 17

Test case dan hasil akan dijelaskan pada tabel 6.1 dibawah ini.

Tabel 6.1 Hasil pengujian unit klas Api algoritme processTalkInput()

Jalur	Prosedur Uji	Expected Result	Result	Status
1.	Menjalankan algoritme Index namun tidak mengisi variabel requestType dan userInput	json_encode berisi array {“status”:”error”,”type”:”empty”,”message”:”RequestType and userInput is required message”,”data”:”empty”}	json_encode berisi array {“status”:”error”,”type”:”empty”,”message”:”RequestType and userInput is required message”,”data”:”empty”}	Valid
2.	Menjalankan algoritme Index dengan variabel Request.requestType	json_encode berisi array {“status”:”error”,”type”:”empty”,”message”:”RequestType and userInput is required message”,”data”:”empty”}	json_encode berisi array {“status”:”error”,”type”:”empty”,”message”:”RequestType and userInput is required message”,”data”:”empty”}	Valid
3.	Menjalankan algoritme Index dengan variabel requestType =talk telah mengisikan uid sebagai unique id/user id,	json_encode berisi array {“status”:”success”,”type”:”talk”,”message”:”data”,”data”}	son_encode berisi array {“status”:”success”,”type”:”talk”,”message”:”data”,”data” }	Valid
4.	Menjalankan algoritme Index dengan variabel requestType =forget telah mengisikan uid sebagai unique id/user id,	json_encode berisi array {“status”:”success”,”type”:”forget”,”message”:”forgetting completed”,”data”:”empty”}	json_encode berisi array {“status”:”success”,”type”:”forget”,”message”:”forgetting completed”,”data”:”empty”}	Valid
5.	Menjalankan	json_encode berisi array	json_encode berisi array	Valid

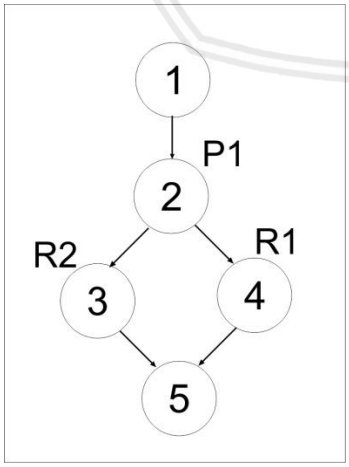
	algoritme Index dengan variabel requestType =sembarang telah mengisikan uid sebagai unique id/user id,	{“status”:”error”,”type”:” requestTpe”,”message”:” invalid request type”,”data”:”empty”}	{“status”:”error”,”type”:”r equestTpe”,”message”:”in valid request type”,”data”:”empty”}	

6.1.2 Pengujian Unit Klas index algoritme POST knowledge

1. *Pseudocode*

1	Begin GET response
2	Body runs validate server input function
3	IF body http response code 400 echo 400 bad request
3	Return
4	Require chatbotparser.php
4	Parser as new chatbotparser (aiml file)
4	Parser insert pattern and template in body
	Read all tag as result
5	Header content type
5	Echo json encode as result

2. *Basis Path Testing*
   a. *Flow Graph*



Gambar 6.2 Flow graph algoritme POST knowledge klas index

b. Cyclomatic Complexity

- V(G) = 2, ada 2 *region* R1, R2
- V(G) = 5 *edges* – 5 *nodes* + 2 = 2
- V(G) = 1 *predicate nodes* + 1 = 2

c. Independent Path

- Jalur 1 = 1 – 2 – 3 – 5
- Jalur 2 = 1 – 2 – 4 – 5

Test case dan hasil akan dijelaskan pada tabel 6.2 dibawah ini.

Tabel 6.2 Hasil pengujian unit klas Course algoritme take()

Jalur	Prosedur Uji	Expected Result	Result	Status
1.	Menjalankan algoritme POST knowledge dengan variable body	Error 400 bad request	Error 400 bad request	Valid
2.	Menjalankan algoritme POST knowledge dengan variable body	Menambah pengetahuan chatbot	Menambah pengetahuan chatbot	Valid

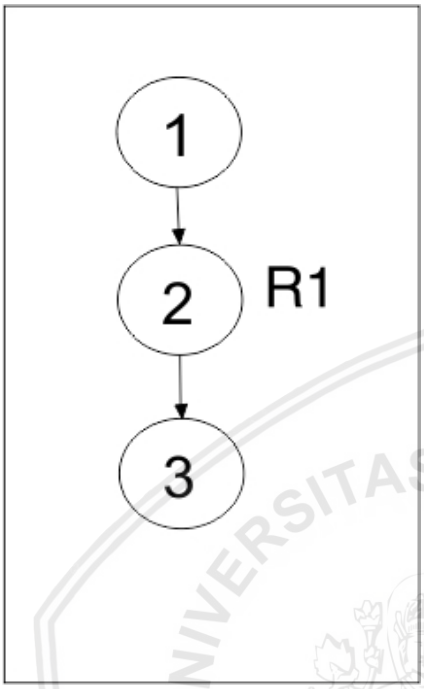
6.1.3 Pengujian Unit Klas *Chatbot*history algoritme index

1. Pseudocode

```
1   Begin index()
2   DB get chatbotmaster database
2   Select all column and rom from the LOG table from database
2   Result data.log as new array
3   Load view admin/header
3   Load view admin/chatbothistory with data
3   Load view admin/footer
```

2. *Basis Path Testing*

a. *Flow Graph*



Gambar 6.3 Flow graph algoritme index klas Chatbothistory

b. *Cyclomatic Complexity*

- V(G) = 1, hanya ada 1 *region* R1
- V(G) = 2 *edges* – 3 *nodes* + 2 = 1
- V(G) = 0 *predicate nodes* + 1 = 1

c. *Independent Path*

- Jalur 1 = 1 – 2 – 3 – 8

Test case dan hasil akan dijelaskan pada tabel 6.3 dibawah ini.

Tabel 6.3 Hasil pengujian unit klas Knowledgebothistory algoritme index()

Jalur	Prosedur Uji	Expected Result	Result	Status
1.	Menjalankan algoritme index	Menampilkan data log	Menampilkan data log	Valid

6.2 Pengujian Integrasi

Pengujian integrasi dilakukan untuk menguji integrasi antar unit-unit yang ada, sehingga dapat berjalan dengan sesuai. Pada Tabel 6.4 ditunjukkan identifikasi dan rencana uji yang dilakukan. Hasil dari pengujian integrasi ditunjukkan pada Tabel 6.5 – Tabel 6.7

Tabel 6.4 Identifikasi dan Perancangan Pengujian Integrasi

No	Nama kelas	Nama Method	Tujuan
1	Knowledgebot	Insert(pattern, template)	Menginputkan pengetahuan baru pada *Chatbot*
	KnowledgeBot	SubmitInsert()	
	ChatbotParser	Insert(pattern, template)	
	index	POST /knowledge(pattern, template)	
	ChatbotParser	Insert(pattern, template)	
2	index	GET /response(requestType, userInput)	Menanyakan suatu topik pertanyaan
	api	processTalkInput(requestType, userInput)	
3	Knowledgebot	Delete(key)	Menghapus pengetahuan *Chatbot*
	Knowledgebot	SubmitDelete()	
	ChatbotParser	Delete(pattern, template, key)	
	index	DELETE /knowledge/{i:key}(pattern, template,key)	
	ChatbotParser	Delete(pattern, template, key)	

Tabel 6.5 Hasil pengujian integrasi No. 1

No Uji	1
Input Pertama	“pattern”: “selamat kamu berhasil” “template”: “terima kasih”
Method dari kelas knowledgebot	Insert(pattern, template)
Output Pertama / Input Kedua	“pattern”:“ selamat kamu berhasil” , “template”:“terima kasih”

Method dari kelas ChatbotParser	Insert(pattern, template)
Output Kedua/ *Input* Ketiga	“header”:’Content-Type: application/json’ “body”:” “pattern”:“ selamat kamu berhasil” , “template”:“terima kasih” ”
Method dari kelas Index	POST knowledge(pattern, template)
Output Kedua/ *Input* Ketiga	“body”:” “pattern”:“ selamat kamu berhasil” , “template”:“terima kasih” ”
Method dari kelas ChatbotParser	Insert(pattern, template)
Expected Result	Menambah pengetahuan pada *aiml*
Result	Menambah pengetahuan pada *aiml*
Status	Valid

Tabel 6.6 Hasil pengujian integrasi No. 2

No Uji	2
Input Pertama	“requestType”:”talk” “userInput”:”println”
Method dari kelas index	GET /response(requestType, userInput)
Output Pertama / Input Kedua	“requestType”:”talk” “userInput”:”println”
Method dari kelas api	processTalkInput(requestType, userInput)
Expected Result	Menampilkan respon berdasarkan masukan
Result	Menampilkan respon berdasarkan masukan
Status	Valid

Tabel 6.7 Hasil pengujian integrasi No.3

No Uji	3
Input Pertama	“key”:137
Method dari kelas knowledgebot	submitDelete()
Output Pertama / Input Kedua	“pattern”:“ selamat kamu berhasil” , “template”:“terima kasih”, “key”:”137”
Method dari kelas ChatbotParser	delete(pattern, template, key)
Output Kedua/ *Input* Ketiga	“header”:’Content-Type: application/json’ “body”:” “pattern”:“ selamat kamu berhasil” , “template”:“terima kasih” ”, “key”:”137”
Method dari kelas Index	DELETE knowledge/[i:key](key)
Output Ketiga/ *Input* Keempat	“body”:” “pattern”:“ selamat kamu berhasil” , “template”:“terima kasih” ”, “key”:”137”
Method dari kelas ChatbotParser	delete(pattern, template,key)
Expected Result	Menghapus pengetahuan pada *aiml* berdasarkan pilihan pengetahuan yang telah dipilih
Result	Menghapus pengetahuan pada *aiml* berdasarkan pilihan pengetahuan yang telah dipilih
Status	Valid

6.3 Pengujian Validasi

Pengujian validasi dilaksanakan untuk menegaskan bahwa semua kebutuhan fungsional dari sistem berjalan sepadan dengan yang dimaksud dengan metode blackbox. Selanjutnya adalah mengenai pengujian validasi berdasarkan skenario *usecase* yang telah dipaparkan pada perancangan sistem.

