TUGAS AKHIR - KI091391

PEMBUATAN KAKAS KOMUNIKASI ANTAR PENGEMBANG PERANGKAT LUNAK

Anugerah Firdaus
NRP 5110100 089

Dosen Pembimbing
Daniel Oranova Siahaan, S.Kom., M.Sc., P.D.Eng.
Rizky Januar Akbar, S.Kom., M.Eng.

JURUSAN TEKNIK INFORMATIKA
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2014

FINAL PROJECT - KI091391

DEVELOPMENT OF A TOOL TO FACILITATE COMMUNICATION AMONG SOFTWARE DEVELOPERS

Anugerah Firdaus
NRP 5110100 089

Advisor
Daniel Oranova Siahaan, S.Kom., M.Sc., P.D.Eng.
Rizky Januar Akbar, S.Kom., M.Eng.

DEPARTMENT OF INFORMATICS
Faculty of Information Technology
Institut Teknologi Sepuluh Nopember
Surabaya 2014

Pembuatan Kakas Komunikasi Antar Pengembang Perangkat Lunak

Nama Mahasiswa	: Anugerah Firdaus
NRP	: 5110100089
Jurusan	: Teknik Informatika FTIf-ITS
Dosen Pembimbing 1	: Daniel O. Siahaan, S.Kom., M.Sc., P.D.Eng.
Dosen Pembimbing 2	: Rizky Januar Akbar, S.Kom., M.Eng.

ABSTRAK

Dalam sebuah proyek pengembangan perangkat lunak yang melibatkan banyak pengembang, komunikasi di antara pengembang sangat penting. Komunikasi dilakukan agar para pengembang dapat membantu pekerjaan individu masing-masing dan dapat bekerjasama pada setiap pekerjaan yang saling berkaitan. Saat bekerja, salah satu permasalahan yang dialami oleh para pengembang adalah kesulitan dalam malakukan pekerjaan. Para pengembang pasti membutuhkan bantuan dari pengembang lain yang memiliki keahlian terkait.

Dibutuhkan sebuah kakas komunikasi di antara para pengembang yang dapat mengklasifikasikan pesan yang diajukan oleh pengembang untuk memfasilitasi komunikasi dan memungkinkan pengembang mencari pengembang lain yang memiliki keahlian terkait. Digunakan metode Latent Semantic Indexing untuk mengklasifikasikan pesan pengembang berdasarkan bidang keahlian pengembang lainnya.

Kakas ini merupakan plugin untuk IDE Eclipse atau aplikasi yang terintegrasi dengan IDE Eclipse. Pesan yang diajukan oleh pengembang kemudian didekomposisi kedalam bentuk matriks yang akan digunakan dalam proses Singular Value Decomposition. Setelah itu, dilakukan pengurangan dimensi matriks. Cosine similarity digunakan untuk menghitung kesamaan pesan terhadap keahlian pengembang untuk mendapatkan pengembang lain dengan bidang keahlian yang sesuai dengan pesan yang diajukan. Kakas telah diuji dengan data uji dan didapatkan hasil rata-rata Kappa 0,61. Hasil pengujian menunjukkan bahwa tingkat reliability sistem adalah baik dan aplikasi dapat mengklasifikasikan pesan sesuai dengan keahlian pengembang lain.

Kata kunci: Cosine Similarity, Latent Semantic Indexing, Pesan, Plugin Eclipse, Singular Value Decomposition.

Development of a Tool to Facilitate Communication Among Software Developers

Student Name	: Anugerah Firdaus
Student ID	: 5110100089
Major	: Teknik Informatika FTIf-ITS
Advisor 1	: Daniel O. Siahaan, S.Kom., M.Sc., P.D.Eng.
Advisor 2	: Rizky Januar Akbar, S.Kom., M.Eng.

ABSTRACT

In a software development project that involves many developers, communication among developers is very important. Communication is performed so that developers can help each individual task and can work together on any tasks related to each other. While working, one of the problems faced by the developers is the difficulty in doing their task. The developers certainly need the help of another developer who posesses related expertise.

We need a tool of communication among developers which can classify messages submitted by developers to facilitate communication and to allow developers find other developers who have related expertise. Latent Semantic Indexing method is used to classify developer's messages based on other developers' areas of expertise.

This tool is a plugin for Eclipse IDE or an application which is integrated into Eclipse IDE. Messages submitted by developers then decomposed into a matrix which will be used in the process of Singular Value Decomposition. After that, the dimension reduction of matrix is performed. Cosine similarity is used to calculate similarity of messages against developers' expertise to get other developers who have related expertise in accordance to submitted messages. The tool has been tested with test data and the average results of Kappa obtained is 0,61. The test results showed that the level of system reliability is good and the application can classify a message according to the expertise of another developer.

Keywords: Cosine Similarity, Eclipse Plugin, Latent Semantic Indexing, Message, Singular Value Decomposition.

LEMBAR PENGESAHAN

Pembuatan Kakas Komunikasi Antar Pengembang Perangkat Lunak

TUGAS AKHIR

Diajukan Guna Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Bidang Studi Rekayasa Perangkat Lunak
Program Studi S-1 Jurusan Teknik Informatika
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember

Oleh :
ANUGERAH FIRDAUS
NRP : 5110 100 089

Disetujui oleh Dosen Pembimbing tugas akhir :

DANIEL ORANOVA SIAHAAN, S.Kom., M.Sc., P.D.Eng. ..... (pembimbing 1)
NIP: 197411232006041001

RIZKY JANUAR AKBAR, S.Kom., M.Eng. ..... (pembimbing 2)
NIP: 5100201301006

SURABAYA
JULI 2014

KATA PENGANTAR

Puji syukur kehadirat Allah Yang Maha Esa karena atas karunia dan rahmat-Nya penulis dapat menyelesaikan Tugas Akhir yang berjudul:

"Pembuatan Kakas Komunikasi Antar Pengembang Perangkat Lunak"

Dalam pembuatan Tugas Akhir ini, penulis mendapatkan banyak bantuan dari berbagai pihak. Oleh karena itu, tanpa mengurangi rasa hormat penulis ingin menyampaikan rasa terima kasih kepada:

1. Orang tua dan keluarga yang selalu memberikan dukungan untuk menyelesaikan Tugas Akhir ini.
2. Bapak Daniel Oranova Siahaan dan Bapak Rizky Januar Akbar selaku dosen pembimbing yang bersedia meluangkan waktu untuk memberikan bimbingan dan petunjuk dalam mengerjakan Tugas Akhir ini.
3. Bapak dan Ibu dosen Jurusan Teknik Informatika ITS yang telah memberikan banyak ilmu pengetahuan kepada penulis selama masa perkuliahan.
4. Seluruh staf dan karyawan Jurusan Teknik Informatika ITS yang memberikan kelancaran administasi akademik penulis.
5. Teman-teman angkatan 2010 Jurusan Teknik Informatika ITS yang bersama-sama berjuang selama empat tahun atas saran, masukan dan dukungan terhadap pengerjaan Tugas Akhir ini.
6. Serta pihak-pihak lain yang tidak dapat disebutkan, dan telah membantu penulis dalam penyusunan Tugas Akhir ini.

Penulis telah berusaha dalam menyusun Tugas Akhir ini dengan sebaik mungkin, tetapi penulis mohon maaf apabila terdapat kesalahan dan kekurangan yang dilakukan oleh penulis.

Kritik dan saran yang membangun dapat disampaikan sebagai bahan perbaikan selanjutnya.

Surabaya, Juli 2014

Anugerah Firdaus

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR KODE SUMBER

DAFTAR TABEL

BAB I
PENDAHULUAN

Bab ini menjelaskan garis besar Tugas Akhir yang meliputi latar belakang, tujuan, rumusan dan batasan permasalahan, metodologi pembuatan Tugas Akhir, dan sistematika penulisan buku Tugas Akhir.

1.1. Latar Belakang

Di masa sekarang, teknologi informasi semakin banyak digunakan oleh manusia dalam berbagai kegiatan sehari-hari. Terdapat berbagai macam perangkat lunak yang dikembangkan dalam berbagai perangkat. Karena kebutuhan manusia akan perangkat lunak yang semakin berkembang, banyak orang, baik individu ataupun organisasi, menjadi pengembang perangkat lunak. Sebuah perangkat lunak dapat dibuat oleh seorang individu atau sebuah organisasi berdasarkan skala kebutuhan. Sebuah proyek pengembangan perangkat lunak skala menengah sampai besar, biasanya dibuat oleh tim pengembang perangkat lunak. Sebuah tim pengembang terdiri dari beberapa orang yang memiliki peran dan keahlian masing-masing. Peran dari masing-masing individu tersebut antara lain adalah: ketua proyek, analis, desainer, *programmer*, dan penguji.

Agar dapat menghasilkan perangkat lunak dengan kualitas yang bagus dan dengan biaya serta waktu yang minimal maka tim pengembang harus memiliki produktivitas kerja yang tinggi. Produktivitas kerja dapat dilihat dari proses pengembangan perangkat lunak dalam sebuah tim. Dalam prosesnya, setiap pengembang memiliki pekerjaan masing-masing sesuai dengan bidang keahlian. Meskipun setiap pengembang bekerja secara individu dalam melakukan pekerjaan, tetapi setiap pengembang pasti bekerja bersama-sama karena setiap pekerjaan dalam sebuah tim tersebut saling berkaitan.

Dalam sebuah tim pengembang, diperlukan untuk saling kerjasama dan saling membantu antar anggota tim. Selain

mengerjakan pekerjaan individu, pengembang juga harus melakukan komunikasi karena komunikasi dalam sebuah tim merupakan hal yang penting dalam proses pengembangan perangkat lunak. Bentuk komunikasi bisa berupa tanya jawab antar individu atau diskusi antar tim pengembang, baik secara tatap muka ataupun melalui media elektronik. Pengembangan perangkat lunak yang baik pasti didukung dengan komunikasi antar anggota tim yang baik. Oleh karena itu, komunikasi antar anggota tim pengembang perlu untuk dilakukan. Apabila mengalami kesulitan dalam mengerjakan pekerjaannya, seorang pengembang akan bertanya pada pengembang lain. Akan tetapi, hal tersebut dapat menimbulkan permasalahan apabila pengembang yang ditanya tidak memiliki keahlian yang sesuai atau tidak paham dengan pekerjaan yang ditanyakan. Hal tersebut tentunya akan menghambat pekerjaan pengembang karena pekerjaan yang ditanyakan belum dapat diselesaikan. Oleh karena itu, pengembang tersebut perlu mencari pengembang lain dengan bidang keahlian yang sesuai dengan pekerjaan yang ditanyakan sehingga dapat membantu dalam mengerjakan pekerjaannya.

Terdapat beberapa perangkat lunak yang berfungsi dalam mengatur komunikasi antar pengembang perangkat lunak. Salah satunya adalah perangkat lunak yang dikembangkan oleh Nishinaka dkk., yaitu STeP_IN (*Socio-Technical Platform for In situ Networking*). Perangkat lunak tersebut berupa kerangka kerja yang dapat digunakan oleh pengembang untuk mengatur komunikasi antar pengembang yang berada dalam lingkungan kerja yang sama. Perangkat lunak ini dapat membantu pengembang perangkat lunak untuk mencari dan mempelajari pustaka Java API. Selain itu juga menyediakan antarmuka pencarian individual untuk pengembang Java, contoh yang menggambarkan penggunaan, dan secara spesifik merupakan suatu mekanisme yang memfasilitasi pengembang Java untuk bertukar keahlian berdasarkan keahlian serta hubungan sosial diantara pengembang [1].

Dalam Tugas Akhir ini akan dibahas mengenai proses klasifikasi pesan sesuai dengan keahlian setiap pengembang.

Untuk mengklasifikasikan pesan yang diajukan oleh pengembang kepada pengembang lain, diperlukan sebuah metode untuk mengklasifikasikan pesan berdasarkan keahlian pengembang lainnya. Salah satu metode yang dapat digunakan untuk klasifikasi adalah *Latent Semantic Indexing* (LSI). LSI adalah teknik yang digunakan dalam pengambilan informasi dimana terdapat sebuah struktur pokok yang merepresentasikan hubungan antar kata. Metode ini menggunakan *Singular Value Decomposition* (SVD) untuk merepresentasikan kata pada setiap deskripsi keahlian pengembang. Setelah itu digunakan perhitungan *Cosine Similarity* untuk menemukan kesamaan antara pesan yang diajukan dengan deskripsi keahlian setiap pengembang. Dari perhitungan tersebut akan ditemukan pengembang lain dengan keahlian yang sesuai dengan pesan yang diajukan.

1.2. Tujuan

Tujuan pembuatan Tugas Akhir ini adalah sebagai berikut.

1. Membuat kakas komunikasi antar pengembang perangkat lunak yang terintegrasi dengan *IDE Eclipse*.
2. Membuat kakas yang dapat melakukan komunikasi berdasarkan klasifikasi pesan yang dikirimkan oleh pengembang perangkat lunak atau pengguna.

1.3. Rumusan Permasalahan

Rumusan permasalahan yang akan diselesaikan pada Tugas Akhir ini adalah sebagai berikut.

1. Bagaimana membuat kakas komunikasi antar pengembang perangkat lunak yang terintegrasi dengan *IDE Eclipse*.
2. Bagaimana mengatur komunikasi berupa pesan dengan mengarahkannya kepada rekan kerja pengembang atau pengguna lain berdasarkan bidang keahlian.

1.4. Batasan Permasalahan

Permasalahan yang akan diselesaikan dalam Tugas Akhir ini memiliki beberapa batasan, di antaranya adalah sebagai berikut.

1. Kakas komunikasi adalah *plugin* untuk *IDE Eclipse* versi 3.5 hingga versi lebih dari 3.5.
2. Pesan yang akan dikirimkan oleh pengguna berhubungan dengan tim proyek perangkat lunak.
3. Pesan yang dikirimkan menggunakan bahasa Indonesia.

1.5. Metodologi

Tahap-tahap yang dilakukan dalam mengerjakan Tugas Akhir ini adalah:

1. **Studi literatur**

   Pada tahap ini dilakukan pengumpulan informasi dan pustaka yang diperlukan dalam proses perancangan dan implementasi perangkat lunak yang dibangun. Pustaka yang digunakan adalah:

   a. konsep *client* dan *server* menggunakan *thread* dan *socket*;
   b. konsep *input/output* pada pemrograman Java;
   c. konsep *plugin* IDE Eclipse;
   d. algoritma *Singular Value Decomposition*;
   e. algoritma *Cosine Similarity*; dan
   f. algoritma *Latent Semantic Indexing*.

2. **Analisis dan perancangan sistem**

   Pada tahap ini dilakukan analisa kebutuhan perangkat lunak untuk menemukan solusi dan memecahkan masalah dari bab rumusan permasalahan. Kebutuhan fungsional utama dari perangkat lunak adalah dapat berkomunikasi dengan mengirimkan pesan pada Eclipse dan klasifikasi pesan sesuai dengan keahlian menggunakan metode *Latent Semantic Indexing* (LSI). Perancangan sistem meliputi perancangan arsitektur, desain kelas sistem, sistem klasifikasi pesan, dan antarmuka.

3. **Implementasi**
   Pada tahap ini dilakukan tahapan dalam pembuataan perangkat lunak. Pengembangan perangkat lunak pada Tugas Akhir ini adalah sebuah *plugin* untuk Eclipse. Masukan dari perangkat lunak ini berupa pesan yang dikirimkan oleh pengguna, kemudian digunakan metode *Latent Semantic Indexing* untuk menemukan hubungan antara pesan yang dikirimkan dengan bidang keahlian pengguna lain. Keluaran yang dihasilkan dari perangkat lunak adalah pesan dikirimkan kepada pengguna dengan tingkat kesamaan yang tinggi terhadap pesan yang dikirimkan oleh pengirim. Dalam pengembangan perangkat lunak ini dibutuhkan adalah:
   a. IDE Eclipse;
   b. Bahasa Pemrograman Java;
   c. *Java Development Kit 7.0*; dan
   d. *Library* Java.

4. **Pengujian dan evaluasi**
   Pada tahap ini dilakukan pengujian terhadap perangkat lunak yang dibuat. Pengujian dilakukan untuk menemukan kesalahan dalam pembuatan perangkat lunak agar dapat segera diperbaiki dan dievaluasi kembali. Percobaan menggunakan data percobaan sebagai uji coba masukan perangkat lunak. Data yang akan dibuat sebagai data masukan adalah data pengguna dan pesan yang akan dikirim. Keluaran dari percobaan adalah pesan dikirimkan kepada pengguna lain. Hasil yang diharapkan dari percobaan adalah pesan dikirimkan kepada pengguna yang memiliki keahlian sesuai dengan pesan tersebut.

5. **Penyusunan buku tugas akhir**
   Pada tahap ini dilakukan dokumentasi dan penyusunan laporan yang menjelaskan dasar teori dan metode yang digunakan dalam Tugas Akhir, proses dan implementasi

perangkat lunak, serta hasil dari implementasi perangkat lunak yang telah didapatkan selama pengerjaan Tugas Akhir.

1.6. Sistematika Penulisan

Buku Tugas Akhir ini bertujuan untuk memberikan gambaran dari pengerjaan Tugas Akhir. Diharapkan, buku ini juga dapat memberikan manfaat dan berguna bagi pembaca untuk melakukan pengembangan lebih lanjut. Buku Tugas Akhir ini terdiri dari beberapa bagian seperti berikut ini.

Bab I Pendahuluan

Bab ini berisi latar belakang permasalahan, tujuan dan manfaat pembuatan Tugas Akhir, rumusan permasalahan, batasan permasalahan, metodologi yang digunakan, dan sistematika penyusunan Buku Tugas Akhir.

Bab II Dasar Teori

Bab ini berisi penjelasan teori penunjang yang berhubungan dengan pokok pembahasan dan mendasari pembuatan Tugas Akhir.

Bab III Analisis dan Perancangan Sistem

Bab ini berisi pembahasan analisa dan perancangan perangkat lunak. Perancangan perangkat lunak meliputi perancangan arsitektur, proses klasifikasi pesan dan antarmuka kakas.

Bab IV Implementasi

Bab ini berisi implementasi dari perancangan perangkat lunak.

Bab V Pengujian dan Evaluasi

Bab ini berisi pembahasan pengujian kegunaan dan fungsionalitas dari perangkat lunak dengan memperhatikan keluaran yang dihasilkan oleh perangkat lunak.

Bab VI Kesimpulan

Bab ini berisi pembahasan kesimpulan dari hasil pengujian yang dilakukan dan saran untuk pengembangan perangkat lunak selanjutnya.

Daftar Pustaka

Berisi daftar referensi yang digunakan untuk pembuatan tugas akhir.

Lampiran

Merupakan bab tambahan yang berisi data tambahan yang penting pada aplikasi ini.

BAB II
DASAR TEORI

Bab ini menjelaskan pembahasan mengenai teori-teori yang menjadi dasar dalam pembuatan Tugas Akhir. Teori-teori tersebut adalah *Singular Value Decomposition, Cosine Similarity*, *Latent Semantic Indexing,* Eclipse, *client/server*, pemrograman Java, dan *Cohen's kappa*.