6.3.1 Pengujian Validasi Bertanya pada *Chatbot*

a. Kasus Uji bertanya pada *chatbot*

Tabel 6.8 Kasus Uji Bertanya pada *chatbot*

Nama Kasus Uji	bertanya pada *chatbot*
Prosedur	1. Masuk ke dalam sistem 2. Menekan Gambar *chatbot* 3. Menginputkan sebuah pertanyaan 4. Menekan enter
Hasil yang diharapkan	Sistem CodeManiac akan menampilkan jawaban berdasarkan pengetahuan yang ada
Hasil	Sistem CodeManiac menampilkan jawaban berdasarkan pengetahuan yang ada
Status	Valid

6.3.2 Pengujian Validasi Peringatan error pada *live coding*

b. Kasus Uji Peringatan error pada *live coding*

Tabel 6.9 Kasus Uji Peringatan error pada *live coding*

Nama Kasus Uji	Kasus Uji Peringatan error pada *live coding*
Prosedur	1. Masuk ke dalam sistem 2. Memilih menu “Courses” 3. Memilih satu materi 4. Menekan tombol “Start Exercise” 5. Memasukan kode java pada form yang tersedia
Hasil yang diharapkan	Sistem CodeManiac akan menampilkan peringatan error pada saat *live coding*
Hasil	Sistem CodeManiac menampilkan peringatan error pada saat *live coding*
Status	Valid

6.3.3 Pengujian Validasi Peringatan saat pengambilan course secara tidak runtut

c. Kasus Uji Peringatan error pada saat pengambilan course secara tidak runtut

Tabel 6.10 Kasus Uji pengambilan course

Nama Kasus Uji	Kasus Uji pengambilan course
Prosedur	1. Masuk ke dalam sistem

	2. Memilih menu “Courses” 3. Memilih satu materi
Hasil yang diharapkan	Sistem CodeManiac akan menampilkan peringatan error pada saat pengambilan course yang tidak runtut
Hasil	Sistem CodeManiac menampilkan peringatan error pada saat pengambilan course yang tidak runtut
Status	Valid

6.3.4 Pengujian Validasi Melihat Log Percakapan Pengguna dengan *chatbot*

d. Kasus Uji melihat log percakapan pengguna dengan *chatbot*

Tabel 6.11 Kasus uji melihat log percakapan pengguna dengan *chatbot*

Nama Kasus Uji	Kasus uji melihat log percakapan pengguna dengan *chatbot*
Prosedur	1. Admin telah masuk ke dalam sistem 2. Admin memilih menu “Content Management” 3. Admin memilih menu “*Chatbot*History”
Hasil yang diharapkan	Sistem CodeManiac akan menampilkan seluruh riwayat percakapan antar pengguna dengan *chatbot*
Hasil	Sistem CodeManiac menampilkan riwayat percakapan antar pengguna dengan *chatbot*
Status	valid

6.3.5 Pengujian Validasi Menambah pengetahuan *chatbot*

e. Kasus Uji menambah pengetahuan *chatbot*.

Tabel 6.12 Kasus Uji menambah pengetahuan *chatbot*

Nama Kasus Uji	Kasus uji menambah pengetahuan *chatbot*.
Prosedur	1. Admin memilih menu “Content Management” 2. Admin memilih menu “Knowledgebot” 3. Admin menekan tombol “New Knowledgebot” 4. Admin mengisi form knowledgebot list 5. Admin menekan tombol save
Hasil yang diharapkan	Data knowledgebot akan bertambah di dalam sistem

Hasil	Data knowledgebot bertambah di dalam sistem
Status	valid

6.3.6 Pengujian Validasi Mengedit pengetahuan *chatbot*

f. Kasus Uji mengedit pengetahuan *chatbot*

Tabel 6.13 Kasus Uji mengedit pengetahuan *chatbot*

Nama Kasus Uji	Kasus uji mengedit pengetahuan *chatbot*.
Prosedur	1. Admin memilih menu "Content Management" 2. Admin memilih menu "Knowledgebot" 3. Admin memilih salah satu pengetahuan bot lalu menekan tombol edit 4. Admin mengedit form knowledgebot list 5. Admin menekan tombol save
Hasil yang diharapkan	Data knowledgebot terbaru akan tersimpan di dalam sistem
Hasil	Data knowledgebot terbaru tersimpan di dalam sistem
Status	Valid

6.3.7 Pengujian Validasi Menghapus pengetahuan *chatbot*

g. Kasus Uji menghapus pengetahuan *chatbot*

Tabel 6.14 Kasus Uji menghapus pengetahuan *chatbot*

Nama Kasus Uji	Kasus Uji menghapus pengetahuan *chatbot*
Prosedur	1. Admin memilih menu "Content Management" 2. Admin memilih menu "Knowledgebot" 3. Admin memilih salah satu pengetahuan bot lalu menekan tombol edit 4. Admin melihat form knowledgebot list

	5. Admin menekan tombol delete
Hasil yang diharapkan	Data knowledgebot yang dipilih admin akan terhapus
Hasil	Data knowledgebot yang dipilih admin terhapus
Status	Valid

6.3.8 Pengujian Validasi Menambah pengetahuan berdasarkan konten *course* yang telah ditambahkan.

h. Kasus Uji menambah pengetahuan berdasarkan konten *course* yang telah ditambahkan.

Tabel 6.15 Kasus Uji menambah pengetahuan berdasarkan konten *course* yang telah ditambahkan.

Nama Kasus Uji	Kasus Uji menghapus pengetahuan *chatbot*
Prosedur	1. Admin memilih menu "Content Management" 2. Admin memilih menu "Course" 3. Admin memilih salah satu course lalu menekan tombol generate knowledge
Hasil yang diharapkan	Data knowledgebot akan bertambah berdasarkan course yang telah dipilih admin
Hasil	Data knowledgebot bertambah berdasarkan course yang telah dipilih admin
Status	Valid

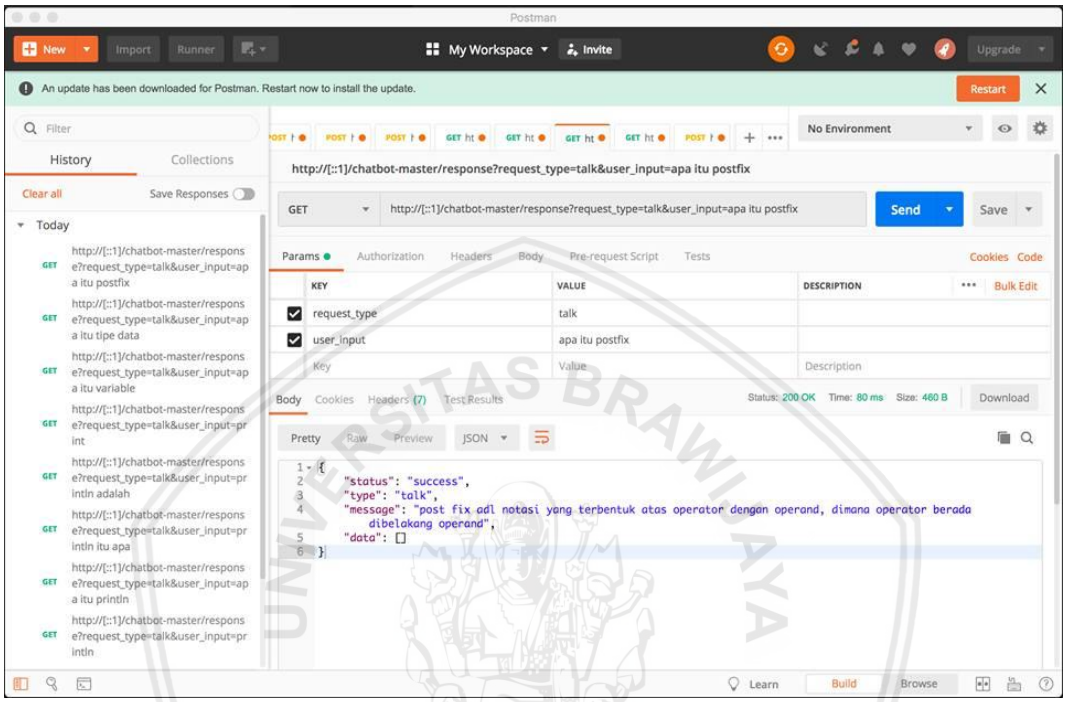
6.4 Pengujian Pengetahuan *Chatbot*

Pengujian Pengetahuan *Chatbot* dilakukan untuk menguji apakah respon pada pengetahuan Chatbot telah sesuai berdasarkan masukan dari pengguna. Pengujian pengetahuan *Chatbot* dapat dilihat pada Tabel 6.14 dan hasil pengujian pengetahuan *Chatbot* dapat dilihat pada Gambar 6.1. pengujian menggunakan aplikasi Postman.

Tabel 6.16 Pengujian Pengetahuan Chatbot

Prosedur	Menjalankan sistem dengan inputan tertentu
Hasil yang diharapkan	Sistem akan merespon sesuai dengan inputan yang diberikan berdasarkan pengetahuan yang ada

Hasil	Sistem merespon sesuai dengan inputan yang diberikan berdasarkan pengetahuan yang ada
Status	Valid



Gambar 6.4 Pengujian Pengetahuan Chatbot dengan POSTMAN

6.5 Pengujian Usabilitas

Pengujian usabilitas dilaksanakan untuk menguji sistem dengan cara menguji sistem secara langsung kepada sejumlah responden untuk mengetahui tingkat kemudahan penggunaan sebuah sistem. Setelah menguji sistem responden akan mengisikan sebuah kuisioner. Dalam pengujian kali ini usability diukur menggunakan pengujian post task study dengan metode Single Ease Question (SEQ) Tujuan dalam pengujian ini untuk mengukur kemudahan yang dirasakan pengguna setelah menyelesaikan semua skenario atau task yang diberikan (Brooke 2011). Penilaian hasil perhitungan terhadap kuesioner pada pengujian usability untuk mengukur pengembangan aplikasi Code Maniac dapat dilihat pada tabel 6.15.

Tabel 6.17 Parameter bobot nilai SEQ

Skor	Kualifikasi	Hasil
85-100%	Sangat Baik (SB)	Berhasil
65-84%	Baik (B)	Berhasil
55-64%	Cukup (C)	Tidak Berhasil

0-54%	Kurang (K)	Tidak Berhasil

Sumber : Arikunto, S. (2009)

6.5.1 Pengujian Usabilitas

Pelaksanaan pengujian usabilitas dilakukan oleh mahasiswa filkom universitas brawijaya. Responden diminta untuk menggunakan aplikasi Code Maniac, mengerjakan *task* dan diberikan angket SEQ dan angket berisikan pertanyaan pengembangan Code Maniac. Terlebih dahulu responden diberikan penjelasan bagaimana menggunakan aplikasi Code Maniac dan selanjutnya responden mencoba aplikasi dengan mengikuti task seperti pada tabel 6.16, setelah menyelesaikan skenario yang diberikan responden diminta untuk mengisi kuesioner tentang pengembangan Code Maniac, responden merupakan mahasiswa filkom universitas brawijaya berjumlah 20 orang.

Tabel 6.18 Daftar Task SEQ

No	Nama Task	Task
1	CM01	Memulai aplikasi CodeManiac
2	CM02	Mengambil salah satu materi
3	CM03	Mulai mengerjakan materi
4	CM04	Bertanya kepada *Chatbot*

Tabel 6.19 Daftar Pertanyaan SEQ

No	List Pertanyaan
Learnability	
1	Menurut saya tampilan Code Maniac sudah menarik?
2	Menurut saya tampilan Chatbot sudah menarik?
3	Menurut saya menu-menu pada aplikasi ini mudah untuk dipahami?
4	Menurut saya tulisan teks yang digunakan mudah dan jelas?
Efficiency	
5	Menurut saya fitur Chatbot pada aplikasi yang ditampilkan dapat dipahami dengan mudah?
6	Menurut saya fitur Chatbot yang di klik dapat menampilkan dengan cepat?
Memorability	

7	Menurut saya tata letak desain interface dapat mudah diingat?
8	Menurut saya letak fitur telah ditempatkan dengan baik?
9	Menurut saya fitur bertanya pada bot mudah digunakan?
Satisfaction	
10	Menurut saya aplikasi ini sangat membantu saya?

Tabel 6.20 Hasil Kuesioner Skor setiap pertanyaan SEQ

No	Nama	Jabatan	Nilai Jawaban Pertanyaan									
			1	2	3	4	5	6	7	8	9	10
1	Responden 1	Mahasiswa	4	4	5	5	5	5	4	5	4	5
2	Responden 2	Mahasiswa	4	4	4	4	4	5	4	4	4	4
3	Responden 3	Mahasiswa	5	5	5	5	5	5	5	5	5	5
4	Responden 4	Mahasiswa	4	4	5	4	4	4	4	4	4	4
5	Responden 5	Mahasiswa	4	4	4	5	5	5	5	4	4	5
6	Responden 6	Mahasiswa	4	4	5	3	4	4	4	4	5	5
7	Responden 7	Mahasiswa	5	4	5	4	5	4	4	4	5	5
8	Responden 8	Mahasiswa	5	5	5	5	5	5	5	5	5	5
9	Responden 9	Mahasiswa	5	5	5	5	5	5	5	5	5	5
10	Responden 10	Mahasiswa	5	5	5	5	5	5	5	5	5	5
11	Responden 11	Mahasiswa	4	4	5	5	4	4	4	4	5	5
12	Responden 12	Mahasiswa	3	3	4	4	4	4	4	4	5	5
13	Responden 13	Mahasiswa	3	3	4	4	4	4	4	4	4	5
14	Responden 14	Mahasiswa	4	4	4	4	4	4	4	4	4	4
15	Responden 15	Mahasiswa	4	4	4	4	4	5	4	4	5	5
16	Responden 16	Mahasiswa	5	5	5	5	5	5	5	5	5	5
17	Responden 17	Mahasiswa	4	4	4	4	4	5	5	4	5	5
18	Responden 18	Mahasiswa	4	4	4	4	4	4	4	4	4	4
19	Responden 19	Mahasiswa	4	4	4	4	4	4	4	4	4	4
20	Responden 20	Mahasiswa	5	5	5	5	5	5	5	5	5	5

Keterangan :

A (Sangat setuju) = 5

B (Setuju) = 4

C (Netral) = 3

D (Tidak setuju) = 2

E (Sangat tidak setuju) = 1

Dalam Tabel 6.19 dibawah ini adalah bobot dari 20 responden. Dari 20 responden akan ditotal jumlah dari setiap jawaban.

Tabel 6.21 Bobot Kuesioner

No	List Pertanyaan	Jawaban				
		A	B	C	D	E
Learnability						
1	Menurut saya tampilan Code Maniac sudah menarik?	7	11	2	0	0
2	Menurut saya tampilan Chatbot sudah menarik?	6	12	2	0	0
3	Menurut saya menu-menu pada aplikasi ini mudah untuk dipahami?	11	9	0	0	0
4	Menurut saya tulisan teks yang digunakan mudah dan jelas?	9	10	1	0	0
Efficiency						
5	Menurut saya fitur Chatbot pada aplikasi yang ditampilkan dapat dipahami dengan mudah?	9	11	0	0	0
6	Menurut saya fitur Chatbot yang di klik dapat menampilkan dengan cepat?	11	9	0	0	0
Memorability						
7	Menurut saya tata letak desain interface dapat mudah diingat?	8	12	0	0	0
8	Menurut saya letak fitur telah ditempatkan dengan baik?	7	13	0	0	0
9	Menurut saya fitur bertanya pada bot mudah digunakan?	12	8	1	0	0
Satisfaction						
10	Menurut saya aplikasi ini sangat membantu saya?	15	5	0	0	0
TOTAL		94	100	6	0	0

Perhitungan & penyelesaian :

Jawaban A = 94 x 5 = 470

Jawaban B = 100 x 4 = 400

Jawaban C = 6 x 3 = 18

Jawaban D = 0 x 2 = 0

Jawaban E = 0 x 1 = 0

Jumlah (Jawaban A + Jawaban B + Jawaban C + Jawaban D) = 888

Jumlah Nilai Maksimal 20 responden x 10 Soal x 5 Jawaban = 1000

Presentase *Usability* = 888/1000 x 100%

= 0,888x 100%

= 88,8 %

Hasil perhitungan terhadap kuisioner pada pengujian usability untuk mengukur pengembangan aplikasi "Code Maniac" dapat dilihat dalam Tabel 6.15. dengan hasil 88% aplikasi masuk kedalam kategori berhasil dengan kualifikasi sangat baik.

Tabel 6.22 Data Responden pertanyaan pengembangan code maniac

No	Pertanyaan Terhadap Pengembangan CodeManiac	S	TS	Total
1	Saya lebih mudah memahami materi pemrograman dasar melalui CodeManiac	20	0	20
2	Saya belajar runtut sesuai pilihan materi	20	0	20
3	Letak chatbot mudah ditemukan	18	2	20
4	Chatbot mudah untuk digunakan	20	0	20
5	Fitur chatbot yang diberikan memudahkan saya memahami materi	19	1	20
6	Fitur chatbot yang tersedia memberi motivasi kepada saya untuk belajar pemrograman dasar lebih dalam	16	4	20
7	Fitur chatbot yang tersedia membuat saya risih karena selalu muncul disaat saya membuat kesalahan	9	11	20
8	Saya lebih mudah mengetahui ketika ada tantangan yang belum saya jawab	16	4	20

6.6 Analisis Hasil Pengujian

Setelah semua pengujian sistem CodeManiac telah selesai dilaksanakan, hasil dari seluruh pengujian dianalisis dan ditarik kesimpulan bahwa hasil dari seluruh pengujian yang telah selesai dilakukan telah memenuhi kebutuhan yang telah dijabarkan pada analisis kebutuhan.

6.6.1 Pengujian Unit

Dari seluruh pengujian unit yang dilakukan yaitu diantaranya `index()` dihasilkan beberapa *basis path* (jalur dasar) yaitu seluruh kemungkinan yang dapat terjadi tersebut telah diuji dan mendapatkan hasil yang valid pada setiap pengujiannya dan akhirnya semua pengujian sudah sepadan dengan kebutuhan yang telah dipaparkan pada analisis kebutuhan.

6.6.2 Pengujian Integrasi

Dalam pengujian integrasi fungsi `Insert()` terdapat 4 fungsi yang terintegrasi dengan fungsi utama yaitu fungsi `Insert()` yaitu fungsi `SubmitInsert(),Insert(),POST Knowledge() dan Insert()` fungsi tersebut merupakan fungsi yang dibuat untuk menguji *expected result* telah sesuai dengan kebutuhan. Selanjutnya fungsi setelah diimplementasikan pengujian dilakukan perbandingan untuk mengetahui apakah *expected result* telah sesuai dengan fungsi yang diimplementasikan.

6.6.3 Pengujian Validasi

Pengujian validasi menguji seluruh kebutuhan fungsional. Berikut ini adalah hasil analisis dari pengujian validasi pada sistem pengembangan *Chatbot* pada CodeManiac yang dipaparkan pada tabel 6.4.