2.1. *Singular Value Decomposition (SVD)*

Dalam SVD, terdapat sebuah matriks $M \times N$, dimana $M \neq N$. Misalkan terdapat $M \times N$ matriks C, dimana setiap baris dalam matriks C merepresentasikan kata, setiap kolom dalam matriks C merepresentasikan dokumen dan setiap sel dalam matriks C berisi bobot kata terhadap dokumen. Maka, SVD dapat ditulis sebagai perkalian tiga matriks yang lain seperti pada Persamaan 2.1.

$$C = U\sigma V^T \quad \textbf{(2.1)}$$

Matriks U adalah matriks $M \times M$, dimana kolom matriks U adalah *orthogonal eigenvector* dari CC^T. Matriks V adalah matriks $N \times N$, dimana kolom matriks V adalah *orthogonal eigenvector* dari C^TC. C^T adalah *transpose* dari matriks C. *Eigenvalue* dari CC^T sama dengan *eigenvalue* dari C^TC. Matriks σ adalah matriks diagonal *singular value* dari C [2].

2.2. *Cosine Similarity*

Untuk mengukur kesamaan antara dua dokumen dalam vektor ruang, caranya adalah dengan mempertimbangkan besarnya perbedaan vektor antara dua vektor dokumen. Tetapi langkah ini memiliki kelemahan, yaitu dua dokumen dengan isi yang sangat mirip dapat memiliki perbedaan vektor yang signifikan dikarenakan salah satu memiliki panjang dokumen yang lebih dari yang lain. Untuk mengimbangi pengaruh panjang dokumen, cara standar untuk mengukur kesamaan antara dua dokumen d_1 dan d_2

adalah dengan menghitung kesamaan kosinus atau *cosine similarity* dari representasi vektor $V(d_1)$ dan $V(d_2)$ seperti pada Persamaan 2.2.

$$sim(d_1, d_2) = \frac{V(d_1).V(d_2)}{|V(d_1)||V(d_2)|} \quad \textbf{(2.2)}$$

Dimana pembilangnya merepresentasikan *dot product* (disebut juga *inner product*) dari vektor $V(d_1)$ dan $V(d_2)$, sedangkan penyebutnya adalah produk dari panjang *Euclidean* [2].

Dalam kasus pencarian informasi, seperti pada metode *Latent Semantic Indexing*, dapat dicari kesamaan *cosinus* dari dua dokumen atau antara *query* dan dokumen. *Cosine Similarity* diantara vektor *query* dan dokumen dapat dihitung seperti pada Persamaan 2.3.

$$sim(q, d) = \frac{V(q).V(d)}{|V(q)||V(d)|} \quad \textbf{(2.3)}$$

2.3. *Latent Semantic Indexing (LSI)*

LSI adalah teknik yang digunakan dalam pengambilan informasi. Asumsi yang mendasari LSI adalah terdapat sebuah struktur pokok atau *latent* yang merepresentasikan hubungan antar kata dalam sejumlah dokumen berdasarkan terjadinya kata dalam dokumen-dokumen tersebut. Sebagai contoh, jika terdapat dokumen *A* yang berisi *(w1, w2)* dan dokumen *B* yang berisi *(w2, w3),* dapat disimpulkan bahwa ada sesuatu yang umum antara dokumen *A* dan *B*.

LSI menerima sebuah vektor atau sebuah matriks frekuensi dari sekumpulan dokumen, dimana setiap baris mewakili satu istilah atau kata, setiap kolom mewakili satu dokumen dan setiap sel berisi nilai bobot kata terhadap dokumen. Representasi dalam vektor ini memiliki kelebihan, yaitu perlakuan yang sama terhadap *query* dan dokumen sebagai vektor, perhitungan nilai induksi berdasarkan *cosine similarity*, dan pengambilan informasi seperti pengelompokkan serta pengklasifikasian. Namun, representasi ini

memiliki kelemahan karena ketidakmampuan dalam mengatasi permasalahan bahasa, seperti sinonim dan polisemi [2].

LSI menggunakan SVD dalam penerapannya. Matriks frekuensi dari kata dan dokumen diuraikan menjadi tiga buah matriks yang berbeda seperti pada Persamaan 2.1. Kemudian dilakukan pengurangan dimensi sebanyak *k* dari ketiga matriks tersebut dan dibuat matriks SVD yang baru.

Digunakan representasi LSI dengan dimensi *k* untuk menghitung kesamaan diantara vektor. Vektor *query q* dapat dituliskan dalam representasi LSI dengan dimensi *k* seperti pada Persamaan 2.4.

$$q_k = \sigma_k^{-1} U_k^T q \quad \textbf{(2.4)}$$

Persamaan 2.4 tidak hanya digunakan untuk menghitung vektor *query q*, tetapi juga dapat digunakan untuk menghitung vektor dokumen *d* seperti pada Persamaan 2.5.

$$d_k = \sigma_k^{-1} U_k^T d \quad \textbf{(2.5)}$$

Setelah menghitung vektor *query* dan dokumen digunakan metode *cosine similarity* untuk menghitung kesamaan diantara *query* dan dokumen seperti pada Persamaan 2.3.

2.4. Eclipse

Eclipse adalah sebuah IDE (*Integrated Development Environment*) atau lingkungan pengembangan yang terintegrasi untuk bahasa pemrograman Java. Eclipse dibuat oleh komunitas *open source* dan digunakan dalam beberapa bidang yang berbeda, seperti pada lingkungan pengembangan untuk Java atau aplikasi Android. Eclipse dapat ditambahkan dengan berbagai komponen perangkat lunak tambahan yang biasa disebut sebagai *plugin* [3].

Meskipun Eclipse dirancang untuk melayani sebagai *open platform*, tetapi Eclipse dirancang demikian sehingga komponen-komponennya dapat digunakan untuk membangun hampir semua aplikasi klien. Dibutuhkan sekumpulan *plugin* untuk membangun

aplikasi *rich client* yang dikenal sebagai *Rich Client Platform* (RCP). Berikut ini adalah komponen yang membentuk RCP [4]:

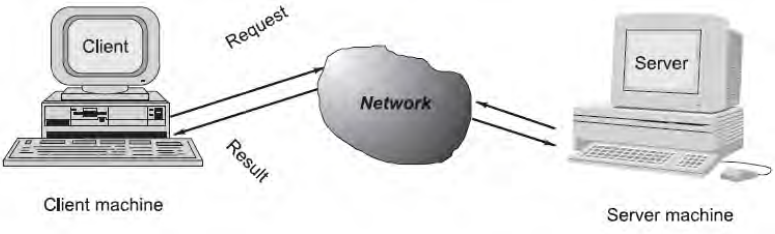
1. *Eclipse Runtime;*
2. *SWT (Standard Widget Toolkit);*
3. *JFace;* dan
4. *Workbench.*

Secara standar Eclipse selalu dilengkapi dengan JDT (*Java Development Tools*). Proyek JDT menyediakan *plugin* yang menerapkan IDE Java dan mendukung pengembangan aplikasi Java, termasuk *plugin* Eclipse [5].

Eclipse juga dilengkapi dengan PDE (*Plugin Development Environment*) yang menyediakan alat untuk membuat, mengembangkan, menguji, *debug*, membangun dan menyebarkan *plugin* Eclipse, fragmen, fitur, *update* situs dan produk RCP. PDE juga menyediakan perkakas OSGi komprehensif, sehingga dapat menjadi lingkungan yang ideal untuk pemrograman komponen, bukan hanya pengembangan *plugin* Eclipse [6].

2.5. *Client/Server*

Sistem berbasis jaringan terdiri dari sebuah *server*, *client* dan media untuk komunikasi. Sebuah komputer yang menjalankan program dan membuat permintaan untuk layanan disebut komputer *client*. Sebuah komputer yang menjalankan program dan menawarkan layanan yang diminta dari satu atau lebih *client* disebut sebagai komputer *server*. Media untuk komunikasi dapat berupa kabel atau jaringan nirkabel. *Socket* menyediakan antarmuka pemrograman jaringan untuk jaringan pada lapisan transportasi. Sebuah *server* yang berjalan pada komputer tertentu memiliki *socket* yang terikat ke sebuah *port* tertentu. *Server* mendengarkan koneksi *socket* untuk *client* agar dapat membuat permintaan koneksi. Jika koneksi berjalan dengan lancar, maka *server* akan menerima koneksi dari *client* [7]. Arsitektur *client/server* dapat dilihat seperti pada Gambar 2.1.



Gambar 2.1 Arsitektur *Client/Server*

2.6. Pemrograman Java

Java adalah bahasa pemrograman berorientasi objek yang dikembangkan oleh Sun Microsystems dan dirilis pada tahun 1995. Pada awalnya Java dimiliki oleh Sun Microsystems, tetapi kemudian dirilis ke *open source*. Diperkenalkan pada tahun 1996 untuk Solaris, Windows, Mac OS Classic dan Linux, Java awalnya dirilis sebagai *Java Development Kit* 1.0 (JDK 1.0). Hal ini termasuk Java *runtime* (mesin virtual dan pustaka kelas) dan *tools* pengembangan (misalnya, Java *compiler*). Kemudian Sun juga menyediakan paket *runtime-only*, disebut *Java Runtime Environment* (JRE) [8]. Bahasa ini dirancang agar lebih mudah dipakai dan *platform independent*, yaitu dapat dijalankan di berbagai jenis sistem operasi dan arsitektur komputer. Beberapa contoh pemrograman Java yang digunakan dalam Tugas Akhir ini adalah *thread, Input/Output*, jaringan dan JAMA.

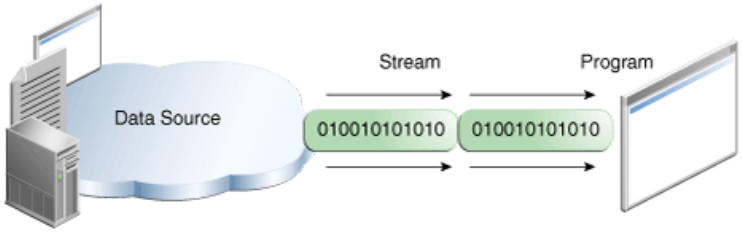
2.6.1. *Thread*

Java adalah bahasa pemrograman *multithreaded* yang berarti pengembang dapat mengembangkan program yang *multithreaded* menggunakan Java. Sebuah program *multithreaded* mengandung dua atau lebih bagian yang dapat berjalan bersamaan dan setiap bagian dapat menangani tugas yang berbeda pada saat yang sama. Hal ini membuat penggunaan secara optimal sumber daya yang tersedia, khususnya bila komputer memiliki beberapa CPU (*Central Processing Unit*).

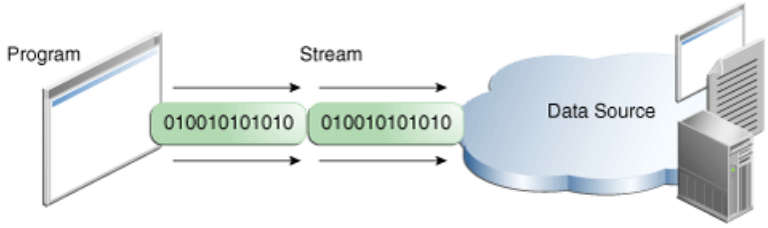
Multithreading memperluas ide *multitasking* ke aplikasi dimana dapat membagi operasi tertentu dalam satu aplikasi menjadi *thread* individu. Setiap *thread* dapat berjalan secara paralel. Sistem operasi membagi waktu proses tidak hanya di kalangan aplikasi yang berbeda, tetapi juga di antara setiap *thread* dalam aplikasi. *Multithreading* memungkinkan pengembang untuk menulis di mana beberapa kegiatan dapat dilanjutkan secara bersamaan dalam program yang sama [9].

2.6.2. *Input/Output*

Pada Java, *Input/Output* (I/O) pada *file* dan jaringan dilakukan berdasarkan aliran atau *stream*, di mana semua objek dapat melakukan perintah I/O yang sama. Sebuah aliran I/O merepresentasikan sumber masukan atau tujuan keluaran dan merupakan serangkaian data. Aliran tersebut dapat merepresentasikan berbagai macam sumber dan tujuan, termasuk *disk file*, perangkat, program, dan memori. Sebuah program menggunakan *input stream* untuk membaca data dari sumber, yaitu satu objek pada satu waktu seperti pada Gambar 2.2. Sebuah program menggunakan *output stream* untuk menulis data ke tujuan, yaitu satu objek pada satu waktu seperti pada Gambar 2.3 [10].



Gambar 2.2 *Input Stream*



Gambar 2.3 *Output Stream*

Java memiliki dua kategori untuk aliran, yaitu aliran *byte* atau *byte stream*, dan aliran karakter atau *character stream*. Program menggunakan *byte stream* untuk melakukan *input* dan *output* dari 8-*bit bytes*. Semua kelas aliran *byte* merupakan turunan dari *InputStream* dan *OutputStream* [11]. Java menyimpan nilai-nilai karakter menggunakan konvensi *Unicode*. Aliran karakter I/O secara otomatis menerjemahkan format internal dari set karakter lokal ke set karakter lokal juga. Dalam karakter lokal barat, set karakter lokal biasanya merupakan *superset 8-bit* ASCII. Semua kelas aliran karakter adalah turunan dari *Reader* dan *Writer* [12].

Java menyediakan mekanisme yang disebut serialisasi objek dimana sebuah objek dapat direpresentasikan sebagai urutan *byte* yang meliputi data objek serta informasi tentang jenis objek dan jenis data yang disimpan dalam objek. Kelas *ObjectInputStream* dan *ObjectOutputStream* adalah aliran tingkat tinggi yang berisi metode untuk serialisasi dan deserialisasi sebuah objek [13].

2.6.3. Jaringan

Pemrograman jaringan merujuk pada penulisan program yang dijalankan di beberapa perangkat atau komputer, dimana semua perangkat terhubung satu sama lain menggunakan jaringan. Paket *java.net* dari API J2SE berisi koleksi kelas dan *interface* yang menyediakan rincian komunikasi tingkat rendah.

Paket *java.net* menyediakan dukungan untuk dua protokol jaringan yang umum, yaitu:

1. TCP (*Transmission Control Protocol*), yang memungkinkan untuk komunikasi antara dua aplikasi. TCP biasanya digunakan melalui *Internet Protocol* (TCP / IP);
2. UDP (*User Datagram Protocol*), protokol dengan sedikit koneksi yang memungkinkan paket data untuk ditransmisikan antara aplikasi.

Pemrograman *Socket* adalah konsep yang paling banyak digunakan dalam jaringan. *Socket* menyediakan mekanisme komunikasi antara dua komputer menggunakan TCP. Sebuah program klien menciptakan *socket* pada akhir komunikasi dan mencoba untuk menyambungkan *socket* tersebut ke *server*. Setelah sambungan dibuat, *server* membuat objek *socket* pada akhir komunikasi. Klien dan *server* dapat berkomunikasi dengan menulis dan membaca dari *socket*.

Kelas *java.net.Socket* merepresentasikan sebuah *socket*, dan kelas *java.net.ServerSocket* menyediakan mekanisme untuk program *server* untuk mendengarkan dan membangun hubungan dengan klien. Setelah koneksi ditetapkan, komunikasi dapat terjadi menggunakan aliran I/O. Setiap *socket* memiliki *OutputStream* dan *InputStream*. *OutputStream* klien terhubung ke *InputStream server*, dan *InputStream* klien terhubung ke *OutputStream server* [14].

2.6.4. JAMA

JAMA adalah pustaka perangkat lunak atau paket aljabar linear dasar untuk Java yang dibuat oleh NIST (*National Institute of Standards and Technology*). Paket ini menyediakan kelas untuk membangun dan memanipulasi matriks. JAMA terdiri dari enam kelas Java, yaitu *Matrix*, *CholeskyDecomposition*, *LUDecomposition*, *QRDecomposition*, *SingularValueDecomposition* dan *EigenvalueDecomposition*. JAMA menyediakan berbagai operasi dasar aritmatika seperti penambahan, pengurangan, perkalian, *transpose* matriks dan norma matriks [15].

2.7. Cohen's Kappa

Penelitian yang mengukur kesepakatan antara dua atau lebih pengamat harus menyertakan sebuah statistik yang memperhitungkan fakta bahwa pengamat terkadang akan setuju atau tidak setuju karena kebetulan. Statistik *Kappa* atau koefisien *Kappa* adalah statistik yang paling umum digunakan untuk tujuan tersebut. *Kappa* bertujuan untuk memberikan pembaca pengukuran kuantitatif tentang besarnya kesepakatan antara pengamat.

Perhitungan ini didasarkan pada perbedaan antara berapa banyak kesepakatan secara aktual atau *observed agreement* dibandingkan dengan berapa banyak kesepakatan yang diperkirakan secara kebetulan atau *expected agreement* [16]. Daftar hasil pengamatan antar pengamat dapat dimasukkan ke dalam tabel seperti pada Tabel 2.1.

Tabel 2.1 Daftar Hasil Pengamatan Antar Pengamat

		Pengamat 1			
		Kategori			**Jumlah**
Pengamat 2		**1**	**2**	**3**	
Kategori	**1**	W_{ii}	W_{ii}	W_{ii}	R_i
	2	W_{ii}	W_{ii}	W_{ii}	R_i
	3	W_{ii}	W_{ii}	W_{ii}	R_i
Jumlah		C_i	C_i	C_i	N

Dalam tabel hasil pengamatan, *W* adalah jumlah nilai yang sesuai antara hasil dari pengamat 1 dan hasil dari pengamat 2 untuk setiap kategori, *R* adalah jumlah hasil setiap baris dan *C* adalah jumlah hasil setiap kolom. Untuk menghitung *observed agreement* atau kesepakatan yang diamati dapat dihitung seperti pada Persamaan 2.6. Sedangkan untuk menghitung *expected agreement* atau kesepakatan yang diharapkan dapat dihitung seperti pada Persamaan 2.7.

$$p_o = \frac{\sum_{i=0}^{n} W_{ii}}{N} \quad \textbf{(2.6)}$$

$$p_e = \sum_{i=0}^{n} \frac{R_i C_i}{N} \quad \textbf{(2.7)}$$

Untuk menghitung nilai *Kappa* digunakan perhitungan seperti pada Persamaan 2.8. Dari hasil perhitungan *Kappa* tersebut dapat dilihat interpretasi dari *Kappa* atau tingkat *reliability* seperti pada Tabel 2.2.

$$K = \frac{p_o - p_e}{1 - p_e} \quad \textbf{(2.8)}$$

Tabel 2.2 Daftar Interpretasi Kappa

Kappa	**Kesepakatan**
< 0	Kurang
0,01 – 0,20	Sedikit
0,21 – 0,40	Cukup
0,41 – 0,60	Sedang
0,61 – 0,80	Baik
0,81 – 0,99	Sangat Baik

BAB III
ANALISIS DAN PERANCANGAN SISTEM

Bab ini menjelaskan pembahasan mengenai analisis dan perancangan perangkat lunak yang akan dibangun. Analisis menjelaskan tentang analisa permasalahan yang diangkat dalam pembuatan Tugas Akhir, deskripsi umum dari sistem yang dibuat dan spesifikasi kebutuhan perangkat lunak. Perancangan sistem menjelaskan tentang rancangan arsitektur dari perangkat lunak dan rancangan antarmuka yang dibuat.