Tabel 6.23 Pengujian Validasi

No	Kebutuhan	Kasus Uji	Hasil yang didapatkan	Status
1	CM-F-1-1	VAL-CM-1	Sistem menampilkan peringatan error pada saat *live coding*	Valid
2	CM-F-1-2	VAL-CM-2	Sistem menampilkan jawaban berdasarkan inputan pengguna	Valid
3	CM-F-1-3	VAL-CM-3	Sistem menampilkan peringatan error pada saat pengguna mengambil course secara tidak runtut	Valid
4	CM-F-1-4	VAL-CM-4	Sistem akan menampilkan peringatan ketika pengguna mengerjakan bab course secara tidak runtut	Valid
5	CM-F-2-1	VAL-CM-7	Sistem dapat menampilkan log hasil percakapan pengguna dengan *chatbot*	Valid
6	CM-F-2-2	VAL-CM-8	Sistem dapat menambah knowledge *chatbot* berdasarkan	Valid

			inputan admin	
7	CM-F-2-3	VAL-CM-9	Sistem dapat mengedit knowledge *chatbot* berdasarkan inputan admin	Valid
18	CM-F-2-4	VAL-CM-10	Sistem dapat menghapus knowledge *chatbot* berdasarkan inputan admin	Valid

6.6.4 Pengujian Pengetahuan Chatbot

Pengujian pengetahuan Chatbot dilakukan untuk menguji pengetahuan chatbot bagaimana respon chatbot terhadap inputan yang diberikan. Pada pengujian pengetahuan Chatbot menghasilkan hasil yang valid pada setiap kategori dimana chatbot akan menjawab berdasarkan pengetahuan yang ada.

6.6.5 Pengujian Usabilitas

Pengujian usabilitas dilakukan dengan menguji aplikasi kepada responden yaitu mahasiswa filkom universitas brawijaya untuk mencoba pengembangan lanjut aplikasi Code Maniac setelah itu responden diminta untuk mengisi kuisioner untuk menentukan nilai dari pengembangan aplikasi. Hasil pengujian usabilitas diukur menggunakan pengujian *post study* dengan metode *System Usability Scale* dengan nilai 88,8 yang berarti masuk kedalam kategori accceptable sehingga aplikasi sudah sesuai dengan kebutuhan pengguna.

BAB 7 PENUTUP

7.1 Kesimpulan

Berdasarkan hasil analisis kebutuhan, perancangan, implementasi, dan pengujian yang telah dilakukan, maka dapat disimpulkan sebagai berikut:

1. Berdasarkan hasil analisis kebutuhan yang dilakukan, pengembangan lanjut aplikasi CodeManiac untuk memberikan motivasi pengguna ini mempunyai 9 kebutuhan fungsional dan 1 kebutuhan non fungsional yang akan membantu dan memotivasi pengguna dalam proses pembelajaran. Penentuan kebutuhan fungsional diperoleh berdasarkan permasalahan yang ada yaitu dibutuhkannya suatu fitur untuk meningkatkan motivasi belajar pengguna. Selain itu juga diperoleh beberapa pemodelan kebutuhan yang digunakan agar memudahkan pengembang dalam memahami sistem, seperti *usecase* diagram, *class* diagram dan sequence diagram.

2. Setelah perancangan yang dilakukan berdasarkan analisis kebutuhan didapat perancangan data, perancangan komponen, perancangan antarmuka. Dalam perancangan arsitektur sistem diperoleh rancangan *class* diagram yang dinyatakan dengan rinci pada setiap operasi dan atributnya. Pada perancangan data dapatkan suatu rancangan database yang berupa PDM(Physical Data Model). Pada perancangan komponen terdapat algoritme-algoritme yang akan digunakan dan diimplementasikan pada sistem. Pada perancangan antarmuka berisi *mock-up* atau Gambaran dari sistem yang akan dibangun.

3. Penerapan AIML digunakan untuk menentukan jawaban ataupun balasan yang tepat berdasarkan pertanyaan/inputan dari pengguna, penerapan AIML tersebut digunakan pada saat implementasi program yang diterapkan setelah analisis kebutuhan. Data AIML yang digunakan diperoleh dari literatur *code error* pada pemrograman java. Data tersebut digunakan sebagai basis pengetahuan dari *chatbot.*

4. Pengembangan *chatbot* turut memotivasi pengguna dengan aktif menampilkan pesan berdasarkan perilaku pengguna.

5. Pengujian Usability mengenai kemudahan penggunaan aplikasi codemaniac yaitu sebesar 88,8 dimana angka sudah memenuhi standar dan tergolong acceptable dan baik.

6. Hasil pengembangan lanjut dari penelitian yang dilakukan adalah sebuah website yang mampu memotivasi pengguna dengan menerapkan AIML pada

basis pengetahuan *chatbot* dengan hasil pengujian black box dan white box 100% valid.

7.2 Saran

Berdasarkan penelitian yang telah dilakukan, maka penulis memberikan sejumlah saran yang dapat digunakan untuk penelitian selanjutnya, yakni:

1. Pengembangan sistem yang berjalan lebih baik, pengembangan lanjut pada *platform mobile* atau piranti cerdas lain seperti *smart watch.*
2. Menambahkan fitur lain seperti pengenalan/pendeteksian wajah pengguna.
3. Memperluas batasan inputan oleh pengguna seperti inputan suara, tautan link website, Gambar maupun video.
4. Integrasi dengan sistem lain seperti SIAM UB.

DAFTAR PUSTAKA

Abeysinghe, S. 2008. RESTful PHP Web Services. Packt Publishing, Birmingham.

Alinier, G., Hunt, B., Gordon, R. and Harwood, C. (2006) 'Issues and innovations in nursing education: Effectiveness of intermediate-fidelity simulation training technology in undergraduate nursing education', Journal of Advanced Nursing, vol. 54, no. 3, pp. 359–369.

Ambler, W., S, 2004. The object primer: agile modeling-driven development with UML 2.0. Cambridge University Press.

Bastari, I., D., 2017. Pengembangan Sistem Pembelajaran Pemrograman Java yang Atraktif Berbasis Website. Skripsi, Program Studi Sistem Informasi, Fakultas Ilmu Komputer, Universitas Brawijaya, Malang.

Brooke John. 2011. Measuring Usability With The System Usability Scale (SUS). http://www.measuringu.com/sus.php. Diakses pada tanggal 22 Desember 2018.

Cohen B. E, Nycz. M, 2006. Learning Objects and *E-Learning*: an Informing Science Perspective. Interdisciplinary Journal of Knowledge and Learning Objects, vol.2, 2006.

Gamage, D., Perera, I., & Fernando, S. 2014. Improving e-Learning to meet challenges in 21st century. 14th International Conference on Advances in ICT for Emerging Regions.

Jucius, J., M., 1971. Personnel Management. The Irwin series in management. 1971, Irwin, New York.

Karna, N. 2017. New Model of *e-Learning* based on Knowledge Management System. 2017 2nd International Conferences on Information Technology, Information Systems and Electrical Engineering (ICITISEE).

Keller, J., 2010. Motivational Design for Learning and Performance: The ARCS Model Approach, 2010, Springer Science & Business Media, New York.

Kristanto, A. 2003. Kecerdasan Buatan. Yogyakarta: Graha Ilmu.

Laksitowening A. K, Yanuaifiani P. A, Wibowo A. F. Y, 2016. Enhancing *E-Learning* System to Support Learning Style Based Personalization. 2nd International Conference on Science in Information Technology (ICSITech) 2016.

Noesgaard. S, Orngreen. R, *The Effectiveness of E-Learning: An Explorative and Integrative Review of Definitions, Methodologies and Factors that Promote e-Learning Effectiveness.* The Electronic Journal of eLearning Volume 13 Issue 4 2015, (pp278-290).

Pressman, R.S. 2010, Software Engineering : a practitioner's approach, McGraw-Hill, New York.

Utdirartatmo, F. 2001. Teknik Kompilasi.Yogyakarta: J&J Learning.

Rosenberg. M, 2001: *E-Learning*: Strategies for Delivering Knowledge in the Digital Age. McGraw-Hill Education 2001, Michigan University.

S. Wallace. Richard, 2003. *The Elements of AIML Style*. ALICE A.I. Foundation.

S. Wallace. Richard, 2005. *Be Your Own Botmaster*. ALICE A.I. Foundation.

S. Wallace. Richard, 2009. *Parsing the Turing Test: Philosophical and Methodological Issues in the Quest for the Thinking Computer.*

S. Wallace. Richard, 2010. Pandorabots. [online] Tersedia melalui: <https://www.pandorabots.com/botmaster/en/home> [Diakses 17 Januari 2018].

Sullivan, K. 2009. *An AIML Interpreter.* Thesis, Department of Computer Science and Engineering, Faculty of Technical Engineering, Czech Technical University in Prague, Prague.

Setiaji. B, Utami. E, Al Fatta. H, 2013. Membangun Chatbot Berbasis AIML dengan Arsitektur Pengetahuan Modular. Seminar Nasional Teknologi Informasi dan Multimedia 2013, STMIK AMIKOM Yogyakarta.

Sardiman,A.M.2006.Interaksi dan Motivasi Belajar Mengajar.Jakarta:Grafindo.

Turing, A.M. "Computing Machinery and Intelligence." Mind: A Quarterly Review of Psychology and Philosophy, October 1950: 433-460.

Whitten Bentley. 2007. System analysis & Design Methods, McGraw-Hill/Irwin, New York.