3.1. Analisis

Tahap ini dibagi menjadi beberapa bagian, antara lain analisa permasalahan, deskripsi umum sistem, dan spesifikasi kebutuhan perangkat lunak.

3.1.1. Analisis Permasalahan

Permasalahan utama dalam pembuatan Tugas Akhir ini adalah bagaimana mengatur komunikasi atau pesan antar pengembang perangkat lunak dalam sebuah *plugin* Eclipse dengan mengklasifikasikan pesan yang dikirimkan pengembang berdasarkan keahlian pengembang lainnya dalam sebuah tim proyek perangkat lunak.

Latent Semantic Indexing digunakan untuk mengklasifikasikan pesan terhadap deskripsi keahlian, yaitu *library* dan *component*, dari pengembang lain dalam sebuah tim proyek perangkat lunak.

3.1.2. Deskripsi Umum Sistem

Sistem yang dibuat adalah sebuah aplikasi *client* dan *server*. *Server* berupa aplikasi *desktop* yang dijalankan pada sebuah komputer *server*. Sedangkan *client* berupa kakas bantu atau sebuah *plugin* untuk IDE Eclipse. Masukan untuk aplikasi ini adalah teks pesan dari pengguna. Kakas ini dapat mengklasifikasikan pesan yang dikirimkan pengembang berdasarkan keahlian pengembang lain dalam sebuah tim proyek perangkat lunak pada lingkungan

kerja yang sama. Keahlian pengembang didasarkan pada *library* dan *component* yang dipilih oleh setiap pengembang dalam sebuah tim. Keluaran dari aplikasi ini adalah pesan dikirimkan kepada pengguna dengan keahlian terkait dengan pesan yang dikirimkan.

Dengan kakas bantu ini, diharapkan pengembang sebagai pengguna dapat melakukan komunikasi dengan pengembang lainnya. Selain itu juga diharapkan dapat mengatur komunikasi antara pengembang dan membantu pengembang mencari pengembang lain dengan keahlian yang sesuai dengan pesan yang diajukan sehingga pesan dapat disampaikan kepada pengembang yang sesuai.

3.1.3. Spesifikasi Kebutuhan Perangkat Lunak

Bagian ini berisi kebutuhan perangkat lunak yang terdiri dari aktor dan kasus penggunaan yang diuraikan dalam bentuk diagram kasus penggunaan, kelas analisis, diagram urutan dan diagram aktivitas. Setiap diagram menjelaskan perilaku dari perangkat lunak. Selain itu juga terdapat kebutuhan perangkat lunak dalam sistem yang meliputi kebutuhan fungsional dan kebutuhan non fungsional. Pada bab ini akan dijelaskan spesifikasi pada masing-masing kebutuhan fungsional dan kebutuhan non fungsional.

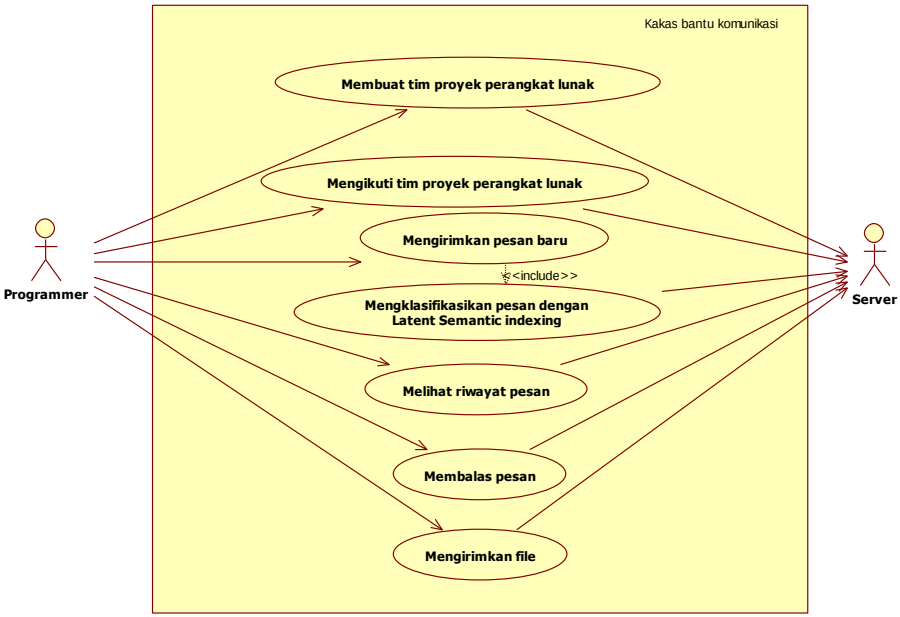
3.1.3.1. Aktor

Aktor adalah entitas yang terlibat dan berinteraksi dengan perangkat lunak. Aktor atau pengguna dari perangkat lunak ini adalah pengembang perangkat lunak atau *programmer* yang bekerja secara tim. Selain pengembang sebagai aktor, terdapat juga *server* perangkat lunak sebagai entitas luar.

3.1.3.2. Kasus Penggunaan

Berdasarkan analisa spesifikasi kebutuhan sistem dan analisa aktor dari sistem dibuat kasus penggunaan sistem. Kasus penggunaan digambarkan dalam sebuah diagram kasus penggunaan. Diagram kasus penggunaan dapat dilihat pada

Gambar 3.1. Untuk penjelasan dari setiap kasus penggunaan dapat dilihat pada Tabel 3.1.



Gambar 3.1 Diagram Kasus Penggunaan

Tabel 3.1 Daftar Kode Diagram Kasus Penggunaan

Kode Kasus Penggunaan	Nama
UC-0001	Membuat tim proyek perangkat lunak
UC-0002	Mengikuti tim proyek perangkat lunak
UC-0003	Mengirimkan pesan baru
UC-0004	Melihat riwayat pesan
UC-0005	Membalas pesan
UC-0006	Mengirimkan *file*

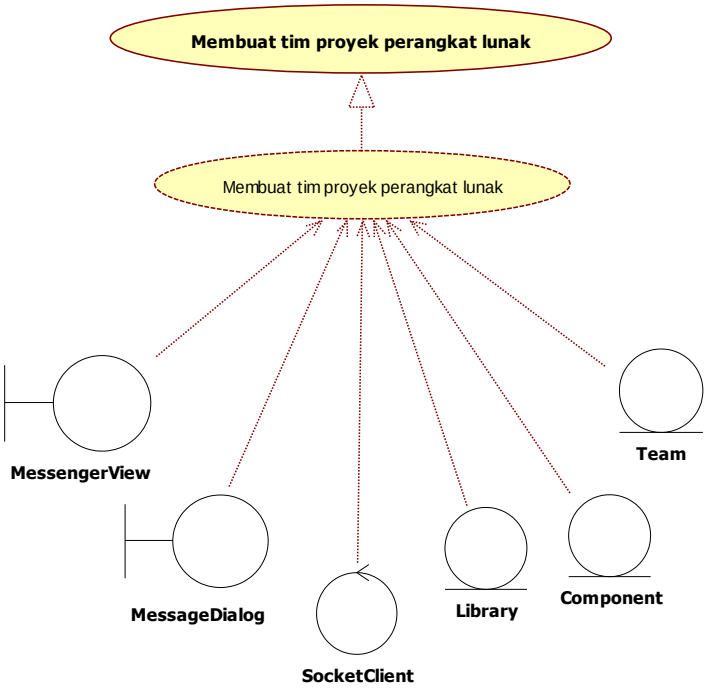
3.1.3.2.1 Membuat Tim Proyek Perangkat Lunak

Kasus penggunaan ini mendeskripsikan bagaimana pengguna dapat membuat tim proyek perangkat lunak dalam kakas bantu komunikasi. Pada kasus penggunaan ini, sistem akan menerima *input* berupa nama tim proyek perangkat lunak, *library* dan *component* yang digunakan pada proyek perangkat lunak beserta deskripsinya. Setelah itu, sistem akan mengirimkan masukan kepada *server* untuk kemudian dimasukkan ke dalam *database*. Spesifikasi kasus penggunaan dapat dilihat pada Tabel 3.2. kelas analisis, diagram aktivitas dan diagram urutan dari kasus penggunaan dapat dilihat pada Gambar 3.2, Gambar 3.3 dan Gambar 3.4.

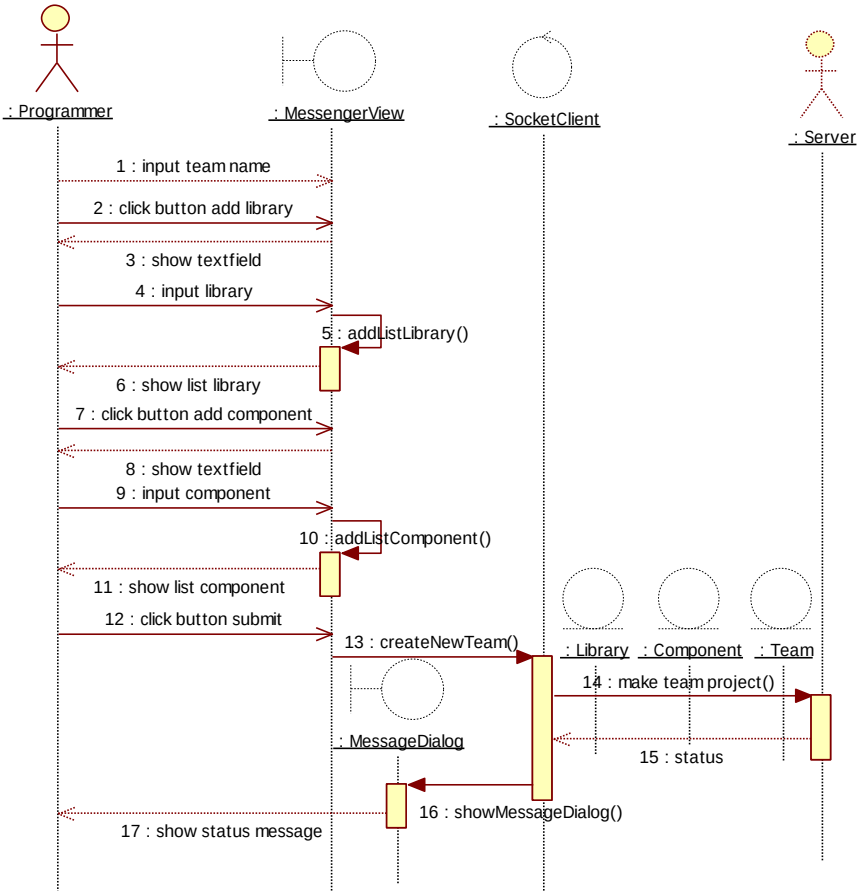
Tabel 3.2 Spesifikasi Kasus Penggunaan Membuat Tim Proyek Perangkat Lunak

Nama	Membuat tim proyek perangkat lunak
Kode	UC-0001
Deskripsi	Membuat sebuah tim proyek perangkat lunak dengan deskripsi setiap *library* dan *component* yang digunakan dalam proyek perangkat lunak.
Tipe	Fungsional
Pemicu	Pengguna menekan tombol *submit* untuk membuat tim proyek perangkat lunak.
Aktor	Pengguna
Kondisi Awal	Pengguna berada pada halaman *tab Team* dan belum menekan tombol *Submit.*
Aliran: - **Kejadian Normal**	1. Pengguna mengetikkan nama tim. 2. Pengguna menekan tombol *Add Library.* 3. Sistem menampilkan *textfield* untuk nama dan deskripsi *library.* 4. Pengguna mengetikkan nama dan deskripsi *library.* 5. Pengguna menekan tombol *Add.* 6. Sistem menambahkan *library* pada daftar *library.* 7. Pengguna menekan tombol *Add Component.* 8. Sistem menampilkan *textfield* untuk nama dan deskripsi *component.*

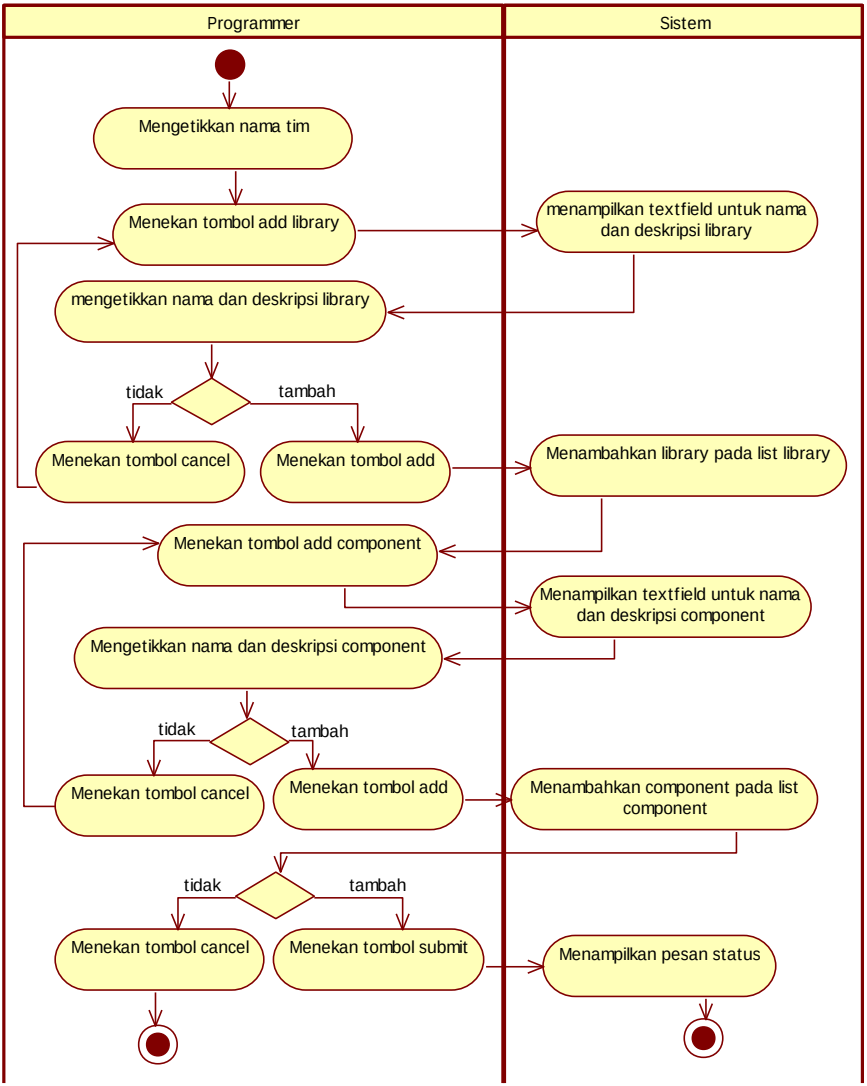
	9. Pengguna mengetikkan nama dan deskripsi *component.* 10. Pengguna menekan tombol *Add.* 11. Sistem menambahkan *component* pada daftar *component.* 12. Pengguna menekan tombol *Submit.* 13. Sistem mengirimkan detil tim baru kepada *server.* 14. Sistem menampilkan pesan bahwa tim telah berhasil dibuat.
- **Kejadian Alternatif**	2a. **Pengguna menghapus *library*** 2a.1. Pengguna memilih *library* pada daftar *library.* 2a.2. Pengguna menekan tombol *Delete.* 2a.3. Sistem menghapus *library* dari daftar *library.* 7a. **Pengguna menghapus *component*** 7a.1. Pengguna memilih *component* dari daftar *component.* 7a.2. Pengguna menekan tombol *Delete.* 7a.3. Sistem menghapus *component* dari daftar *component.*
- **Kejadian Penge-cualian**	5a. **Pengguna tidak menambahkan *library*** 5a.1. Pengguna menekan tombol *Cancel.* 5a.2. Sistem tidak menambahkan *library* pada daftar *library.* 10a. **Pengguna tidak menambahkan *component*** 10a.1. Pengguna menekan tombol *Cancel.* 10a.2. Sistem tidak menambahkan *component* pada daftar *component.* 12a. **Pengguna tidak membuat tim** 12a.1. Pengguna menekan tombol *Cancel.* 12a.2. Sistem menghapus semua *library* pada daftar *library.* 12a.3. Sistem menghapus semua *component* pada daftar *component.* 14a. **Tim tidak berhasil dibuat** 14a.1. Sistem menampilkan pesan bahwa tim tidak berhasil dibuat.
Kondisi Akhir	Sistem menampilkan pesan bahwa tim berhasil dibuat.



Gambar 3.2 Kelas Analisis Membuat Tim Proyek Perangkat Lunak



Gambar 3.3 Diagram Urutan Membuat Tim Proyek Perangkat Lunak



Gambar 3.4 Diagram Aktivitas Membuat Tim Proyek Perangkat Lunak

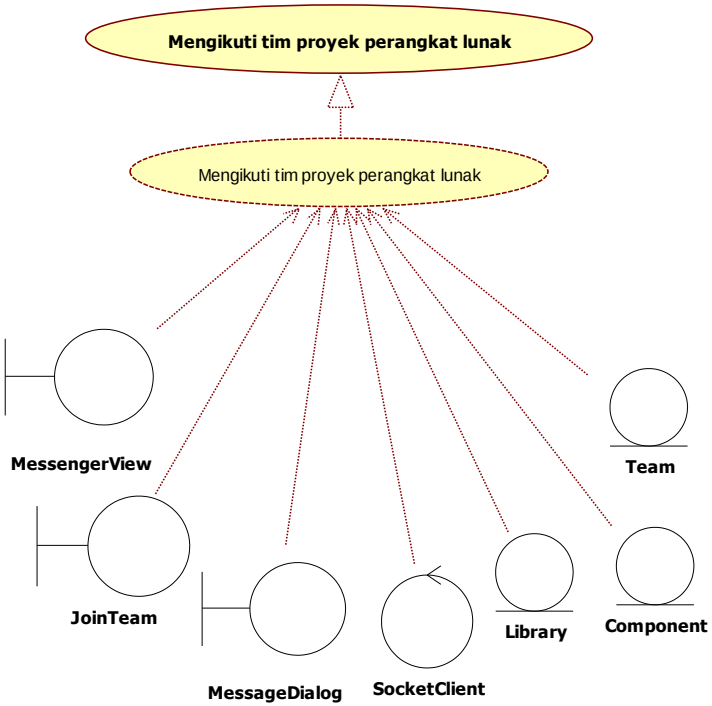
3.1.3.2.2 Mengikuti Tim Proyek Perangkat Lunak

Kasus penggunaan ini mendeskripsikan bagaimana pengguna dapat mengikuti tim proyek yang ada dalam aplikasi. Pada kasus penggunaan ini, sistem akan menerima masukan berupa nama tim proyek perangkat lunak, *library* dan *component* yang dipilih. Setelah itu, sistem akan mengirimkan masukan kepada *server* untuk dimasukkan ke *database*. Spesifikasi kasus penggunaan dapat dilihat pada Tabel 3.3. Kelas analisis, diagram aktivitas, dan diagram urutan dari kasus penggunaan dapat dilihat pada Gambar 3.5, Gambar 3.6 dan Gambar 3.7.