LAMPIRAN A KUESIONER MAHASISWA FILKOM

A.1 Pertanyaan Identitas Responden

Nama :

NIM :

A.2 Pertanyaan Terhadap Pemrograman Dasar

Keterangan :

- S : Setuju
- TS : Tidak Setuju

Tabel 7.1 Pertanyaan terhadap pemrograman dasar

No	Pertanyaan	S	T
1	Selama kegiatan pembelajaran pemgrograman dasar berlangsung saya menemui beberapa kendala		
2	Saya beberapa kali menemui kesulitan memahami materi pemrograman dasar yang disampaikan		
3	Pembelajaran pemrograman dasar dengan menggunakan teknik bermain sambil belajar (*gamification*) membantu saya dalam memahami materi		
4	Fitur tambahan untuk memotivasi akan sangat membantu saya dalam mempelajari pemrograman dasar		
5	Saya terkadang tidak mau bertanya kepada pemateri pemrograman dasar ketika saya menemui hal yang tidak dimengerti		

A.3 Pertanyaan Terhadap Pengembangan Code Maniac

Keterangan :

- S : Setuju
- TS : Tidak Setuju

Tabel 7.2 Pertanyaan terhadap Pengembangan code maniac

No	Pertanyaan	S	TS
1	Saya lebih mudah memahami materi pemrograman dasar melalui CodeManiac		
2	Saya belajar runtut sesuai pilihan materi		

3	Letak chatbot mudah ditemukan		
4	Chatbot mudah untuk digunakan		
5	Fitur chatbot yang diberikan memudahkan saya memahami materi		
6	Fitur chatbot yang tersedia memberi motivasi kepada saya untuk belajar pemrograman dasar lebih dalam		
7	Fitur chatbot yang tersedia membuat saya risih karena selalu muncul disaat saya membuat kesalahan		
8	Saya lebih mudah mengetahui ketika ada tantangan yang belum saya jawab		

A.4 Karakteristik Responden

Tabel 7.3 Karakteristik Responden

Karakteristik Responden		Frekuensi	Total
Angkatan	2013	1	20
	2014	19	

A.5 Data Responden

Tabel 7.4 Data Responden pertanyaan tentang pemrograman dasar

No	Pertanyaan Terhadap Pemrograman Dasar	S	T	Total
1	Selama kegiatan pembelajaran pemgrograman dasar berlangsung saya menemui beberapa kendala	17	3	20
2	Saya beberapa kali menemui kesulitan memahami materi pemrograman dasar yang disampaikan	17	3	20
3	Pembelajaran pemrograman dasar dengan menggunakan teknik bermain sambil belajar (*gamification*) membantu saya dalam memahami materi	19	1	20
4	Fitur tambahan untuk memotivasi akan sangat membantu saya dalam mempelajari pemrograman dasar	20	0	20
5	Saya terkadang tidak mau bertanya kepada pemateri pemrograman dasar ketika saya menemui hal yang tidak dimengerti	15	5	20

Tabel 7.5 Data Responden perntanyaan pengembangan code maniac

No	Pertanyaan Terhadap Pengembangan CodeManiac	S	TS	Total

1	Saya lebih mudah memahami materi pemrograman dasar melalui CodeManiac	20	0	20
2	Saya belajar runtut sesuai pilihan materi	20	0	20
3	Letak chatbot mudah ditemukan	18	2	20
4	Chatbot mudah untuk digunakan	20	0	20
5	Fitur chatbot yang diberikan memudahkan saya memahami materi	19	1	20
6	Fitur chatbot yang tersedia memberi motivasi kepada saya untuk belajar pemrograman dasar lebih dalam	16	4	20
7	Fitur chatbot yang tersedia membuat saya risih karena selalu muncul disaat saya membuat kesalahan	9	11	20
8	Saya lebih mudah mengetahui ketika ada tantangan yang belum saya jawab	16	4	20

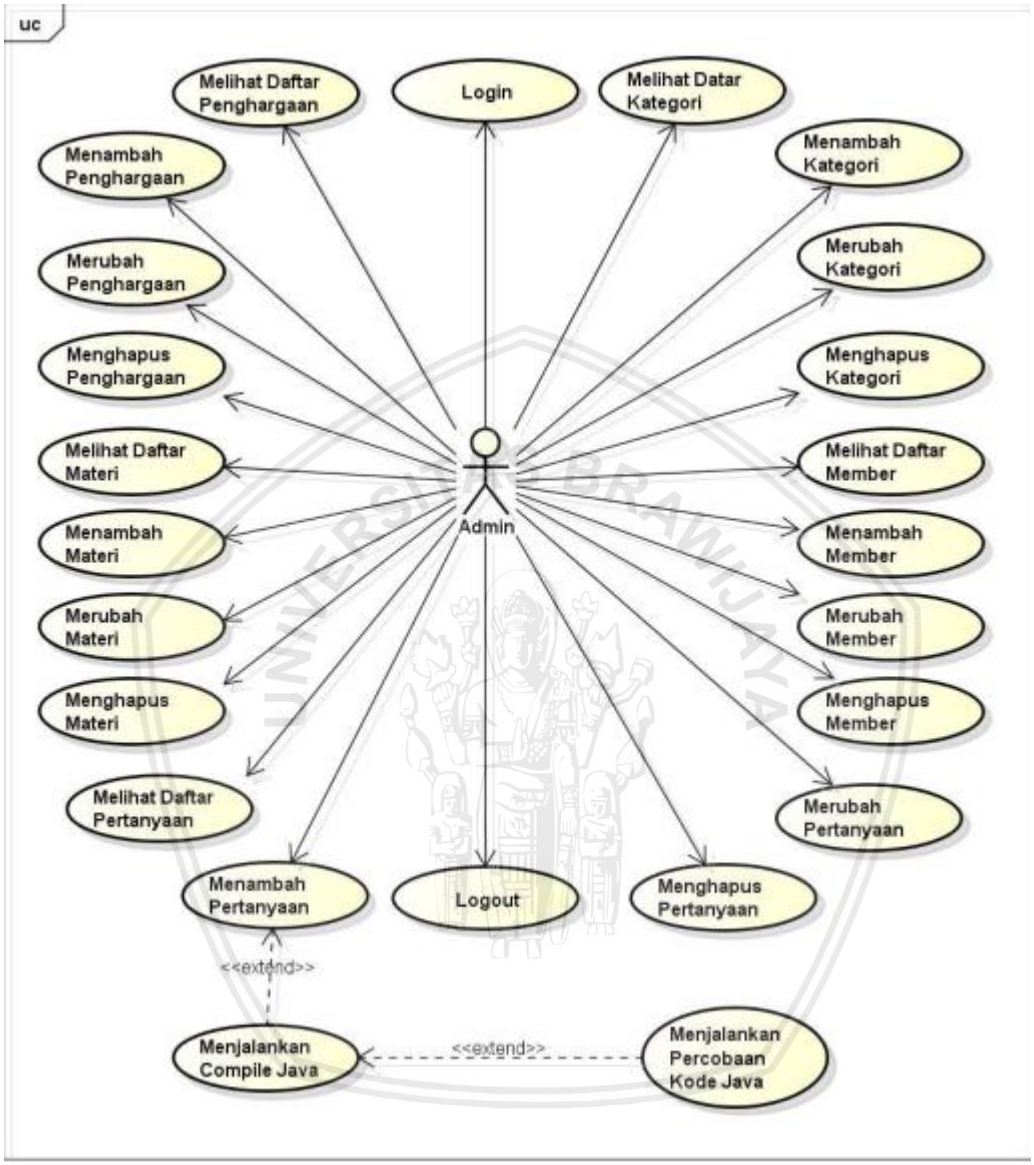
LAMPIRAN B DAFTAR KEBUTUHAN CODE MANIAC (COMA)

Tabel 8.7.6 Kebutuhan fungsional Code Maniac

No	Kode Fungsi	Kebutuhan Fungsional
1	CM – 1 – 001	Pengguna dapat melakukan login untuk mendapatkan autorisasi sebagai member.
2	CM – 1 – 002	Pengguna dapat melakukan registrasi untuk dapat melakukan login pada sistem sebagai member.
3	CM – 2 – 001	Admin dapat melihat daftar penghargaan yang tersedia pada sistem.
4	CM – 2 – 002	Admin dapat menambah penghargaan baru pada sistem.
5	CM – 2 – 003	Admin dapat melakukan perubahan penghargaan pada sistem.
6	CM – 2 – 004	Admin dapat menghapus penghargaan dari sistem.
7	CM – 2 – 005	Admin dapat melihat daftar kategori yang tersedia pada sistem.
8	CM – 2 – 006	Admin dapat menambah kategori baru ke dalam sistem.
9	CM – 2 – 007	Admin dapat merubah kategori pada sistem.
10	CM – 2 – 008	Admin dapat menghapus kategori pada sistem.
11	CM – 2 – 009	Admin dapat melihat daftar materi yang tersedia.
12	CM – 2 – 010	Admin dapat melakukan penambahan materi baru pada sistem.
13	CM – 2 – 011	Admin dapat melakukan perubahan materi pada sistem.
14	CM – 2 – 012	Admin dapat menghapus materi dari sistem.
15	CM – 2 – 013	Admin dapat melihat daftar pertanyaan yang tersedia pada sistem, baik pertanyaan latihan maupun pertanyaan untuk tantangan.
16	CM – 2 – 014	Admin dapat menambah pertanyaan baru ke dalam sistem.
17	CM – 2 – 015	Admin dapat melakukan perubahan pertanyaan pada sistem.
18	CM – 2 – 016	Admin dapat menghapus pertanyaan dari sistem.
19	CM – 2 – 017	Admin dapat melihat daftar member yang terdaftar pada sistem.