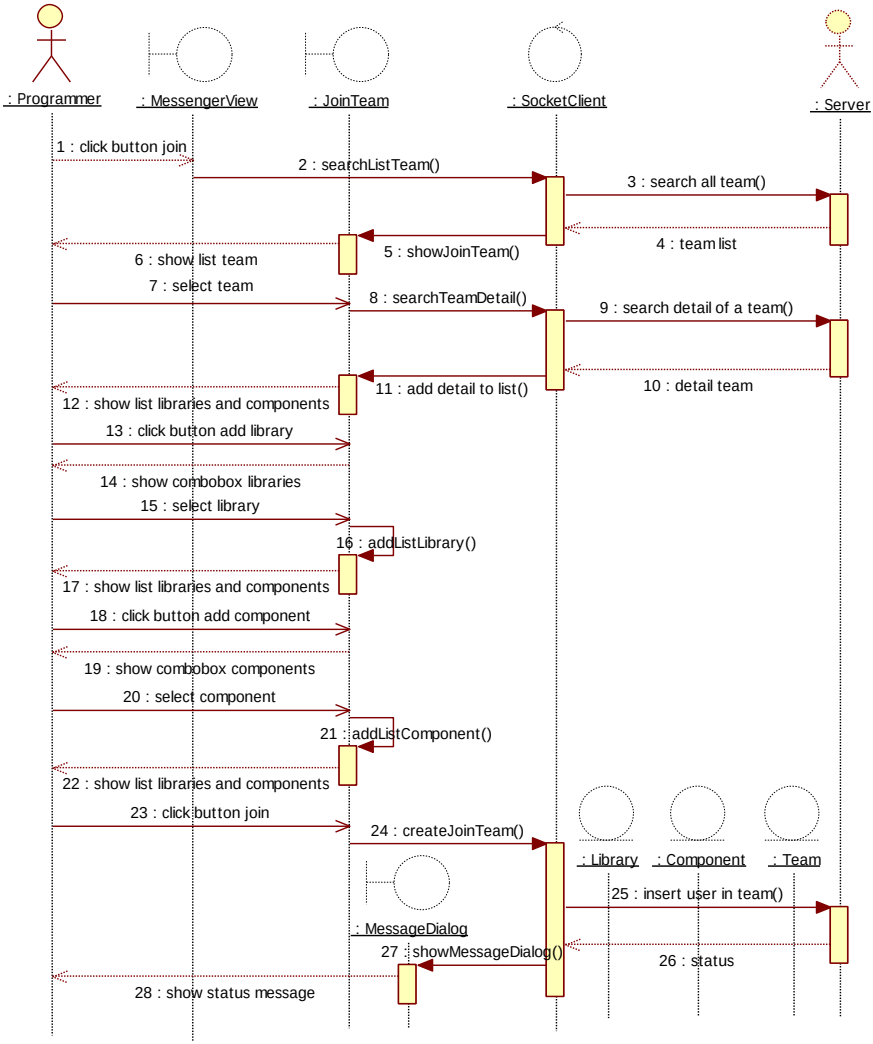
Tabel 3.3 Spesifikasi Kasus Penggunaan Mengikuti Tim Proyek Perangkat Lunak

Nama	Mengikuti tim proyek perangkat lunak
Kode	UC-0002
Deskripsi	Mengikuti sebuah tim proyek perangkat lunak yang sudah tersedia dengan menyertakan *library* dan *component*.
Tipe	Fungsional
Pemicu	Pengguna menekan tombol *Join* untuk mengikuti tim proyek perangkat lunak.
Aktor	Pengguna
Kondisi Awal	Pengguna berada pada halaman *tab Team* dan belum menekan tombol *Join.*
Aliran: - **Kejadian Normal**	1. Pengguna menekan tombol *Join.* 2. Sistem mengirimkan permintaan daftar tim kepada *server.* 3. Sistem menampilkan antarmuka jendela *JoinTeam.* 4. Pengguna memilih tim pada *combobox* tim. 5. Pengguna menekan tombol *Select.* 6. Sistem mengirimkan permintaan detil tim yang dipilih kepada *server.* 7. Sistem menampilkan daftar *library* dan *component.* 8. Pengguna menekan tombol *Add Library.* 9. Sistem menampilkan antarmuka *panel library.* 10. Pengguna memilih *library* pada *combobox library.* 11. Pengguna menekan tombol *Add.*

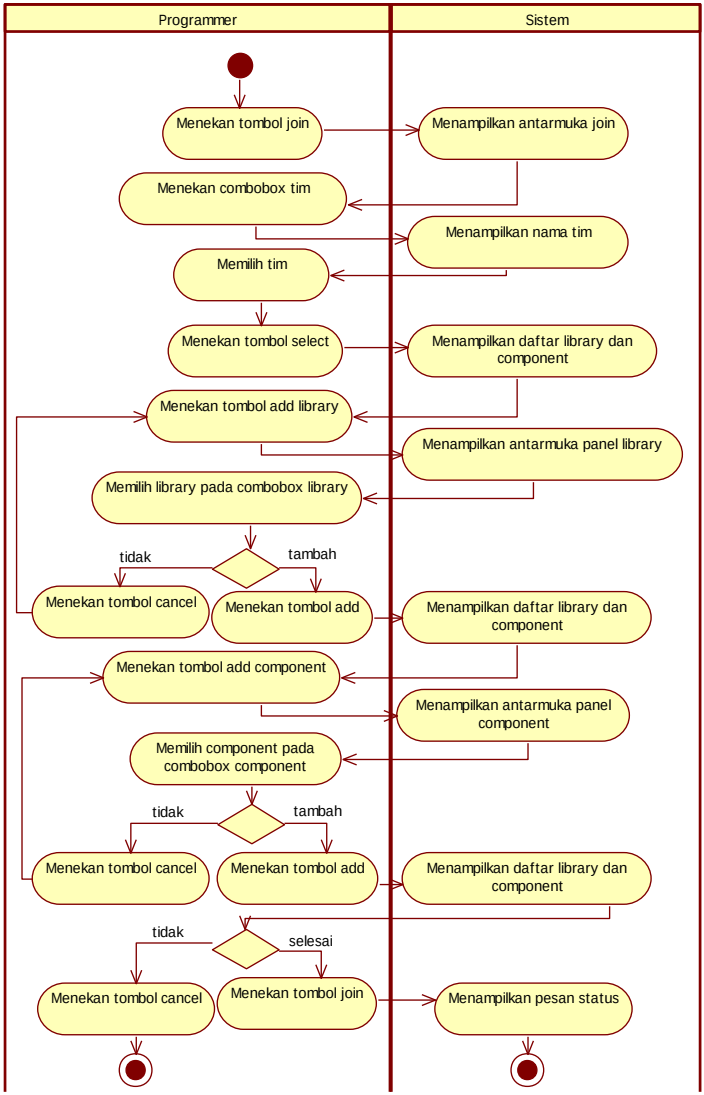
	12. Sistem menampilkan daftar *library* dan *component.* 13. Pengguna menekan tombol *Add Component.* 14. Sistem menampilkan antarmuka *panel component.* 15. Pengguna memilih *component* pada *combobox component.* 16. Pengguna menekan tombol *Add.* 17. Sistem menampilkan daftar *library* dan *component* pada antarmuka jendela *JoinTeam.* 18. Pengguna menekan tombol *Join.* 19. Sistem mengirimkan detil tim yang dipilih kepada *server.* 20. Sistem menampilkan pesan bahwa pengguna berhasil mengikuti tim.
- **Kejadian Alternatif**	8a. **Pengguna menghapus *library*** 8a.1. Pengguna memilih *library* dari daftar *library.* 8a.2. Pengguna menekan tombol *Delete.* 8a.3. Sistem menghapus *library* dari daftar *library* 13a. **Pengguna menghapus *component*** 13a.1. Pengguna memilih *component* dari daftar *component.* 13a.2. Pengguna menekan tombol *Delete.* 13a.3. Sistem menghapus *component* dari daftar *component.*
- **Kejadian Penge-cualian**	11a. **Pengguna tidak menambahkan *library*** 11a.1. Pengguna menekan tombol *Cancel.* 11a.2. Sistem tidak menambahkan *library* pada daftar *library.* 16a. **Pengguna tidak menambahkan *component*** 16a.1. Pengguna menekan tombol *Cancel.* 16a.2. Sistem tidak menambahkan *component* pada daftar *component.* 18a. **Pengguna tidak mengikuti tim** 18a.1. Pengguna menekan tombol *Cancel.* 18a.2. Sistem menghapus semua *library* pada daftar *library.* 18a.3. Sistem menghapus semua *component* pada daftar *component.*
Kondisi Akhir	Sistem menampilkan pesan bahwa pengguna berhasil mengikuti tim.



Gambar 3.5 Kelas Analisis Mengikuti Tim Proyek Perangkat Lunak



Gambar 3.6 Diagram Urutan Mengikuti Tim Proyek Perangkat Lunak



Gambar 3.7 Diagram Aktivitas Mengikuti Tim Proyek Perangkat Lunak

3.1.3.2.3 Mengirimkan Pesan Baru

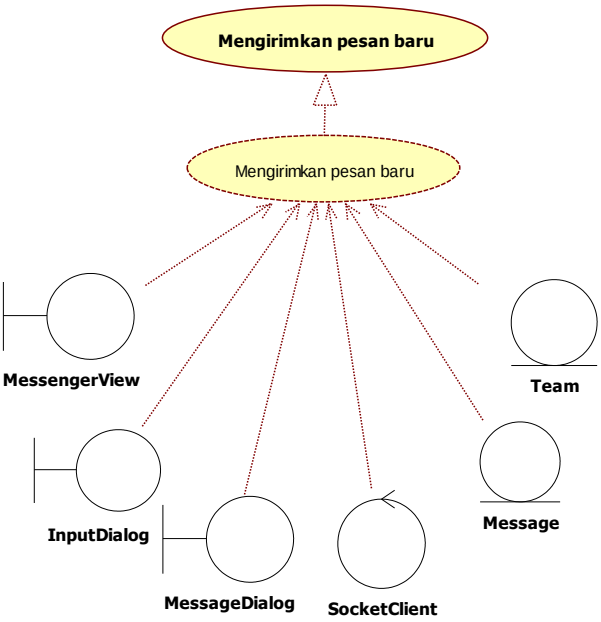
Kasus penggunaan ini mendeskripsikan bagaimana pengguna menggunakan kakas bantu untuk mengirimkan pesan kepada pengguna lainnya. Pada kasus penggunaan ini, sistem akan menerima masukan berupa sebuah teks pesan dan tim proyek perangkat lunak. Setelah itu, sistem akan mengirimkan masukan kepada *server* untuk melakukan klasifikasi pesan menggunakan metode *Latent Semantic Indexing*. Kemudian pesan akan dikirimkan kepada pengguna lain sesuai dengan hasil klasifikasi.

Spesifikasi kasus penggunaan dapat dilihat pada Tabel 3.4. Kelas analisis, diagram aktivitas dan diagram urutan dari kasus penggunaan dapat dilihat pada Gambar 3.8, Gambar 3.9 dan Gambar 3.10.

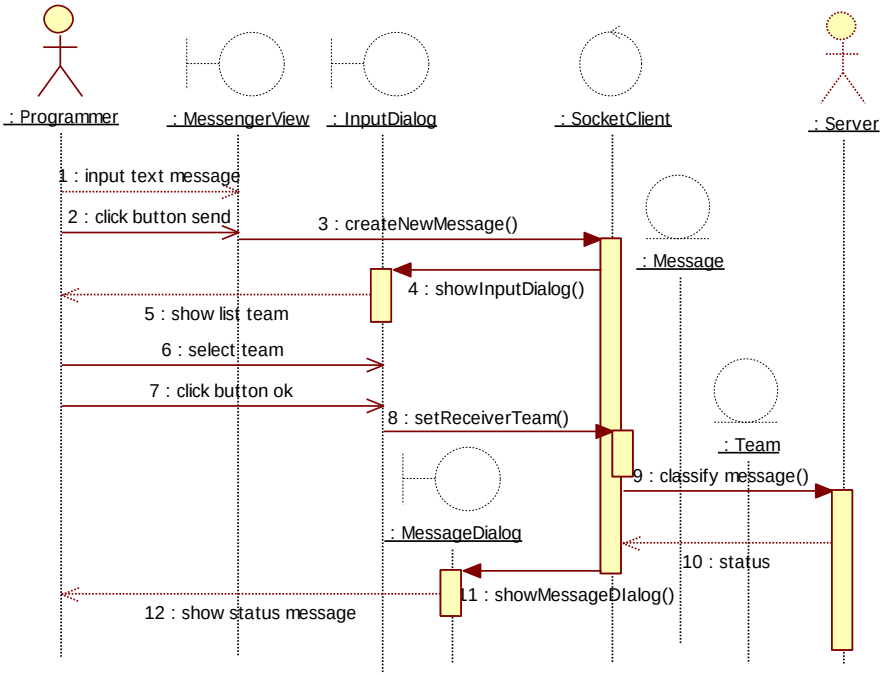
Tabel 3.4 Spesifikasi Kasus Penggunaan Mengirimkan Pesan Baru

Nama	Mengirimkan pesan baru
Kode	UC-0003
Deskripsi	Mengirimkan pesan kepada pengguna lain dalam sebuah tim proyek perangkat lunak yang sama.
Tipe	Fungsional
Pemicu	Pengguna menekan tombol *Send* untuk mengirimkan pesan.
Aktor	Pengguna
Kondisi Awal	Pengguna berada pada halaman *tab Message* dan belum terdapat teks pesan yang diketikkan oleh pengguna.
Aliran: - **Kejadian Normal**	1. Pengguna mengetikkan teks pesan. 2. Pengguna menekan tombol *Send.* 3. Sistem menampilkan *input dialog* tim. 4. Pengguna memilih tim. 5. Pengguna menekan tombol *Ok.* 6. Sistem mengirimkan pesan dan nama tim kepada *server* untuk diklasifikasikan. 7. Sistem menampilkan pesan bahwa pesan telah terkirim.
- **Kejadian Alternatif**	Tidak ada.

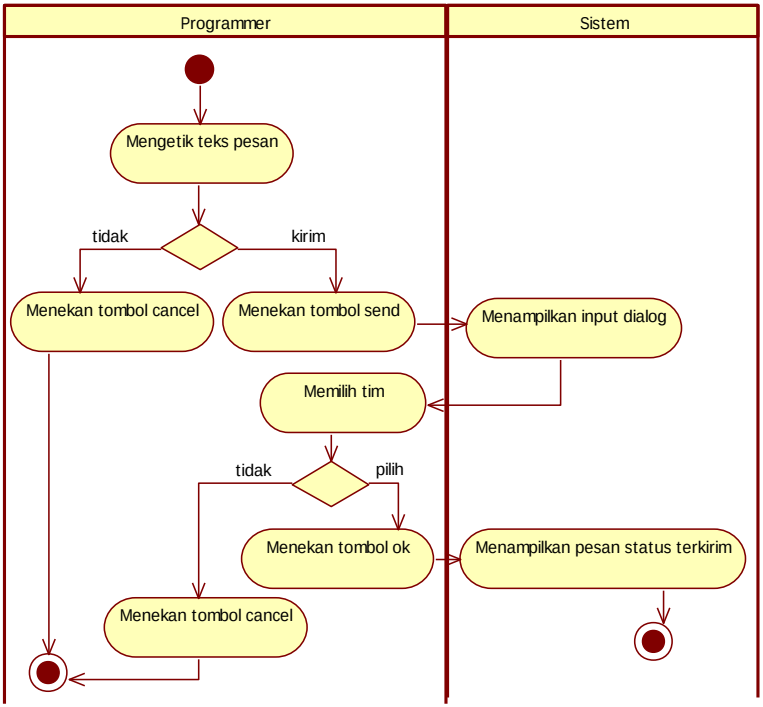
- **Kejadian Penge-cualian**	2a. **Pengguna tidak mengirimkan pesan** 2a.1 Pengguna menekan tombol *Cancel.* 2a.2 Sistem menghapus teks pesan. 5a. **Pengguna tidak memilih tim** 5a.1 Pengguna menekan tombol *Cancel.* 5a.2 Sistem menghapus teks pesan. 7a **Pesan tidak terkirim** 7a.1 Sistem menampilkan pesan bahwa pesan tidak terkirim.
Kondisi Akhir	Sistem menampilkan pesan bahwa pesan telah dikirimkan.



Gambar 3.8 Kelas Analisis Mengirimkan Pesan Baru



Gambar 3.9 Diagram Urutan Mengirimkan Pesan Baru



Gambar 3.10 Diagram Aktivitas Mengirimkan Pesan Baru

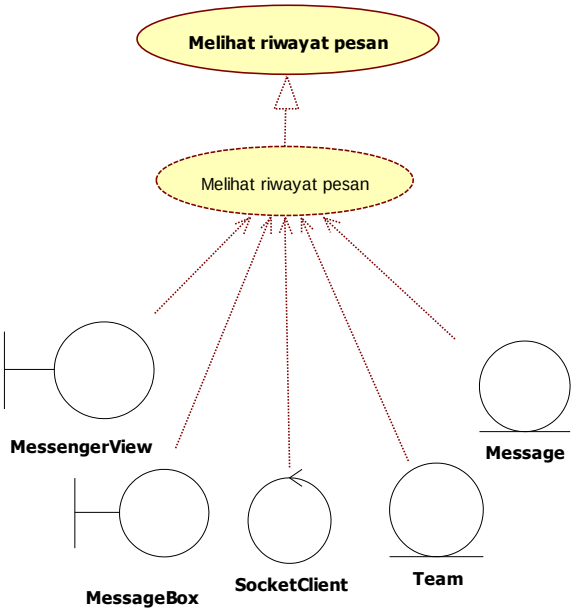
3.1.3.2.4 Melihat Riwayat Pesan

Kasus penggunaan ini mendeskripsikan bagaimana pengguna menggunakan kakas bantu untuk melihat riwayat pesan dari pengguna dengan pengguna lain. Pada kasus penggunaan ini, sistem akan menerima masukan berupa nama tim dan nama anggota tim yang dipilih. Setelah itu, sistem akan mengirimkan masukan kepada *server* untuk kemudian mencari riwayat pesan pengguna dengan anggota tim yang dipilih. Spesifikasi kasus penggunaan dapat dilihat pada Tabel 3.5. Kelas analisis, diagram aktivitas dan diagram urutan dari kasus penggunaan dapat dilihat pada Gambar 3.11, Gambar 3.12 dan Gambar 3.13.