20	CM – 2 – 018	Admin dapat melakukan penambahan member baru ke dalam sistem.
21	CM – 2 – 019	Admin dapat melakukan perubahan pada member.
22	CM – 2 – 020	Admin dapat menghapus member dari sistem.
23	CM – 2 – 021	Admin dapat menjalankan *Compile* Kode Java.
24	CM – 2 – 022	Admin dapat menjalankan Percobaan Kode Java.
25	CM – 2 – 023	Admin dapat melakukan logout untuk keluar dari sistem.
26	CM – 2 – 024	Admin dapat melakukan login untuk mendapatkan autorisasi sebagai admin.
27	CM – 1 – 001	Member dapat melihat profil.
28	CM – 1 – 002	Member dapat melihat daftar riwayat latihan.
29	CM – 1 – 003	Member dapat melihat detil riwayat latihan.
30	CM – 1 – 004	Member dapat melihat daftar tantangan.
31	CM – 1 – 005	Member dapat mengajukan tantangan.
32	CM – 1 – 006	Member dapat mengerjakan tantangan.
33	CM – 1 – 007	Member dapat melihat detil riwayat tantangan.
34	CM – 1 – 008	Member dapat melihat daftar riwayat penghargaan.
35	CM – 1 – 009	Member dapat melihat daftar teman.
36	CM – 1 – 010	Member dapat mencari teman baru.
37	CM – 1 – 011	Member dapat melihat daftar permintaan pertemanan.
38	CM – 1 – 012	Member dapat melihat daftar materi.
39	CM – 1 – 013	Member dapat melihat daftar materi berdasarkan kategori.
40	CM – 1 – 014	Member dapat melihat detil materi.
41	CM – 1 – 015	Member dapat mengerjakan latihan materi.
42	CM – 1 – 016	Member dapat menjalankan Compile Kode Java.
43	CM – 1 – 017	Member dapat menjalakan Percobaan Kode Java.
44	CM – 1 – 018	Member dapat melakukan logout untuk keluar dari sistem.

LAMPIRAN C PEMODELAN KEBUTUHAN CODEMANIAC (COMA)



Gambar 8.7.1 Diagram Usecase Admin

Berikut penjelasan dari diagram usecase pada Gambar 8.0.1.

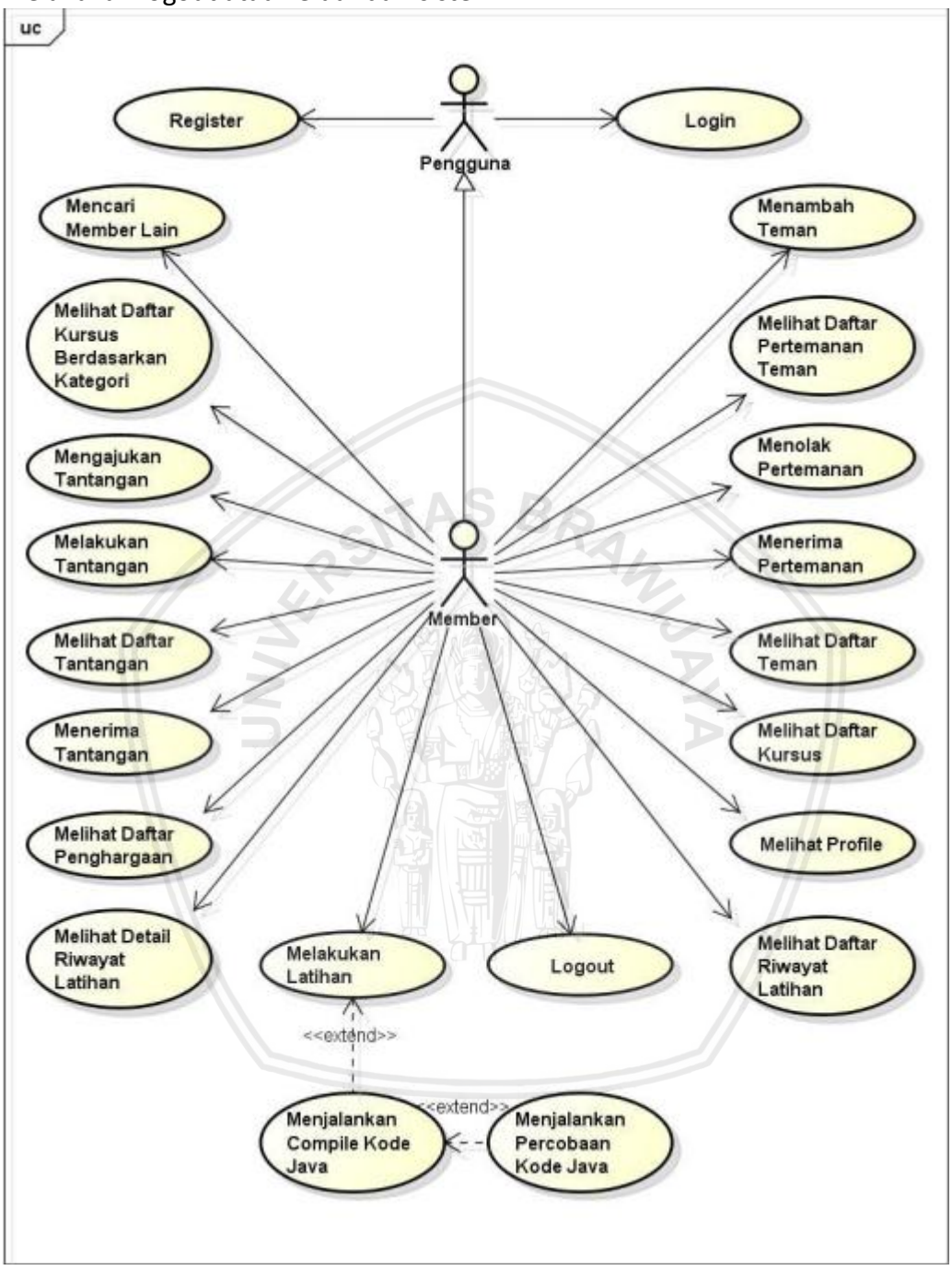
Pada usecase Login aktor dapat melakukan login untuk mendapatkan autorisasi sebagai admin. Selanjutnya, pada usecase Melihat Daftar Penghargaan aktor dapat melihat seluruh penghargaan yang tersedia pada sistem. Kemudian, pada usecase Menambah Penghargaan aktor dapat melakukan penambahan penghargaan baru ke dalam sistem, aktor dapat melakukan penambahan penghargaan setelah aktor memilih menu melihat daftar penghargaan. Pada usecase Merubah Penghargaan aktor dapat melakukan perubahan pada seluruh penghargaan pada sistem, aktor dapat melakukan perubahan penghargaan setelah aktor memilih menu melihat daftar penghargaan. Pada usecase

Menghapus Penghargaan aktor dapat melakukan menghapus seluruh penghargaan dari sistem, aktor dapat melakukan penghapusan penghargaan pada menu melihat daftar penghargaan. Kemudian, pada usecase Melihat Daftar Kategori aktor dapat melihat seluruh kategori yang tersedia pada sistem. Selanjutnya, pada usecase Menambah Kategori aktor dapat melakukan penambahan kategori baru kedalam sistem, aktor dapat melakukan ini yang tersedia pada menu Melihat Daftar Kategori. Pada usecase Merubah Kategori aktor dapat melakukan seluruh perubahan pada kategori pada sistem, aktor dapat melakukan ini yang tersedia pada menu Melihat Daftar Kategori. Pada usecase Menghapus Kategori aktor dapat melakukan penghapusan seluruh kategori pada sistem, aktor dapat melakukan penghapusan kategori yang tersedia pada Menu Melihat Daftar Kategori. Kemudian, pada usecase Melihat Daftar Materi aktor dapat melihat seluruh materi yang tersedia pada sistem. Selanjutnya, pada usecase Menambah Materi aktor dapat melakukan penambahan materi baru kedalam sistem, aktor dapat melakukan penambahan materi yang tersedia pada menu Melihat Daftar Materi. Pada usecase Merubah Materi aktor dapat melakukan perubahan pada materi yang terdapat didalam sistem, aktor dapat melakukan perubahan materi yang tersedia pada menu Melihat Daftar Materi.

Pada usecase Menghapus Materi aktor dapat menghapus materi yang terdapat didalam sistem, aktor dapat melakukan penghapusan materi yang tersedia pada menu Melihat Daftar Materi. Kemudian, pada usecase Melihat Daftar Member aktor dapat melihat seluruh member yang terdaftar pada sistem. Selanjutnya, pada usecase Menambah Member aktor dapat melakukan penambahan member baru ke dalam sistem, aktor dapat melakukan penambahan member yang tersedia pada menu Melihat Daftar Member. Pada usecase Merubah Member aktor dapat melakukan perubahan pada member, aktor dapat melakukan perubahan data member yang tersedia pada menu Melihat Daftar Member. Pada usecase Menghapus Member aktor dapat menghapus member dari sistem, aktor dapat menghapus member yang tersedia pada menu Melihat Daftar Member. Kemudian, pada usecase Melihat Daftar Pertanyaan aktor dapat melihat seluruh pertanyaan pada sistem. Selanjutnya, pada usecase Menambah Pertanyaan aktor dapat melakukan penambahan pertanyaan baru ke dalam sistem, aktor dapat melakukan penambahan pertanyaan yang tersedia pada menu Melihat Daftar Pertanyaan. Kemudian, pada usecase Merubah Pertanyaan aktor dapat melakukan perubahan pertanyaan di dalam sistem, aktor dapat melakukan perubahan pertanyaan yang tersedia pada menu Melihat Daftar Pertanyaan. Selanjutnya, pada usecase Menghapus Pertanyaan aktor dapat menghapus pertanyaan dari sistem, aktor dapat menghapus pertanyaan yang tersedia pada menu Melihat Daftar Pertanyaan.

Pada usecase Menjalankan Compile Kode Java aktor dapat memastikan bahwa kode java yang dimasukkan tidak memiliki kesalahan penulisan pada saat pembuatan soal baru. Kemudian, pada usecase Menjalankan Percobaan Kode Java aktor dapat memastikan bahwa kode java yang dimasukkan berjalan dengan

baik melalui beberapa percobaan. Selanjutnya, pada usecase Logout aktor dapat melakukan logout atau keluar dari sistem.



Gambar 8.7.2 Diagram Usecase Member dan Pengguna

Berikut penjelasan dari diagram use-case pada Gambar 8.0.2.

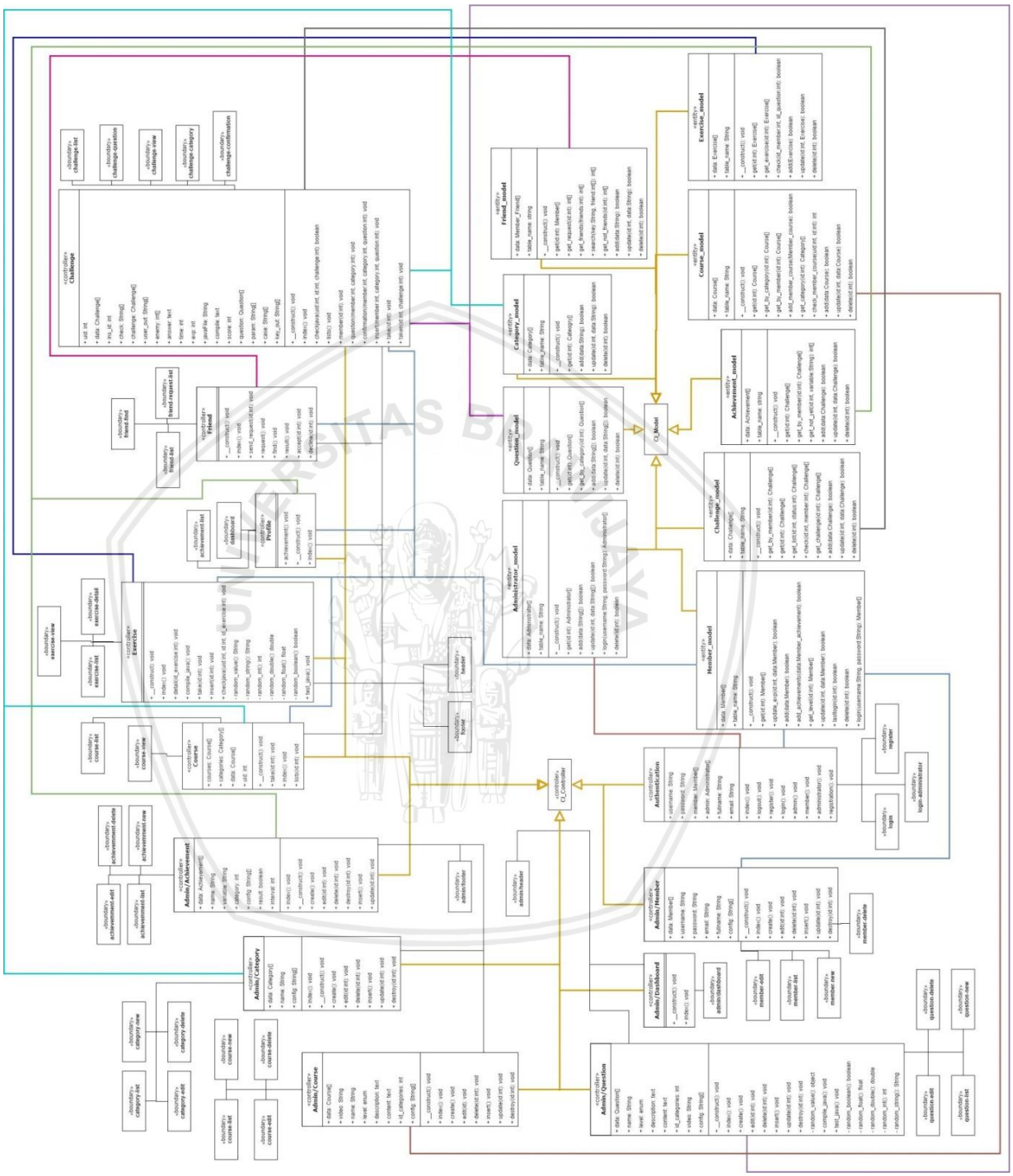
Pada usecase Login aktor dapat melakukan login untuk mendapatkan autorisasi sebagai member. Kemudian, pada usecase Registrasi aktor dapat melakukan pendaftaran untuk dapat hak akses melakukan login pada sistem sebagai member. Selanjutnya, pada usecase Melihat Profil aktor dapat melihat data secara detil yakni nama, foto , level, penghargaan terakhir, materi terakhir yang

dilihat serta tantangan terakhir. Pada usecase Melihat Daftar Riwayat Latihan aktor dapat melihat daftar seluruh latihan yang pernah dikerjakan, aktor dapat melihat daftar riwayat latihan yang tersedia pada menu Melihat Profil. Pada usecase Melihat Detil Riwayat Latihan aktor dapat melihat jawaban dan seluruh hasil uji coba jawaban pada pelaksaan latihan yang telah dilakukan, aktor dapat melihat daftar riwayat detil yang tersedia. Selanjutnya, pada usecase Melihat Daftar Tantangan aktor dapat melihat daftar seluruh tantangan yang pernah dikerjakan oleh member maupun tantangan yang ditujukan ke member. Kemudian, pada usecase Mengajukan Tantangan aktor dapat memberikan tantangan dalam bentuk latihan soal dengan member yang lain serta untuk hasil perbandingan, aktor dapat mengajukan tantangan yang tersedia pada menu Melihat Daftar Teman.

Pada usecase Menjalankan Tantangan aktor dapat mengerjakan tantangan yang ditujukan kepada member maupun yang aktor tujukan kepada member lain, aktor dapat menjalankan tantangan yang tersedia pada Menu Melihat Daftar Tantangan. Kemudian, pada usecase Melihat Detil Riwayat Tantangan aktor dapat melihat seluruh hasil uji coba jawaban member dan jawaban dari member lain pada pelaksanaan tantangan, aktor dapat melihat detil riwayat tantangan yang tersedia pada menu Melihat Daftar Tantangan. Selanjutnya, pada usecase Melihat Daftar Riwayat Penghargaan aktor dapat melihat seluruh penghargaan yang telah didapatkan setelah melakukan tantangan oleh member. Pada usecase Melihat Daftar Teman aktor dapat melihat seluruh member lain yang memiliki hubungan pertemanan dengan member.

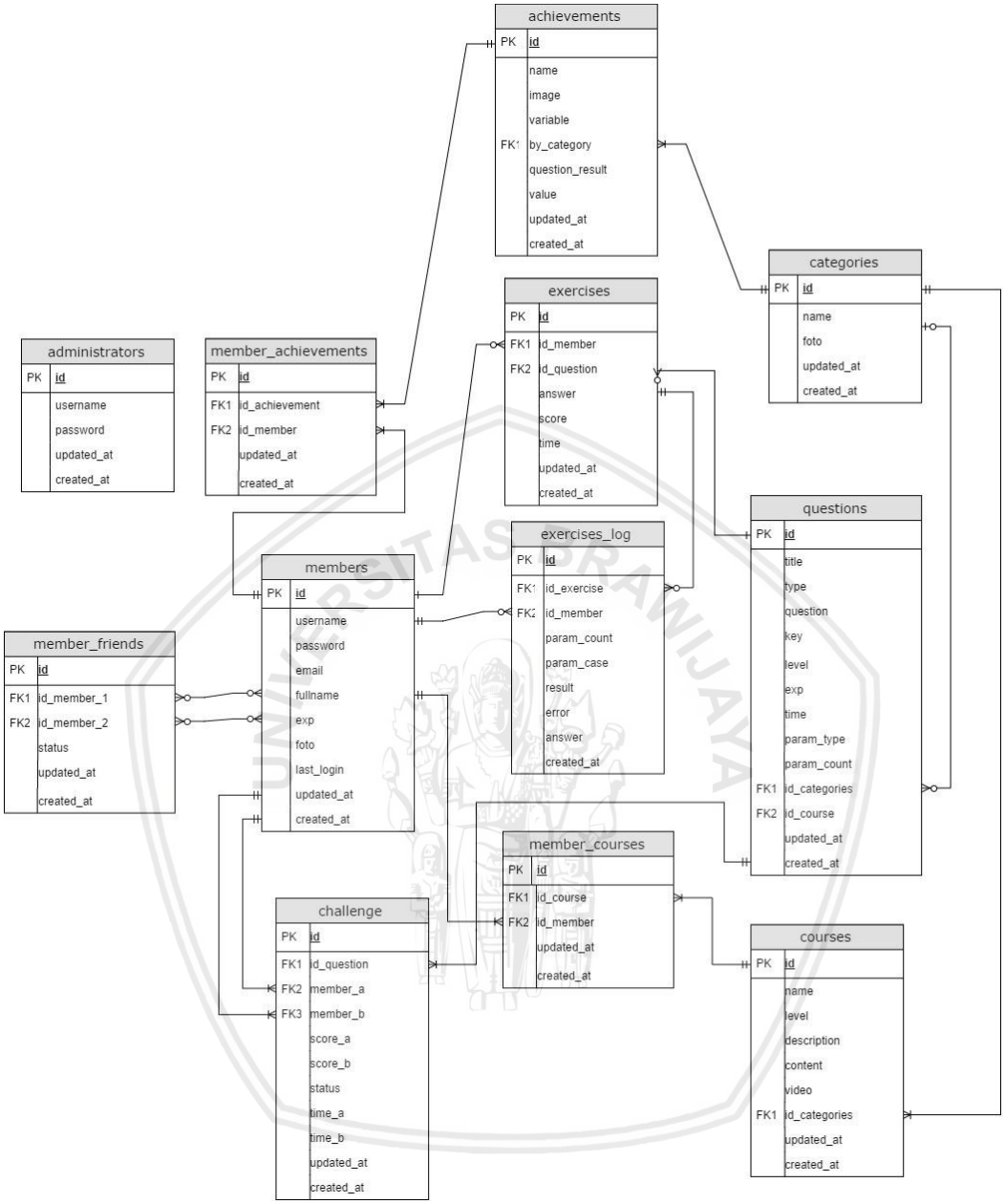
Pada usecase Mencari Teman Baru aktor dapat melakukan pencarian terhadap member lain yang belum memiliki hubungan pertemanan dengan member, aktor dapat melakukan pencarian teman baru yang tersedia pada menu Melihat Daftar Teman. Kemudian, pada usecase Melihat Daftar Permintaan Pertemanan aktor dapat melihat seluruh permintaan hubungan pertemanan dari member lain, yang memungkinkan member untuk menerima atau menolak pertemanan. Selanjutnya, pada Melihat Daftar Kursus aktor dapat melihat seluruh materi dari seluruh kategori yang tersedia pada sistem. Kemudian, pada usecase Melihat Daftar Kursus Berdasarkan Kategori aktor dapat melihat seluruh materi pada kategori tertentu yang telah dipilih member, aktor dapat melihat daftar kursus berdasarkan kategori yang tersedia pada menu Melihat Daftar Kursus. Pada usecase Melakukan Latihan aktor dapat mengerjakan pertanyaan latihan pada setiap materi yang tersedia didalam sistem, aktor dapat melakukan latihan yang tersedia pada menu Melihat Daftar Kursus. Kemudian, pada usecase Menjalankan Compile Kode Java aktor dapat memastikan bahwa kode java yang dimasukkan tidak memiliki kesalahan penulisan pada saat pengerjaan latihan materi. Selanjutnya, pada usecase Menjalankan Percobaan Kode Java aktor dapat memastikan bahwa kode java yang dimasukkan berjalan dengan baik melalui beberapa percobaan pada saat pengerjaan latihan materi. Kemudian, pada usecase Logout aktor dapat keluar dari hak akses sistem.