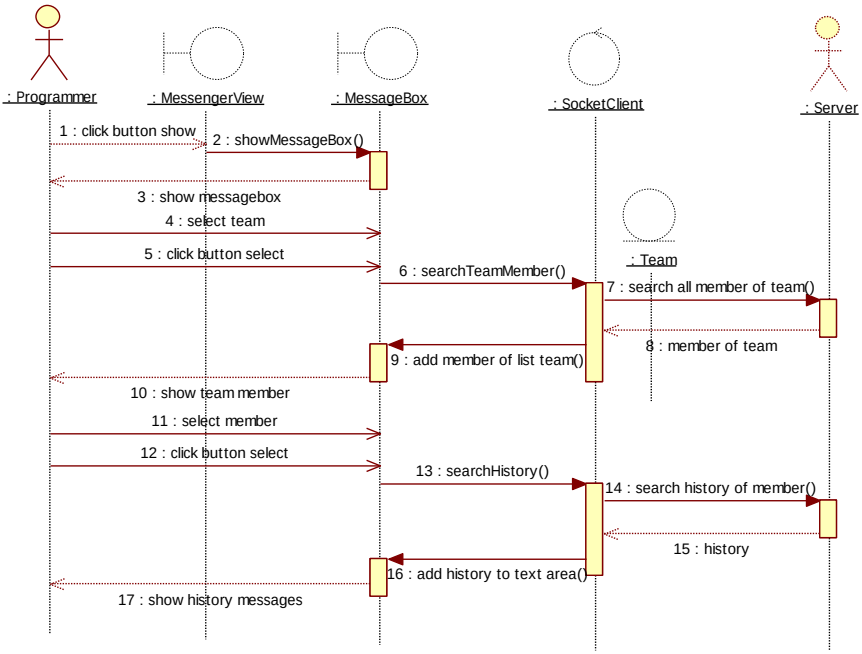
Tabel 3.5 Spesifikasi Kasus Penggunaan Melihat Riwayat Pesan

Nama	Melihat riwayat pesan
Kode	UC-0004
Deskripsi	Melihat riwayat pesan antara pengguna dengan pengguna lain dalam sebuah tim proyek perangkat lunak yang sama.
Tipe	Fungsional
Pemicu	Pengguna menekan tombol *Select* untuk melihat riwayat pesan.
Aktor	Pengguna
Kondisi Awal	Terdapat riwayat pesan antara pengguna dengan pengguna lain dan pengguna berada pada halaman *tab Message.*
Aliran: - **Kejadian Normal**	1. Pengguna menekan tombol *Show.* 2. Sistem menampilkan antarmuka jendela *Message Box.* 3. Pengguna memilih tim pada *combobox* tim. 4. Pengguna menekan tombol *Select.* 5. Sistem mengirimkan permintaan daftar anggota tim yang dipilih kepada *server*. 6. Sistem menampilkan daftar pengguna pada *combobox* anggota tim. 7. Pengguna memilih anggota tim pada *combobox* anggota. 8. Pengguna menekan tombol *Select.* 9. Sistem mengirimkan permintaan riwayat pesan pengguna dengan anggota tim yang dipilih kepada *server.* 10. Sistem menampilkan riwayat pesan pengguna.
- **Kejadian Alternatif**	Tidak ada.
- **Kejadian Penge-cualian**	4a. **Pengguna tidak memilih tim** 4a.1. Pengguna menekan tombol *Cancel.* 4a.2. Sistem menutup antarmuka jendela *MessageBox.* 7a. **Pengguna tidak memilih Pengguna** 7a.1. Pengguna menekan tombol *Cancel.* 7a.2. Sistem tidak menampilkan riwayat pesan.

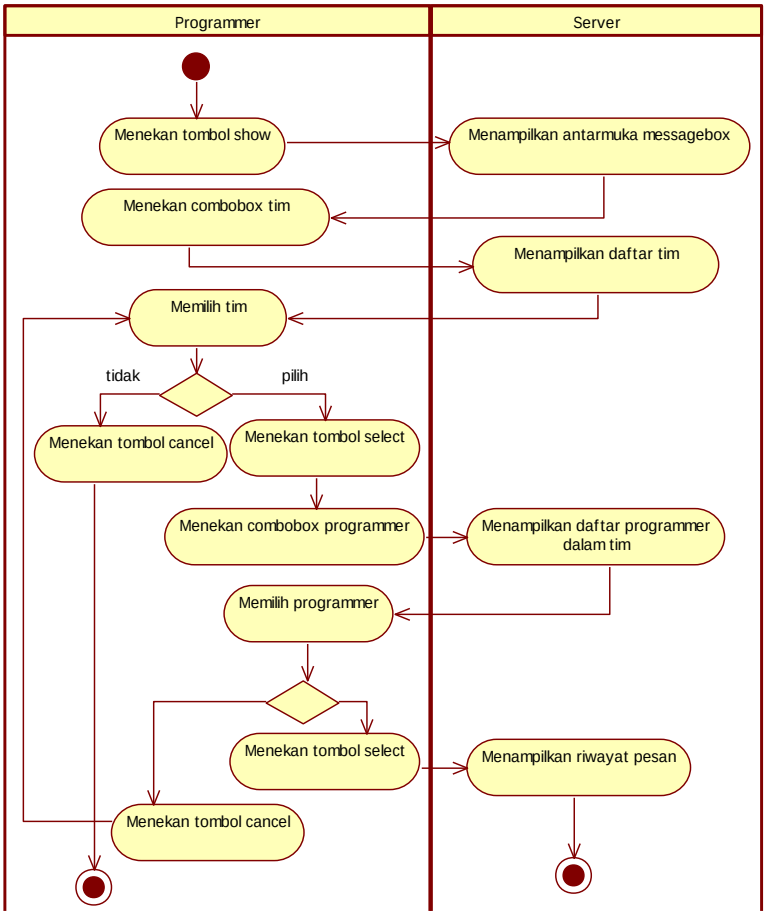
Kondisi Akhir	Sistem menampilkan riwayat pesan pengguna dengan pengguna lain.



Gambar 3.11 Kelas Analisis Melihat Riwayat Pesan



Gambar 3.12 Diagram Urutan Melihat Riwayat Pesan



Gambar 3.13 Diagram Aktivitas Melihat Riwayat Pesan

3.1.3.2.5 Membalas Pesan

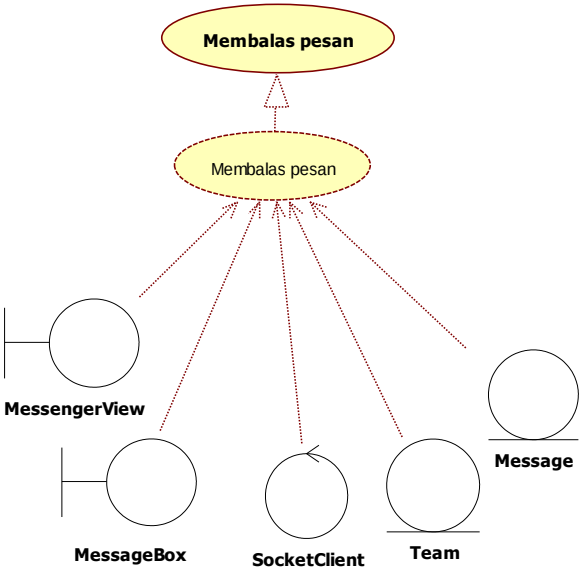
Kasus penggunaan ini mendeskripsikan bagaimana pengguna menggunakan kakas bantu untuk membalas pesan yang telah diterima dari pengguna lain. Pada kasus penggunaan ini, sistem akan menerima masukan berupa teks pesan yang akan

dikirimkan. Setelah itu, sistem akan mengirimkan masukan kepada *server* untuk kemudian dikirimkan kepada penerima pesan. Spesifikasi kasus penggunaan dapat dilihat pada Tabel 3.6. Kelas analisis, diagram aktivitas dan diagram urutan dari kasus penggunaan dapat dilihat pada Gambar 3.14, Gambar 3.15 dan Gambar 3.16.

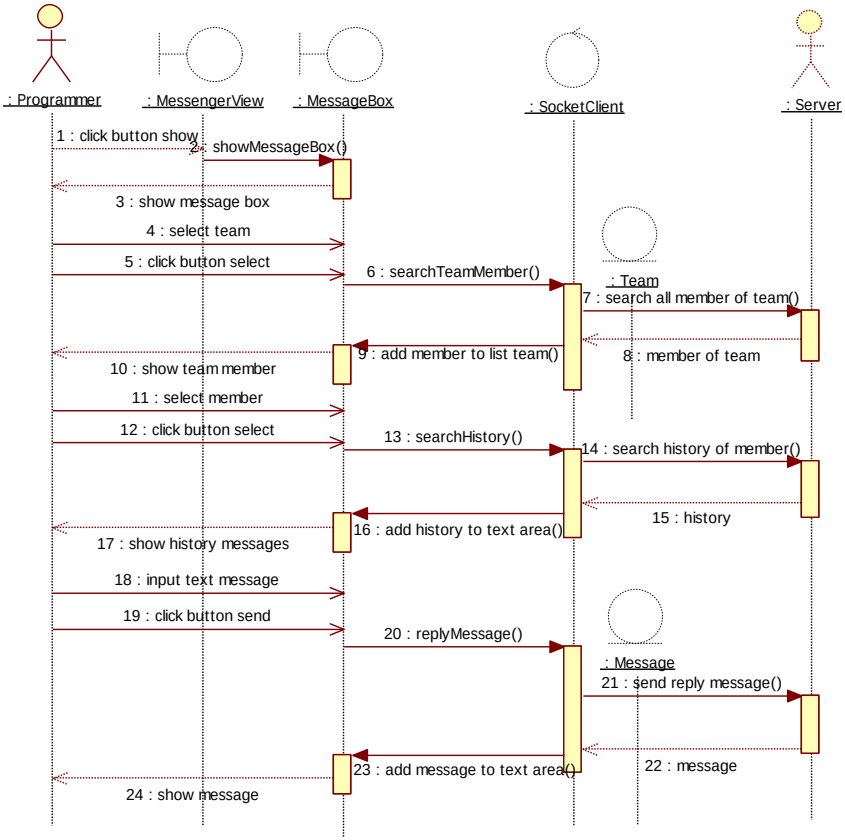
Tabel 3.6 Spesifikasi Kasus Penggunaan Membalas Pesan

Nama	Membalas pesan
Kode	UC-0005
Deskripsi	Membalas pesan yang diterima dari pengguna lain dalam sebuah tim proyek perangkat lunak yang sama.
Tipe	Fungsional
Pemicu	Pengguna menekan tombol *Send* untuk membalas pesan.
Aktor	Pengguna
Kondisi Awal	Terdapat pesan yang telah diterima oleh pengguna dan pengguna berada pada halaman *tab Message.*
Aliran: - **Kejadian Normal**	1. Pengguna menekan tombol *Show.* 2. Sistem menampilkan antarmuka jendela *Message Box.* 3. Pengguna memilih tim pada *combobox* tim. 4. Pengguna menekan tombol *Select.* 5. Sistem mengirimkan permintaan daftar anggota tim yang dipilih kepada *server*. 6. Sistem menampilkan daftar pengguna pada *combobox* anggota tim. 7. Pengguna memilih anggota tim pada *combobox* anggota. 8. Pengguna menekan tombol *Select.* 9. Sistem mengirimkan permintaan riwayat pesan pengguna dengan anggota tim yang dipilih kepada *server.* 10. Sistem menampilkan riwayat pesan pengguna. 11. Pengguna mengetikkan teks pesan. 12. Pengguna menekan tombol *Send.* 13. Sistem mengirimkan pesan kepada *server*. 14. Sistem menampilkan pesan yang telah dikirimkan pada riwayat pesan.

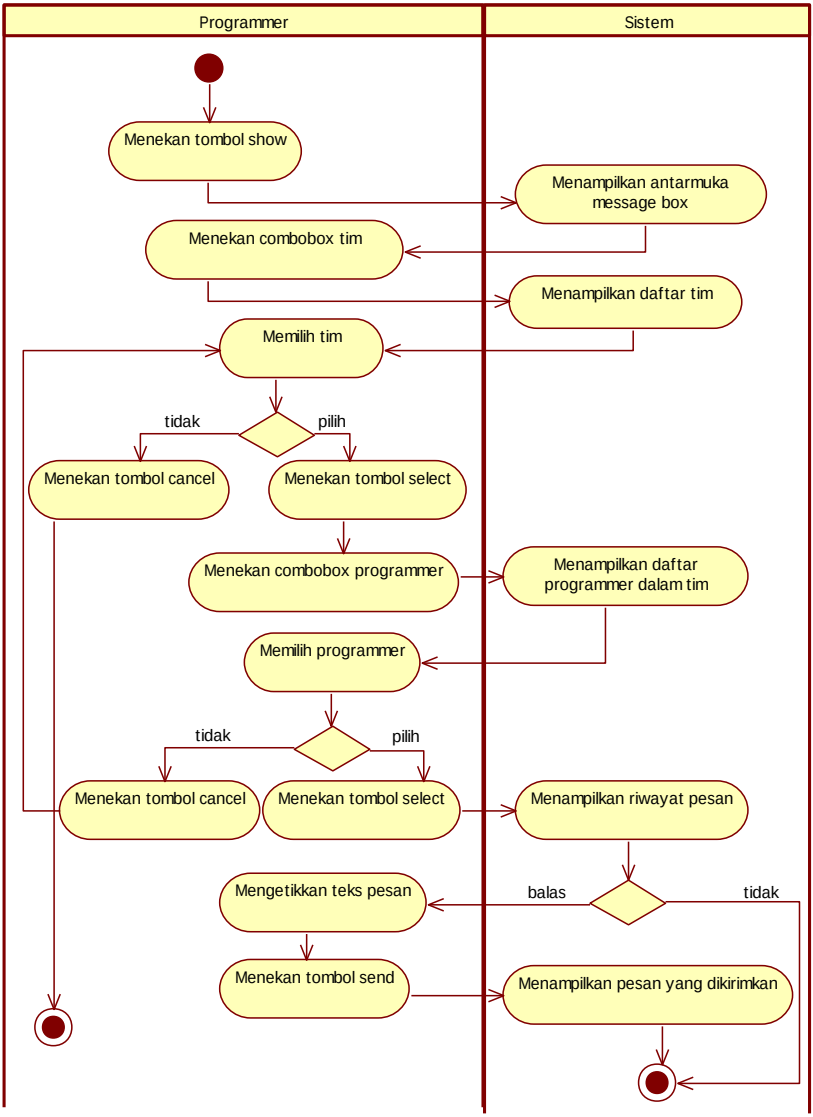
- **Kejadian Alternatif**	Tidak ada.
- **Kejadian Penge-cualian**	4a. **Pengguna tidak memilih tim** 4a.1. Pengguna menekan tombol *Cancel.* 4a.2. Sistem menutup antarmuka jendela *MessageBox.* 7a. **Pengguna tidak memilih Pengguna** 7a.1. Pengguna menekan tombol *Cancel.* 7a.2. Sistem tidak menampilkan riwayat pesan.
Kondisi Akhir	Sistem menampilkan pesan yang telah dikirimkan pada riwayat pesan.



Gambar 3.14 Kelas Analisis Membalas Pesan



Gambar 3.15 Diagram Urutan Membalas Pesan



Gambar 3.16 Diagram Aktivitas Membalas Pesan

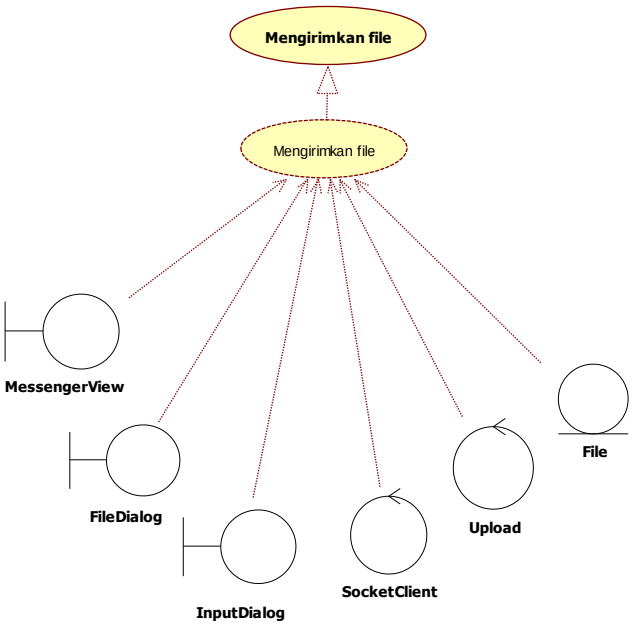
3.1.3.2.6 Mengirimkan *File*

Kasus Penggunaan ini mendeskripsikan bagaimana pengguna menggunakan kakas bantu untuk mengirimkan *file* kepada pengguna lainnya. Pada kasus penggunaan ini, sistem akan menerima masukan berupa *file* yang akan dikirimkan dan nama pengguna penerima. Setelah itu, sistem akan mengirimkan masukan kepada *server* untuk kemudian mengirimkan *file* kepada pengguna tujuan. Spesifikasi kasus penggunaan dapat dilihat pada Tabel 3.7. Kelas analisis, diagram aktivitas dan diagram urutan dari kasus penggunaan dapat dilihat pada Gambar 3.17, Gambar 3.18 dan Gambar 3.19.

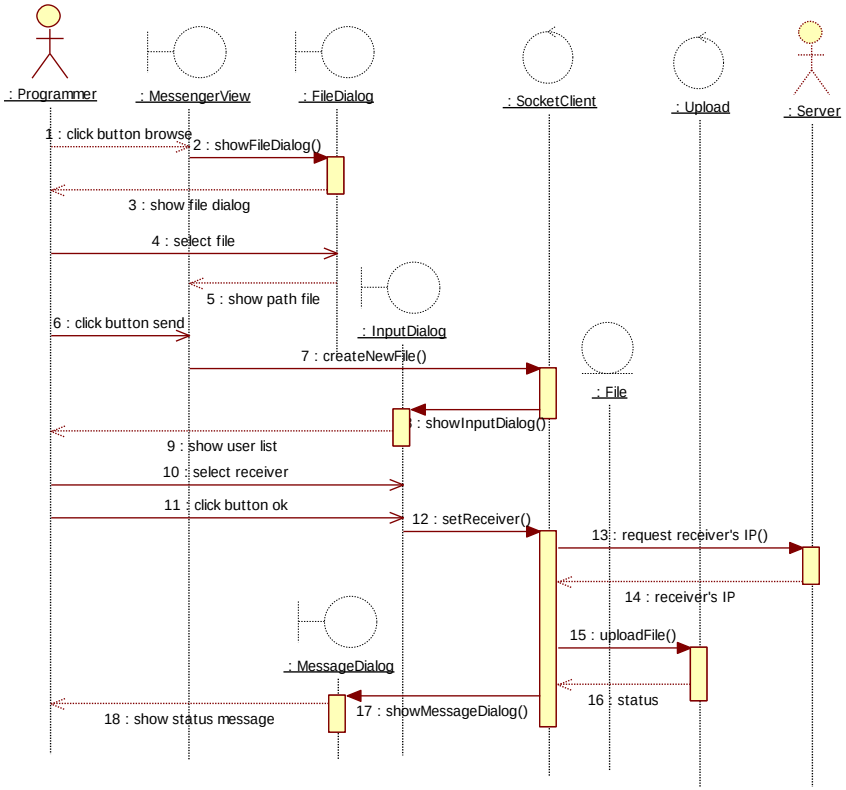
Tabel 3.7 Spesifikasi Kasus Penggunaan Mengirimkan *File*

Nama	Mengirimkan *file*
Kode	UC-0006
Deskripsi	Mengirimkan *file* kepada pengguna lain.
Tipe	Fungsional
Pemicu	Pengguna menekan tombol *Send* untuk mengirimkan *file.*
Aktor	Pengguna
Kondisi Awal	Pengguna berada pada halaman *tab File* dan belum memilih *file.*
Aliran: - **Kejadian Normal**	1. Pengguna menekan tombol *Browse.* 2. Sistem menampilkan *file dialog.* 3. Pengguna memilih *file.* 4. Pengguna menekan tombol *Save.* 5. Sistem menampilkan alamat dari *file* yang dipilih. 6. Pengguna menekan tombol *Send.* 7. Sistem menampilkan *input dialog* daftar pengguna. 8. Pengguna memilih penerima. 9. Pengguna menekan tombol *Ok.* 10. Sistem mengirimkan permintaan untuk mendapatkan alamat IP dari penerima kepada *server*. 11. Sistem mengirimkan *file* kepada penerima dengan alamat IP yang diterima dari *server*.

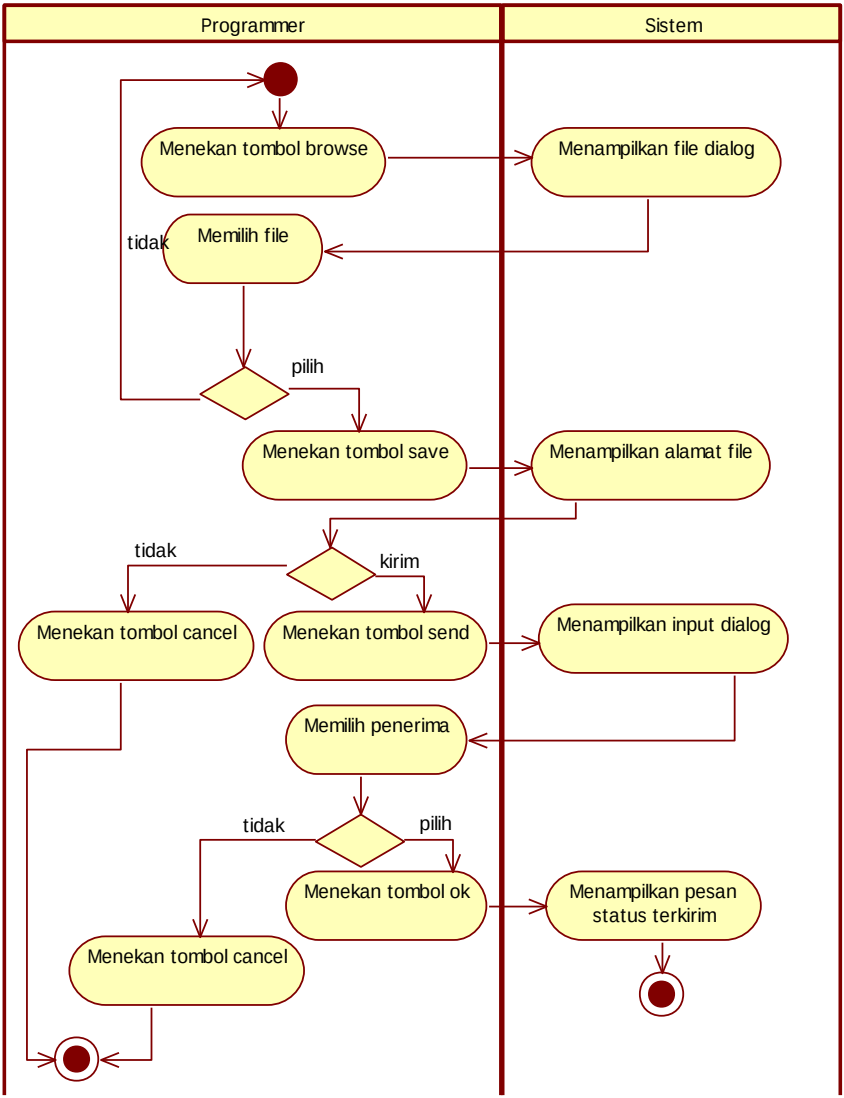
	12. Sistem menampilkan pesan bahwa *file* telah terkirim.
- **Kejadidan Alternatif**	Tidak ada.
- **Kejadian Penge-cualian**	4a. **Pengguna tidak memilih *file*** 4a.1. Sistem menutup *file dialog.* 6a. **Pengguna tidak mengirimkan *file*** 6a.1. Pengguna menekan tombol *Cancel.* 6a.2. Sistem tidak mengirimkan *file.* 9a. **Pengguna tidak memilih penerima** 9a.1. Pengguna menekan tombol *Cancel.* 9a.2. Sistem tidak mengirimkan *file.* 10a. ***File* tidak terkirim** 10a.1. Sistem menampilkan pesan bahwa *file* tidak terkirim.
Kondisi Akhir	Sistem menampilkan pesan bahwa *file* telah terkirim.



Gambar 3.17 Kelas Analisis Mengirimkan File



Gambar 3.18 Diagram Urutan Mengirimkan *File*



Gambar 3.19 Diagram Aktivitas Mengirimkan *File*

3.1.3.3. Kebutuhan Fungsional

Kebutuhan fungsional berisi proses yang harus dimiliki sistem. Kebutuhan fungsional mendefinisikan layanan yang harus disediakan dan reaksi sistem terhadap masukan atau pada situasi tertentu. Daftar kebutuhan fungsional dapat dilihat pada Tabel 3.8.

Tabel 3.8 Daftar Kebutuhan Fungsional Perangkat Lunak

Kode Kebutuhan	Kebutuhan Fungsional	Deskripsi
F-0001	Membuat tim proyek perangkat lunak	Pengguna dapat membuat tim proyek perangkat lunak dengan *library* dan *component* yang digunakan beserta deskripsinya
F-0002	Mengikuti tim proyek perangkat lunak	Pengguna dapat mengikuti salah satu tim proyek perangkat lunak yang tersedia dengan memilih dan menyertakan *library* serta *component* yang digunakan dalam proyek perangkat lunak.
F-0003	Mengirimkan pesan baru	Pengguna dapat mengirimkan sebuah teks pesan kepada pengguna lain dalam sebuah tim proyek perangkat lunak.
F-0004	Melihat riwayat pesan	Pengguna dapat melihat riwayat pesan dengan pengguna lain dalam sebuah tim proyek perangkat lunak.
F-0005	Membalas pesan	Pengguna dapat melihat pesan yang diterima dari pengguna lain dalam sebuah tim proyek perangkat lunak dan dapat membalas pesan tersebut.
F-0006	Mengirimkan *file*	Pengguna dapat mengirimkan sebuah *file* yang dipilih kepada pengguna lainnya.

3.1.3.4. Kebutuhan Non Fungsional

Kebutuhan non fungsional adalah kebutuhan yang berkaitan dengan kendala pada pelayanan atau fungsi sistem. Kebutuhan non fungsional memberikan batasan pada kebutuhan fungsional. Daftar kebutuhan non fungsional dapat dilihat pada Tabel 3.9.

Tabel 3.9 Daftar Kebutuhan Non Fungsional Perangkat Lunak

Kode Kebutuhan	Kebutuhan Non Fungsional	Deskripsi
NF-0001	Koneksi	Aplikasi hanya dapat diakses oleh pengguna apabila pengguna sudah terhubung dengan jaringan lokal atau berada pada jaringan yang sama dengan *server* perangkat lunak.
NF-0002	Keamanan	Aplikasi hanya dapat diakses oleh pengguna yang sudah mendaftar pada aplikasi untuk mendapatkan akun. Dan setiap pengguna melakukan *login* dengan akun milik sendiri.

3.2. Perancangan Sistem

Perancangan perangkat lunak dibagi menjadi beberapa bagian yaitu perancangan arsitektur dan perancangan antarmuka.

3.2.1. Perancangan Arsitektur

Perancangan arsitektur menjelaskan rancangan dari arsitektur sistem dan kelas yang digunakan untuk membangun perangkat lunak. Pada subbab ini, hubungan dan perilaku antar kelas digambarkan dengan lebih jelas. Perangkat lunak ini terdiri dari dua arsitektur, yaitu arsitektur *client* dan arsitektur *server*. Pada arsitektur *client* terdapat tiga jenis kelas yaitu kelas *boundary*, kelas *control*, dan kelas *entity*. Pada arsitektur *server* terdapat dua jenis kelas yaitu kelas *control*, dan kelas *entity*. Subbab ini dibagi menjadi dua bagian, yaitu arsitektur *client* dan arsitektur *server.*

3.2.1.1. Arsitektur *Client*

Diagram kelas untuk arsitektur *client* dapat dilihat pada Gambar 3.20. Terdapat sepuluh kelas penyusun arsitektur *client* yang terdiri dari tiga kelas *boundary*, tiga kelas *control* dan empat kelas *entity*. Kelas *boundary* terdiri dari kelas *MessengerView, JoinTeam,* dan *MessageBox.* Kelas *boundary* hanya berhubungan dengan kelas *control*. *MessengerView* adalah kelas dengan *extend*

ViewPart yang merupakan bawaan dari *library org.eclipse.ui.part*. Kelas ini digunakan untuk membuat tampilan *view* pada aplikasi RCP (*Rich Client Platform*). Pada kelas ini terdapat antarmuka untuk *login* bagi pengguna ke aplikasi, *register* pengguna baru ke aplikasi, membuat tim proyek perangkat lunak dan mengirimkan pesan serta mengirimkan *file.*

JoinTeam adalah kelas antarmuka untuk kasus penggunaan mengikuti tim proyek perangkat lunak. Pada antarmuka ini pengguna dapat memilih tim proyek, melihat daftar *library* dan *component* yang dipilih dan memilih *library* serta *component* untuk dihapus. Pada kelas ini terdapat dua buah *panel* tambahan untuk antarmuka, yaitu *panel library* dan *panel component*. *Panel library* menampilkan antarmuka untuk memilih *library* dan menampilkan deskripsi dari *library* yang dipilih. *Panel component* menampilkan antarmuka untuk memilih *component* dan menampilkan deskripsi dari *component* yang dipilih.

MessageBox adalah kelas antarmuka untuk kasus penggunaan membalas pesan. Kelas ini menampilkan antarmuka untuk menampilkan riwayat pesan berdasarkan tim dan anggota tim serta dapat membalas pesan dari pengguna lain.

Kelas *control* terdiri dari tiga kelas yaitu kelas *SocketClient, Upload,* dan *Download.* Kelas *SocketClient* merupakan kelas yang memiliki hubungan dengan semua kelas *boundary* dan kelas *entity.* Kelas ini mengimplementasikan antarmuka *Runnable* dari *thread*. Pada kelas ini terdapat method *run* yaitu metode yang dijalankan dengan *thread* lain. Metode ini memiliki perulangan *internal* yang akan terus dipanggil hingga tidak digunakan lagi. Metode *run* berfungsi sebagai penerima *streaming* data dari *server*. Selain itu juga terdapat metode *sendServer* yang berfungsi sebagai pengirim atau *streaming* data ke *server*.

Kelas *Upload* dan *Download* digunakan pada saat pengguna mengirimkan sebuah *file*. Kelas *Upload* digunakan pengirim untuk mengirimkan *file* sedangkan kelas *Download* digunakan penerima untuk mengunduh *file* yang diterima.

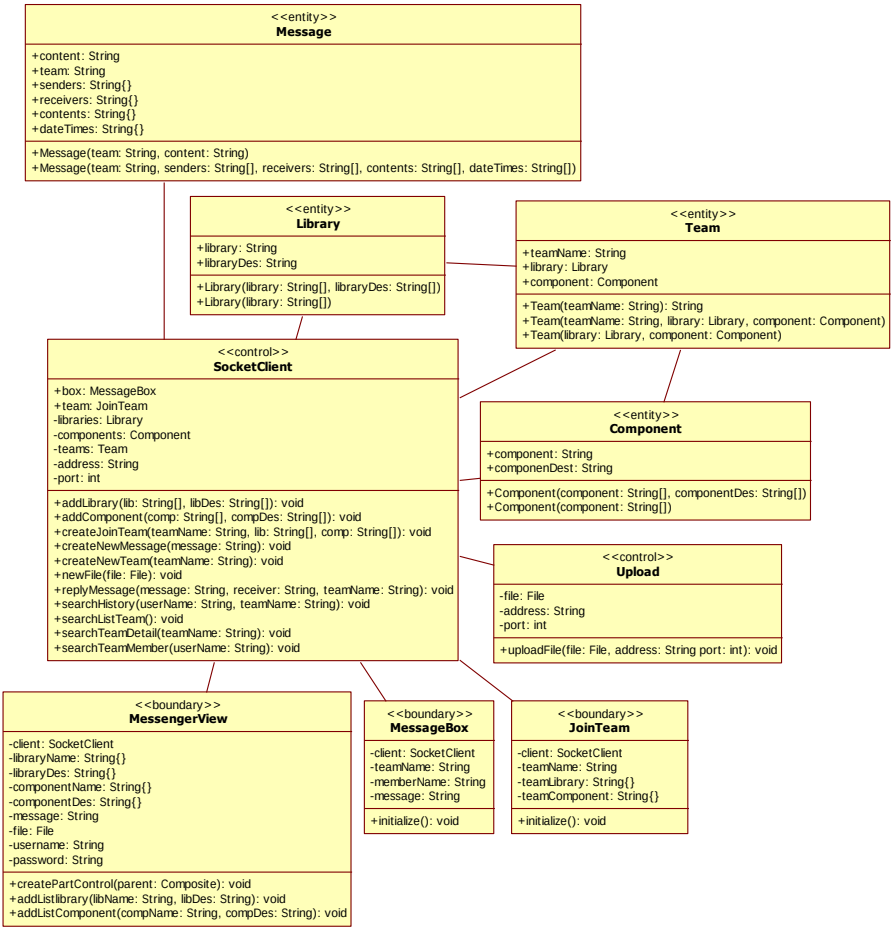
Kelas *entity* terdiri dari empat kelas yaitu kelas *Library, Component, Team,* dan *Message*. Kelas *entity* hanya dapat diakses oleh kelas *control* karena kelas ini hanya berhubungan dengan kelas *control.* Kelas *Library* berisi atribut untuk nama-nama *library* beserta deskripsinya. Kelas *Component* berisi atribut untuk nama-nama *component* beserta deskripsinya. Kelas *Team* berisi atribut untuk nama tim proyek perangkat lunak beserta *library* dan *component.* Kelas *message* berisi atribut pesan yang dikirimkan pengguna, pengirim pesan, penerima pesan dan waktu saat pesan dikirimkan.

3.2.1.2. Arsitektur *Server*

Diagram kelas untuk arsitektur *server* dapat dilihat pada Gambar 3.21. Terdapat tujuh kelas penyusun arsitektur *server* yang terdiri dari tiga kelas *control* dan empat kelas *entity*.

Kelas *control* terdiri dari tiga kelas yaitu kelas *ThreadServer, SocketServer* dan *LatentSemanticIndexing*. Kelas *ThreadServer* merupakan kelas turunan dari *java.lang.Thread* yang memiliki semua metode untuk membuat dan menjalankan *thread.* Metode paling penting adalah *run*, yang bisa dibeban-lebihkan untuk melakukan tugas yang dibutuhkan. Atau dengan kata lain *run* adalah metode yang akan dijalankan bersamaan dengan *thread* lain. Metode *run* berfungsi sebagai penerima *streaming* data dari *client*. Selain itu juga terdapat metode *sendClient* yang berfungsi sebagai pengirim atau *streaming* data ke *client*.

Kelas *SocketServer* merupakan kelas yang mengimplementasikan *Runnable* dari *thread*. Pada kelas ini, menerima *client* yang mengirimkan permintaan koneksi kepada *server. Client* tersebut kemudian dibuatkan *thread* dengan tipe *ThreadServer.* Di kelas ini juga terdapat metode *handleClient* yang berfungsi untuk memproses data yang diterima dari *client* sesuai dengan tipe, seperti: *login, register*, membuat tim, mengikuti tim, dan mengirimkan pesan. Di kelas ini juga berisi metode-metode untuk manipulasi data dari *database* MySQL.

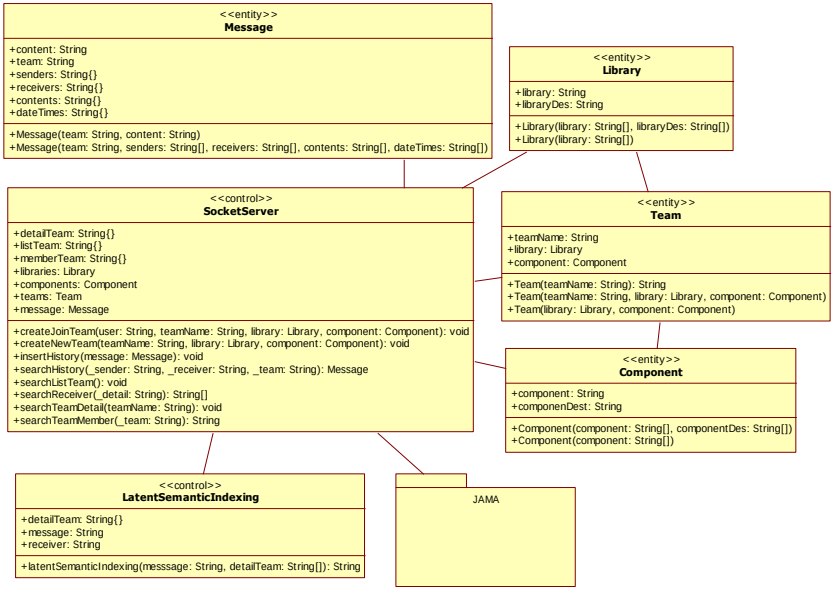


Gambar 3.20 Diagram Kelas Arsitektur *Client*

Kelas *LatentSemanticIndexing* merupakan kelas yang berfungsi untuk melakukan klasifikasi pesan yang diterima oleh *server*. Dalam kelas ini pesan yang diterima dari *client* diuraikan kedalam sebuah matriks frekuensi yang berisi bobot dari setiap kata dan dokumen. Dokumen yang digunakan dalam kelas ini adalah

deskripsi dari setiap *library* dan *component* sebuah tim proyek dimana pesan tersebut berhubungan dengan tim tersebut. Digunakan metode *cosine similarity* untuk menemukan kesamaan antara pesan dengan dokumen. Dari kesamaan tersebut sistem akan menentukan pengguna yang memiliki keahlian terkait.

Kelas *entity* terdiri dari lima kelas yaitu kelas *Library, Component, Team,* dan *Message*. Isi dari setiap kelas tersebut sama dengan kelas *entity* pada arsitektur *client*.