LAMPIRAN D PERANCANGAN ARSITEKTUR CODEMANIAC (COMA)



Gambar 8.3 Class Diagram Code Maniac

LAMPIRAN E PERANCANGAN DATA CODEMANIAC (COMA)



Gambar 8.4 Entity Relational Diagram

1. Tabel Administrator

Nama tabel : administrator

Jumlah *field* : 5

Fungsi : Untuk menyimpan biodata admin

Struktur tabel player

Tabel 8.7.7 Tabel Player

No.	Nama *field*	Tipe	Lebar	Keterangan
1	Id	Int(11)	-	id admin
2	Username	Varchar(255)	-	Username admin
3	Password	Varchar(32)	-	Password admin
4	Update_at	Timestamp	-	Tanggal diperbarui
5	Created_at	Timestamp	-	Tanggal dibuat

2. Tabel Achievements

Nama tabel : achievement

Jumlah field : 10

Fungsi : Untuk menyimpan data achievement

Tabel 8.7.8 Tabel Achievement

No.	Nama *field*	Tipe	Lebar	Keterangan
1	Id	Int(11)	-	id admin
2	Parent	Int(11)		
3	Name	Varchar(255)		Nama achievement
4	Image	text		Gambar medal
5	Variable	Enum('course','exercise', 'challenge')		Jenis achievement
6	By_category	Int(11)		
7	Question_result	Enum('100','0')	-	
8	value	Int(11)	-	
9	Update_at	Timestamp	-	Tanggal

				diperbarui
10	Created_ad	Timestamp	-	Tanggal dibuat

3. Tabel Categories

Nama tabel : categories

Jumlah field : 5

Fungsi : Untuk menyimpan data kategori

Tabel 8.7.9 Tabel Categories

No.	Nama *field*	Tipe	Lebar	Keterangan
1	Id	Int(11)	-	id admin
2	Name	Varchar(255)	-	Username admin
3	foto	text	-	Foto kategori
4	Update_at	Timestamp	-	Tanggal diperbarui
5	Created_ad	Timestamp	-	Tanggal dibuat

4. Tabel Challenge

Nama tabel : challenge

Jumlah field : 5

Fungsi : Untuk menyimpan biodata admin

Tabel 8.7.10 Tabel Challenge

No.	Nama *field*	Tipe	Lebar	Keterangan
1	Id	Int(11)	-	id member
2	Id_question	Int(11)		Id pertanyaan
3	Member_a	Int(11)		Member penantang
4	Member_b	Int(11)		Member penerima

5	Score_a	Int(11)		Skor penantang
6	Score_B	Int(11)		Skor penerima
7	Time_a	Time	-	Waktu penantang
8	Time_b	Time	-	Waktu penerima
9	Status	Tiny_int(1)		status
10	Update_at	Timestamp	-	Tanggal diperbarui
11	Created_ad	Timestamp	-	Tanggal dibuat

5. Tabel Courses

Nama tabel : course

Jumlah field : 9

Fungsi : Untuk menyimpan data course

Tabel 8.7.11 Tabel Courses

No.	Nama *field*	Tipe	Lebar	Keterangan
1	Id	Int(11)	-	id admin
2	Name	Int(11)		Nama course
3	Level	Enum('beginner','intermediate', advance')		Tingkat kesulitan
4	Description	Varchar(255)		Deskripsi course
5	Content	Text		Isi course
6	Video	Text		
7	Id_categories	Int(11)	-	Id kategori
8	Update_at	Timestamp	-	Tanggal

				diperbarui
9	Created_ad	Timestamp	-	Tanggal dibuat

6. Tabel Exercises

Nama tabel : exercises

Jumlah field : 8

Fungsi : Untuk menyimpan biodata admin

Tabel 8.7.12 Tabel Exercises

No.	Nama *field*	Tipe	Lebar	Keterangan
1	Id	Int(11)	-	id
2	Id_member	Int(11)		Id member
3	Id_question	Int(11)		Id pertanyaan
4	Answer	Longtext		Jawaban
5	Score	Int(11)		Skor
6	time	Varchar(20)		Waktu pengerjaan
7	Update_at	Timestamp	-	Tanggal diperbarui
8	Created_ad	Timestamp	-	Tanggal dibuat

7. Tabel Exercises Log

Nama tabel : exercises Log

Jumlah field : 9

Fungsi : Untuk menyimpan log exercises

Tabel 8.7.13 Tabel Exercises Log

No.	Nama *field*	Tipe	Lebar	Keterangan
1	Id	Int(11)	-	id
2	Exercise_id	Int(11)		
3	Member_id	Int(11)		
4	Param_count	Int(11)		
5	Param_case	Text		
6	Result	Text		
7	error	Text	-	
8	answer	Text		
9	Created_ad	Timestamp	-	Tanggal dibuat

8. Tabel Members

Nama tabel : member

Jumlah field : 11

Fungsi : Untuk menyimpan data member

Tabel 8.7.14 Tabel Members

No.	Nama *field*	Tipe	Lebar	Keterangan
1	Id	Int(11)	-	id
2	Username	Varchar(255)		Username member
3	Password	Varchar(32)		Password member
4	Email	Varchar(255)		Email member

5	Fullname	Varchar(255)		Nama lengkap member
6	Exp	Int(11)		Total experience
7	Foto	Varchar(255)	-	Foto member
8	Community	Varchar(255)		komunitas
9	Last_login	Datetime		Tanggal terakhir login
10	Update_at	Timestamp	-	Tanggal diperbarui
11	Created_ad	Timestamp	-	Tanggal dibuat

9. Tabel Member Achievements

Nama tabel : member achievements

Jumlah field : 5

Fungsi : Untuk menyimpan data penghargaan member

Tabel 8.7.15 Tabel Achievements

No.	Nama *field*	Tipe	Lebar	Keterangan
1	Id	Int(11)	-	id
2	Id_achievement	Int(11)		
3	Id_member	Int(11)		
4	Update_ad	timestamp		
5	Created_ad	Timestamp	-	Tanggal dibuat

10. Tabel Member Activities

Nama tabel : member activities

Jumlah field : 5

Fungsi : Untuk menyimpan data aktifitas member

Tabel 8.7.16 Tabel Member Activities

No.	Nama *field*	Tipe	Lebar	Keterangan
1	Id	Int(11)	-	id
2	Member_id	Int(11)		
3	Type	enum('course','challenge', 'exercise','achievement')		
4	Extra_id	Int(11)		
5	Created_ad	Timestamp	-	Tanggal dibuat

11. Tabel Member Courses

Nama tabel : member courses

Jumlah field : 5

Fungsi : Untuk menyimpan data course member

Tabel 8.7.17 Tabel Member Courses

No.	Nama *field*	Tipe	Lebar	Keterangan
1	Id	Int(11)	-	id
2	Id_course	Int(11)		
3	Id_member	Int(11)		
4	Updated_at	Timestamp		
5	Created_at	Timestamp	-	Tanggal dibuat

12. Tabel Member Friends

Nama tabel : administrator

Jumlah field : 5

Fungsi : Untuk menyimpan biodata admin

Tabel 8.7.18 Tabel Member Friends

No.	Nama *field*	Tipe	Lebar	Keterangan
1	Id	Int(11)	-	id
2	Id_member_1	Int(11)		
3	Id_member_2	Int(11)		
4	status	Tinyint(4)		
5	Updated_at	Timestamp		
6	Created_at	Timestamp	-	Tanggal dibuat

13. Tabel Questions

Nama tabel : Questions

Jumlah field : 15

Fungsi : Untuk menyimpan data question

Tabel 8.7.19 Tabel Questions

No.	Nama *field*	Tipe	Lebar	Keterangan
1	Id	Int(11)	-	Id
2	Title	Varchar(255)		Judul
3	Type	enum('exercise','challenge')		Tipe
4	Question	Text		Pertanyaan

5	Key	Text		Jawaban
6	Level	Enum('beginner','intermediate', advance')		Level kesulitan
7	Exp	Int(11)	-	
8	Time	Int(11)		
9	Param_type	Enum('random','manual')		
10	Param_count	Int(11)		
11	Param_case	longtext		
12	Id_categories	Int(11)		Id category
13	Id_course	Int(11)		Id course
14	Update_at	Timestamp	-	Tanggal diperbarui
15	Created_ad	Timestamp	-	Tanggal dibuat

14. Tabel Question log

Nama tabel : question log

Jumlah field : 7

Fungsi : Untuk menyimpan log question

Tabel 8.7.20 Tabel Question Log

No.	Nama *field*	Tipe	Lebar	Keterangan
1	Id	Int(11)	-	
2	Question_id	Int(11)		
3	Param_count	Int(11)		
4	Param_case	text		

5	Result	Text		
6	error	Text		
7	Created_at	Timestamp	-	