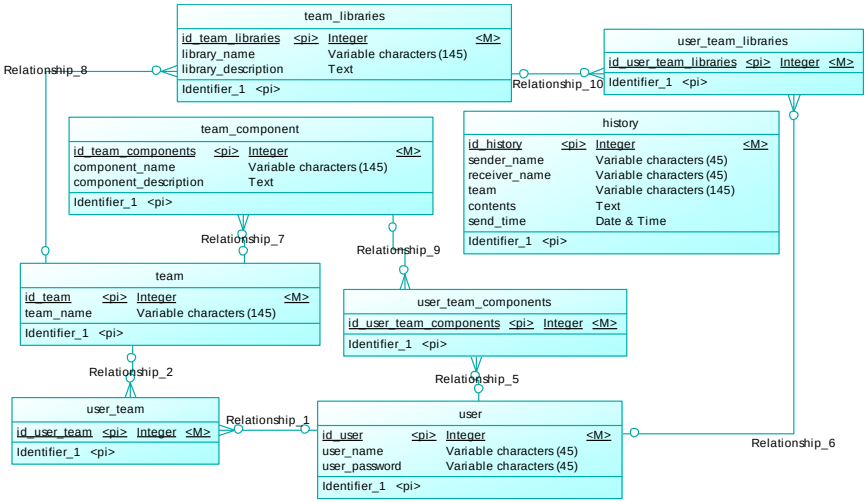


Gambar 3.21 Diagram Kelas Arsitektur *Server*

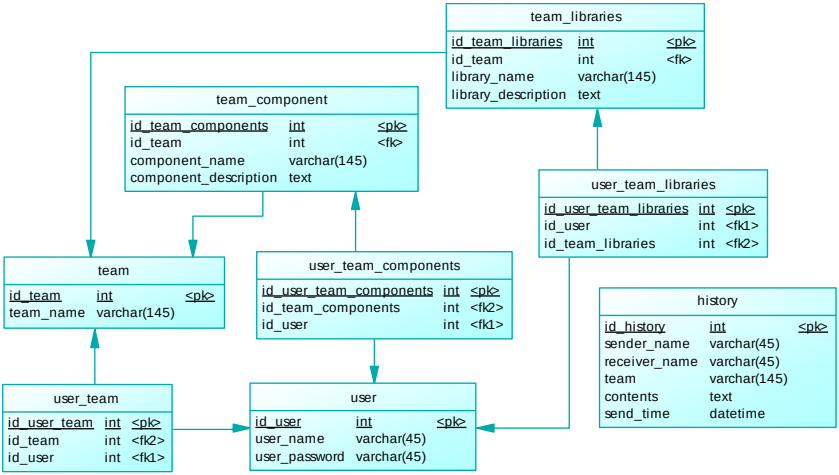
3.2.1.2.1 Perancangan Basis Data

Perangkat lunak yang digunakan untuk mengelola basis data disebut *Database Management System* (DBMS). DBMS yang digunakan dalam aplikasi ini adalah MySQL. Ilustrasi perancangan basis data digambarkan menggunakan CDM (*Conceptual Data Model*) dapat dilihat pada Gambar 3.22 dan PDM (*Physical Data*

Model) dapat dilihat pada Gambar 3.33. Spesifikasi basis data sistem dijelaskan pada Tabel 3.10.



Gambar 3.22 CDM Basis Data Aplikasi



Gambar 3.23 PDM Basis Data Aplikasi

Tabel 3.10 Spesifikasi Basis Data Sistem

No	Tabel	Atribut	Tipe Data	Fungsi
1	user	id_user	Int	Menyimpan data pengguna
		user_name	Varchar (45)	
		user_password	Varchar (45)	
2	team	id_team	Int	Menyimpan tim proyek perangkat lunak
		team_name	Varchar (145)	
3	team_ libraries	id_team_ libraries	Int	Menyimpan data *library* dari sebuah tim proyek perangkat lunak
		id_team	Int	
		library_name	Varchar (145)	
		library_ description	Text	
4	team_ component	id_team_ component	Int	Menyimpan data *component* dari sebuah tim proyek perangkat lunak
		id_team	Int	
		component_ name	Varchar (145)	
		component_ description	Text	
5	user_team	id_user_team	Int	Menyimpan data tim setiap pengguna
		id_team	Int	
		id_user	Int	
6	user_team_ libraries	id_user_team_ libraries	Int	Menyimpan *library* pengguna dalam sebuah tim
		id_user	Int	
		id_team_ libraries	Int	
7	user_team_ components	id_user_team_ component	Int	Menyimpan *component* pengguna dalam sebuah tim
		id_user	Int	
		id_team_ component	Int	
8	history	id_history	Int	Menyimpan riwayat pesan yang
		Sender_name	Varchar (45)	
		receiver_name	Varchar (45)	

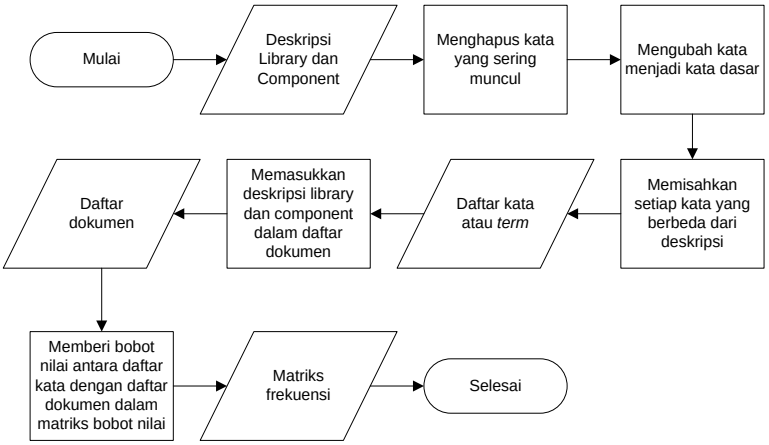
No	Tabel	Atribut	Tipe Data	Fungsi
		team	Varchar (145)	dikirimkan pengguna
		contents	Text	
		send_time	Datetime	

3.2.1.2.2 Perancangan Klasifikasi Pesan

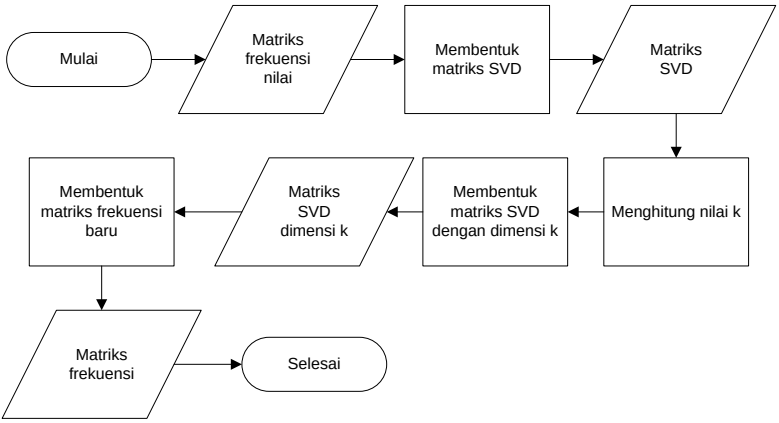
Bagian ini menjelaskan proses klasifikasi pesan yang dikirimkan pengguna pada aplikasi yang dikembangkan. Untuk mengklasifikasikan pesan dari pengguna, diimplementasikan pada kelas *LatentSemanticIndexing*. Klasifikasi pesan menggunakan metode *Latent Semantic Indexing* sebagai dasar teori. Masukan dari klasifikasi ini adalah teks pesan dan deskripsi *library* atau *component* dari tim yang dipilih pengguna pada saat mengirimkan pesan. Setiap deskripsi *library* dan *component* direpresentasikan sebagai sebuah dokumen. Dari dokumen tersebut dibuat juga daftar kata. Daftar kata dibentuk dari semua dokumen dengan memisahkan setiap kata yang berbeda. Kemudian daftar kata tersebut dibandingkan dengan daftar kata yang sering muncul dalam dokumen atau *stopwords*. Setelah itu, dilakukan proses *stemming* atau proses mengubah suatu kata bentukan menjadi kata dasar. Kemudian dibuat daftar nilai frekuensi dimana setiap baris merepresentasikan kata dari daftar kata dan setiap kolom merepresentasikan dokumen. Setiap kata diberi bobot nilai yang dibandingkan dengan semua dokumen. Apabila sebuah kata yang dibandingkan terdapat dalam sebuah dokumen maka diberi nilai 1 dan 0 jika tidak terdapat dalam dokumen. Bobot nilai tersebut kemudian direpersentasikan ke dalam matriks frekuensi. Diagram alir untuk membentuk matriks frekuensi dapat dilihat seperti pada Gambar 3.24.

Matriks frekuensi yang telah dibentuk kemudian digunakan untuk membentuk matriks *Singular Value Decomposition*, yaitu matriks U, matriks σ dan matriks V. Setelah itu dihitung nilai k untuk mengurangi dimensi ketiga matriks tersebut. Kemudian dibentuk lagi matriks U, matriks σ dan matriks V dengan pengurangan dimensi sebanyak k untuk membuat matriks

frekuensi yang baru. Diagram alir untuk membentuk matriks frekuensi baru dapat dilihat seperti pada Gambar 3.25.

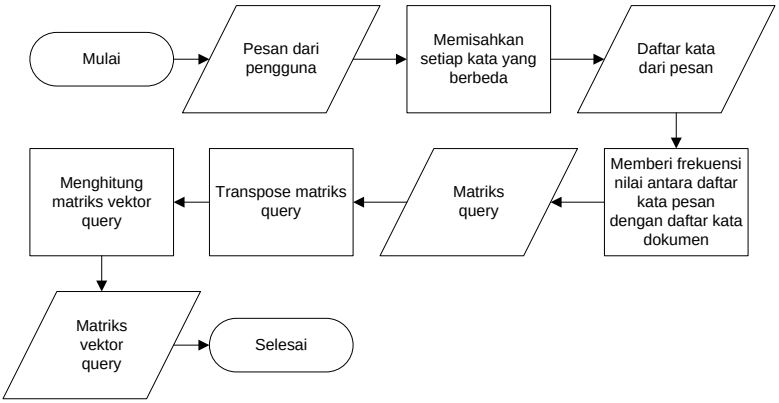


Gambar 3.24 Diagram Alir Membentuk Matriks Frekuensi



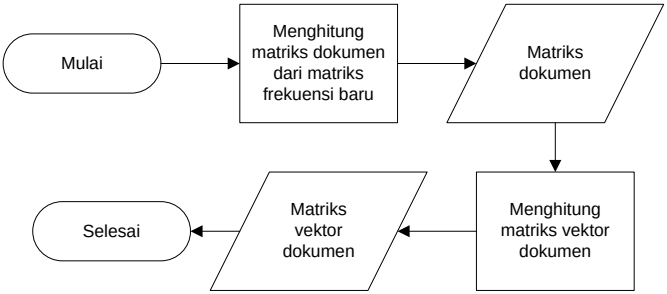
Gambar 3.25 Diagram Alir Untuk Membentuk Matriks Frekuensi Baru

Sama seperti dengan deskripsi *library* dan *component,* teks pesan dari pengguna juga dipisahkan setiap kata yang berbeda dan dengan membandingkan dengan daftar kata yang tidak perlu. Setelah itu diberi bobot nilai yang dibandingkan dengan daftar kata. Nilai 1 jika sama dan nilai 0 jika tidak terdapat dalam daftar kata. Daftar nilai *query* tersebut kemudian dibentuk dalam sebuah matriks *query* dan dilakukan *transpose* matriks. Kemudian dihitung matriks vektor *query* dengan melakukan perkalian antara *transpose* matriks *query*, matriks U_k dan *inverse* matriks σ_k. Diagram alir untuk membentuk matriks vektor *query* dapat dilihat seperti pada Gambar 3.26.



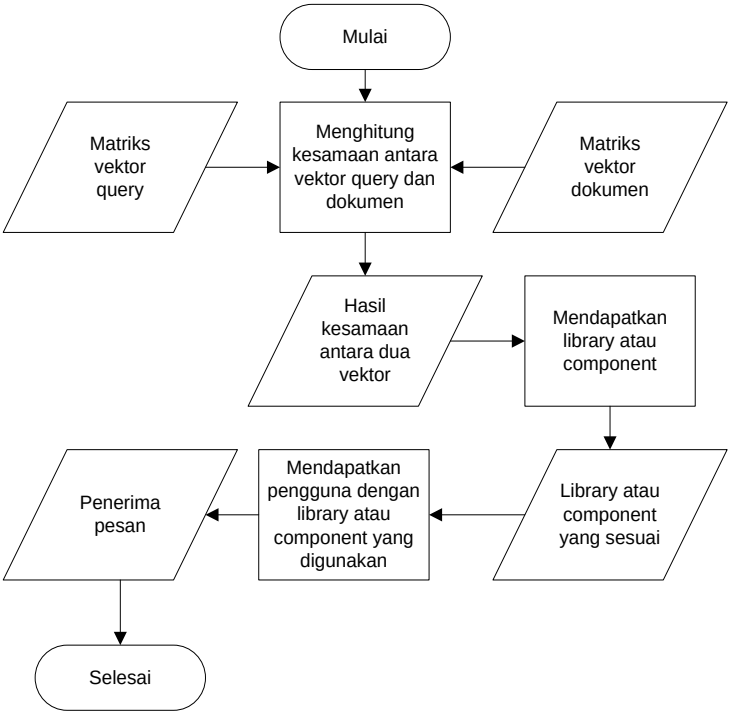
Gambar 3.26 Diagram Alir Membentuk Matriks Vektor *Query*

Deskripsi dari *library* dan *component* yang telah direpresentasikan menjadi dokumen juga dibuat matriks dokumen sesuai dengan jumlah dari dokumen dengan nilai yang sesuai dengan matriks frekuensi baru. Kemudian dilakukan *transpose* matriks dokumen. Setelah itu dihitung matriks vektor dokumen dengan melakukan perkalian antara *transpose* matriks dokumen, matriks U_k dan *inverse* matriks σ_k. Diagram alir untuk membentuk matriks vektor dokumen dapat dilihat pada Gambar 3.27.



Gambar 3.27 Diagram Alir Untuk Membentuk Matriks Vektor Dokumen

Setelah mendapatkan matriks vektor *query* dan dokumen dilakukan perhitungan kesamaan diantara kedua matriks tersebut. Digunakan metode *cosine similarity* untuk menghitung kesamaan diantara dua matriks. Setiap dokumen memiliki nilai kesamaan yang berbeda dengan *query*. Oleh karena itu, diambil nilai kesamaan tertinggi dari semua dokumen tersebut. Kemudian ditentukan pengguna yang menggunakan *library* atau *component* dalam tim proyek yang sesuai dengan dokumen tersebut. Setelah itu *server* mengirimkan pesan dari pengirim kepada penerima yang didapatkan dari hasil klasifikasi. Karena setiap keahlian dapat dimiliki oleh lebih dari satu orang pengguna, maka pesan akan dikirimkan kepada seluruh pengguna dengan keahlian sesuai dengan hasil klasifikasi. Jika tidak didapatkan nilai kesamaan antara *query* dan dokumen maka pesan tidak akan dikirimkan kepada pengembang lain. Diagram alir untuk mendapatkan penerima pesan dapat dilihat seperti pada Gambar 3.28.



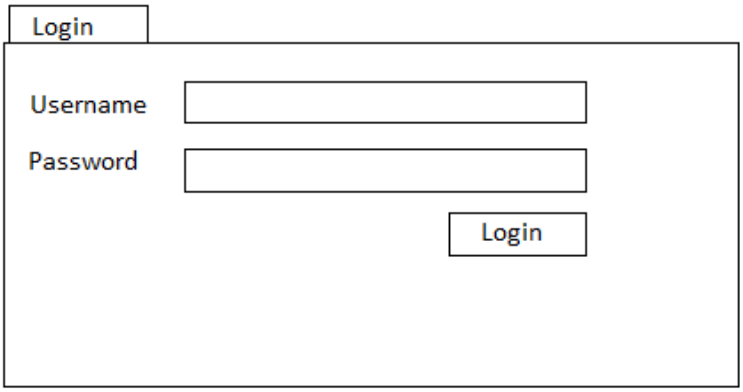
Gambar 3.28 Diagram Alir Untuk Mendapatkan Penerima Pesan

3.2.2. Perancangan Antarmuka

Pada bagian ini akan dijelaskan mengenai perancangan antarmuka dari perangkat lunak yang dibuat. Terdapat satu antarmuka utama yaitu *MessengerView* yang merupakan halaman pada bagian *view* di Eclipse. Antarmuka tersebut dibagi menjadi empat halaman *tab view.* Halaman *tab* tersebut adalah *Login, Register, Team, Message* dan *File.* Selain itu juga terdapat dua jendela pendukung fungsional sistem yaitu jendela *JoinTeam* dan *MessageBox.* Untuk jendela *JoinTeam* juga terdapat dua *panel* tambahan yaitu *panel library* dan *component.*

3.2.2.1. Halaman *Tab Login*

Halaman ini merupakan tampilan bagi pengguna untuk melakukan *login* pada sistem. Pada halaman ini pengguna memasukkan teks *username* dan *password* sesuai dengan *username* dan *password* saat melakukan *register* pada sistem. Jika proses *login* berhasil maka pengguna dapat menggunakan halaman-halaman lain untuk melakukan proses lainnya. Rancangan halaman *login* dapat dilihat pada Gambar 3.29. Penjelasan mengenai atribut-atribut yang terdapat pada halaman ini dapat dilihat pada Tabel 3.11.



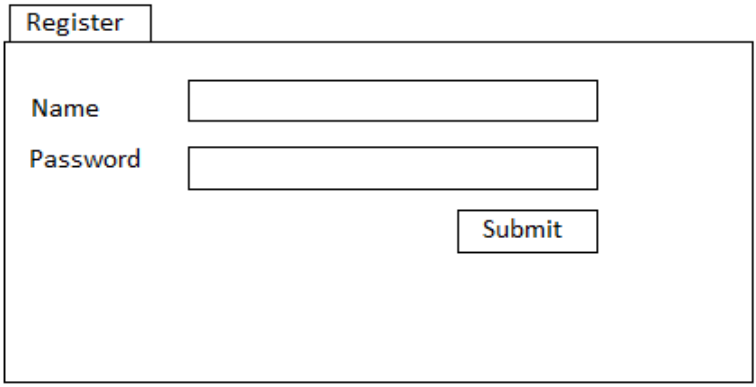
Gambar 3.29 Rancangan Tampilan Halaman *Tab Login*

Tabel 3.11 Spesifikasi Atribut Antarmuka Halaman *Tab Login*

No	Nama Atribut Antarmuka	Jenis Atribut	Kegunaan	Jenis Masukan / Keluaran
1	textUsername	*Text*	Memasukkan teks *username* dari pengguna	*String*
2	textPassword	*Text*	Memasukkan teks *password* dari pengguna	*String*
3	Login	*Button*	Menjalankan perintah untuk melakukan proses *login*	*Selection Event*

3.2.2.2. Halaman *Tab Register*

Halaman ini merupakan tampilan bagi pengguna untuk melakukan *register* pada sistem. Pada halaman ini pengguna memasukkan teks *username* dan *password*. Jika proses *register* berhasil maka pengguna dapat melakukan proses *login* pada halaman *tab login*. Rancangan halaman *register* dapat dilihat pada Gambar 3.30. Penjelasan mengenai atribut-atribut yang terdapat pada halaman ini dapat dilihat pada Tabel 3.12.



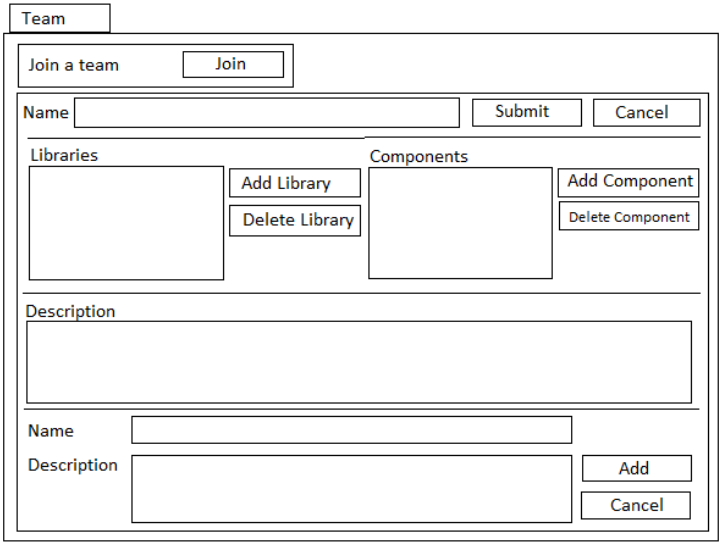
Gambar 3.30 Rancangan Tampilan Halaman *Tab Register*

Tabel 3.12 Spesifikasi Atribut Antarmuka Halaman *Tab Register*

No	Nama Atribut Antarmuka	Jenis Atribut	Kegunaan	Jenis Masukan / Keluaran
1	textName	*Text*	Memasukkan teks *username* dari pengguna	*String*
2	textPass	*Text*	Memasukkan teks *password* dari pengguna	*String*
3	Submit	*Button*	Menjalankan perintah untuk melakukan proses *register*	*Selection Event*

3.2.2.3. Halaman *Tab Team*

Halaman ini merupakan tampilan bagi pengguna untuk membuat tim proyek perangkat lunak. Halaman ini menampilkan *text* untuk memasukkan nama tim, *list* untuk menampilkan daftar *library* dan *component* yang telah dimasukkan, *text* untuk menampilkan deskripsi *library* atau *component* yang dipilih, dan *text* untuk memasukkan nama *library* atau *component* dan deskripsinya. Rancangan halaman *team* dapat dilihat pada Gambar 3.31. Penjelasan mengenai atribut-atribut yang terdapat pada halaman ini dapat dilihat pada Tabel 3.13.



Gambar 3.31 Rancangan Tampilan Halaman *Tab Team*

Tabel 3.13 Spesifikasi Atribut Antarmuka Halaman *Tab Team*

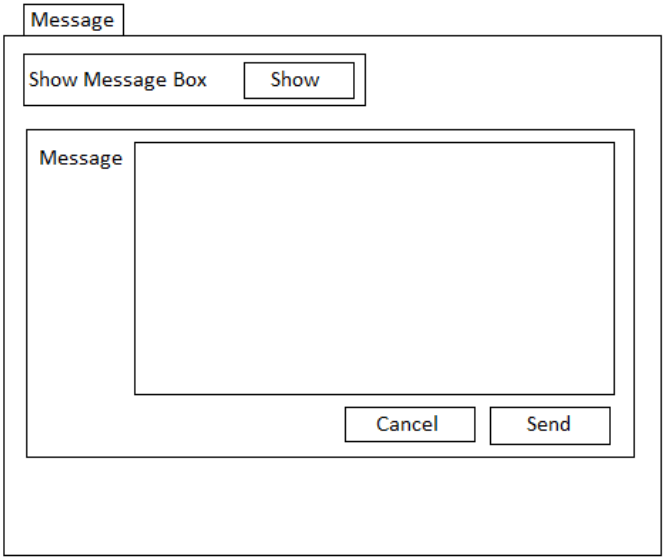
No	Nama Atribut Antarmuka	Jenis Atribut	Kegunaan	Jenis Masukan / Keluaran
1	textTeam Name	*Text*	Memasukkan teks nama tim yang akan dibuat	*String*

No	Nama Atribut Antarmuka	Jenis Atribut	Kegunaan	Jenis Masukan / Keluaran
2	btnSubmit Team	*Button*	Menjalankan perintah untuk membuat tim	*Selection Event*
3	btnCancel CreateTeam	*Button*	Menjalankan perintah untuk tidak membuat tim dan menghapus detil tim yang telah dibuat	*Selection Event*
4	list Library	*List*	Menampilkan daftar *library* yang telah dimasukkan	*String*
5	list Component	*List*	Menampilkan daftar *component* yang telah dimasukkan	*String*
6	btnAdd Library	*Button*	Menjalankan perintah untuk menampilkan *textfield* masukan *library* dan deskripsi	*Selection Event*
7	btnDelete Library	*Button*	Menjalankan perintah untuk menghapus *library* yang dipilih pada *list library*	*Selection Event*
8	btnAdd Component	*Button*	Menjalankan perintah untuk menampilkan *textfield* masukan *component* dan deskripsi	*Selection Event*
9	btnDelete Component	*Button*	Menjalankan perintah untuk menghapus *component* yang dipilih pada *list component*	*Selection Event*
10	text Description	*Text*	Menampilkan deskripsi *library* atau *component* yang dipilih dari *list*	*String*
11	textName	*Text*	Memasukkan teks nama *library* atau *component*	*String*

No	Nama Atribut Antarmuka	Jenis Atribut	Kegunaan	Jenis Masukan / Keluaran
12	textDescript	*Text*	Memasukkan teks deskripsi dari *library* atau *component*	*String*
13	btnAdd	*Button*	Menjalankan perintah untuk melakukan proses penambahan *library* atau *component* pada *list*	*Selection Event*
14	btnCancel Add	*Button*	Menjalankan perintah untuk tidak menambahkan *library* atau *component* pada *list*	*Selection Event*
15	btnJoin	*Buton*	Menjalankan perintah untuk menampilkan antarmuka jendela *JoinTeam*	*Selection Event*

3.2.2.4. Halaman *Tab Message*

Halaman ini merupakan tampilan bagi pengguna untuk mengirimkan teks pesan, dan untuk menampilkan jendela *MessageBox*. Pada halaman ini pengguna dapat memasukkan teks pesan pada *group NewMessage*, dan menekan tombol *show* untuk menampilkan jendela *MessageBox.* Rancangan halaman *message* dapat dilihat pada Gambar 3.32. Penjelasan mengenai atribut-atribut yang terdapat pada halaman ini dapat dilihat pada Tabel 3.14.



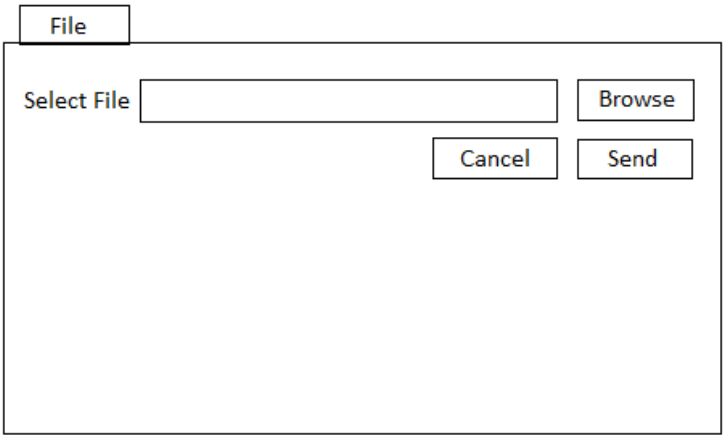
Gambar 3.32 Rancangan Tampilan Halaman *Tab Message*

Tabel 3.14 Spesifikasi Atribut Antarmuka Halaman *Tab Message*

No	Nama Atribut Antarmuka	Jenis Atribut	Kegunaan	Jenis Masukan / Keluaran
1	textMessage	*Text*	Memasukkan teks pesan dari pengguna	*String*
2	btnSend Message	*Button*	Menjalankan perintah untuk mengirimkan pesan	*Selection Event*
3	btnCancel Message	*Button*	Menjalankan perintah untuk tidak mengirimkan pesan	*Selection Event*
4	btnShow	*Button*	Menjalankan perintah untuk menampilkan antarmuka jendela *MessageBox*	*Selection Event*

3.2.2.5. Halaman *Tab File*

Halaman ini merupakan tampilan bagi pengguna untuk mengirimkan *file* kepada pengguna lain. Rancangan halaman *File* dapat dilihat pada Gambar 3.33. Penjelasan mengenai atribut-atribut yang terdapat pada halaman ini dapat dilihat pada Tabel 3.15.



Gambar 3.33 Rancangan Tampilan Halaman *Tab File*

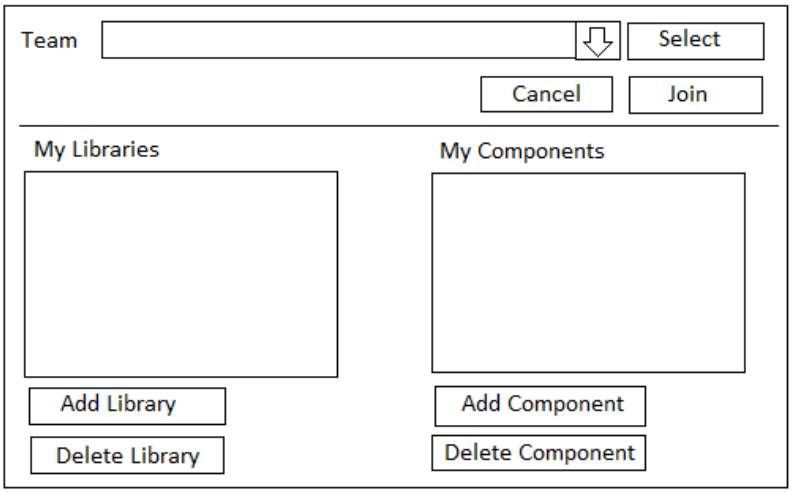
Tabel 3.15 Spesifikasi Atribut Antarmuka Halaman *Tab File*

No	Nama Atribut Antarmuka	Jenis Atribut	Kegunaan	Jenis Masukan / Keluaran
1	textFile	*Text*	Menampilkan alamat *file* yang dipilih pengguna	*String*
2	btnBrowse	*Button*	Menjalankan perintah untuk menampilkan *file dialog*	*Selection Event*
3	btnSendFile	*Button*	Menjalankan perintah untuk melakukan proses pengiriman *file*	*Selection Event*

No	Nama Atribut Antarmuka	Jenis Atribut	Kegunaan	Jenis Masukan / Keluaran
4	btnCancel File	*Button*	Menjalankan perintah untuk tidak melakukan proses pengiriman *file*	*Selection Event*

3.2.2.6. Halaman Jendela *JoinTeam*

Halaman ini merupakan tampilan bagi pengguna untuk mengikuti tim proyek perangkat lunak. Pada halaman ini pengguna dapat memilih tim proyek yang belum diikuti. Setelah itu sistem akan menampilkan daftar *library* dan *component* dari tim tersebut pada *panel library* dan *component.* Pengguna juga dapat menghapus *library* atau *component* yang telah dipilih sebelumnya dari daftar. Rancangan jendela *JoinTeam* dapat dilihat pada Gambar 3.34. Penjelasan mengenai atribut-atribut yang terdapat pada halaman ini dapat dilihat pada Tabel 3.16.



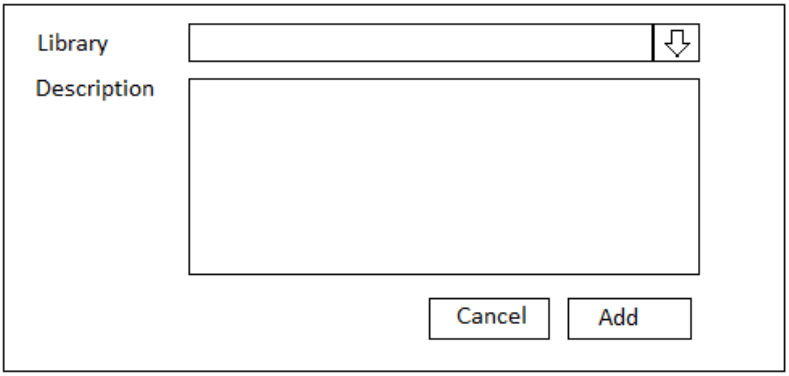
Gambar 3.34 Rancangan Tampilan Halaman Jendela *JoinTeam*

Tabel 3.16 Spesifikasi Atribut Antarmuka Halaman Jendela *JoinTeam*

No	Nama Atribut Antarmuka	Jenis Atribut	Kegunaan	Jenis Masukan / Keluaran
1	comboTeam	*JCombo Box*	Menampilkan daftar tim proyek perangkat lunak yang belum diikuti	*String*
2	btnSelect	*Button*	Menjalankan perintah untuk melakukan proses pencarian *library* dan *component* dari tim yang dipilih	*Selection Event*
3	btnJoin	*Button*	Menjalankan perintah untuk melakukan proses pengiriman data ke *server*	*Selection Event*
4	btnCancel Join	*Button*	Menjalankan perintah untuk tidak melakukan proses pengiriman data ke *server*	*Selection Event*
5	listLibrary	*JList*	Menampilkan daftar *library* yang dipilih	*String*
6	list Component	*JList*	Menampilkan daftar *component* yang dipilih	*String*
7	btnAdd Library	*Button*	Menjalankan perintah untuk menampilkan *panel library*	*Selection Event*
8	btnDelete Library	*Button*	Menjalankan perintah untuk menghapus *library* yang dipilih dari *list library*	*Selection Event*
9	btnAdd Component	*Button*	Menjalankan perintah untuk menampilkan *panel component*	*Selection Event*
10	btnDelete Component	*Button*	Menjalankan perintah untuk menghapus *component* yang dipilih dari *list component*	*Selection Event*

3.2.2.6.1. Halaman *Panel Library*

Halaman ini merupakan tampilan bagi pengguna untuk memilih *library* dari tim proyek perangkat lunak yang dipilih pada jendela *JoinTeam*. Pada halaman ini pengguna dapat memilih *library* yang belum dipilih sebelumnya dan pengguna dapat melihat deskripsi dari *library* tersebut pada *text* deskripsi. Rancangan *panel Library* dapat dilihat pada Gambar 3.35. Penjelasan mengenai atribut-atribut yang terdapat pada halaman ini dapat dilihat pada Tabel 3.17.



Gambar 3.35 Rancangan Tampilan Halaman *Panel Library*

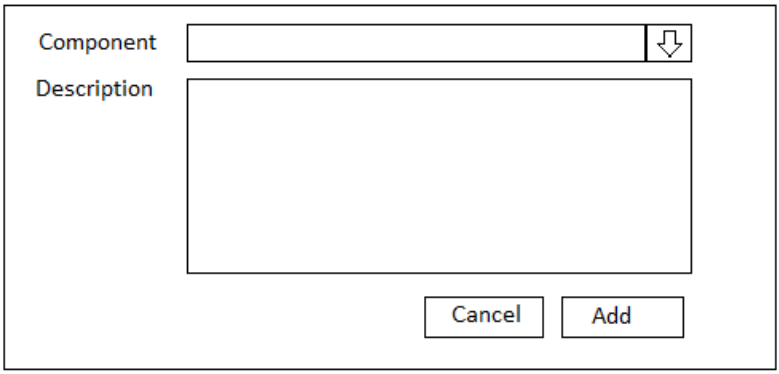
Tabel 3.17 Spesifikasi Atribut Antarmuka Halaman *Panel Library*

No	Nama Atribut Antarmuka	Jenis Atribut	Kegunaan	Jenis Masukan / Keluaran
1	combo Library	*JCombo Box*	Menampilkan daftar *library* yang belum dipilih oleh pengguna	*String*
2	textLibDes	*Text*	Menampilkan deskripsi dari *library* yang dipilih	*String*
3	btnAdd	*Button*	Menjalankan perintah untuk melakukan proses penambahan *library* pada *list library*	*Selection Event*

No	Nama Atribut Antarmuka	Jenis Atribut	Kegunaan	Jenis Masukan / Keluaran
4	btnCancel	*Button*	Menjalankan perintah untuk tidak melakukan proses penambahan *library* pada *list library*	*Selection Event*

3.2.2.6.2. Halaman *Panel Component*

Halaman ini merupakan tampilan bagi pengguna untuk memilih *component* dari tim proyek perangkat lunak yang dipilih pada jendela *JoinTeam*. Pada halaman ini pengguna dapat memilih *component* yang belum dipilih sebelumnya dan pengguna dapat melihat deskripsi dari *component* tersebut pada *text* deskripsi. Rancangan *panel component* dapat dilihat pada Gambar 3.36. Penjelasan mengenai atribut-atribut yang terdapat pada halaman ini dapat dilihat pada Tabel 3.18.



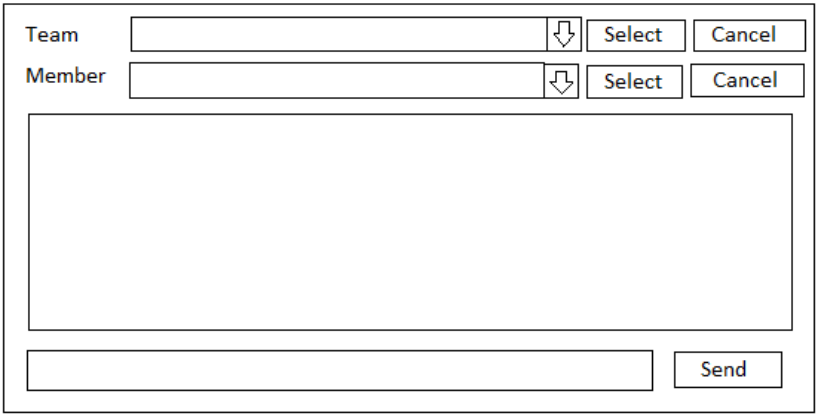
Gambar 3.36 Rancangan Tampilan Halaman *Panel Component*

Tabel 3.18 Spesifikasi Atribut Antarmuka Halaman *Panel Component*

No	Nama Atribut Antarmuka	Jenis Atribut	Kegunaan	Jenis Masukan / Keluaran
1	combo Component	*JCombo Box*	Menampilkan daftar *component* yang belum dipilih oleh pengguna	*String*
2	textCompDes	*Text*	Menampilkan deskripsi dari *component* yang dipilih	*String*
3	btnAdd2	*Button*	Menjalankan perintah untuk melakukan proses penambahan *component* pada *list component*	*Selection Event*
4	btnCancel2	*Button*	Menjalankan perintah untuk tidak melakukan proses penambahan *component* pada *list component*	*Selection Event*

3.2.2.7. Halaman Jendela *MessageBox*

Halaman ini merupakan tampilan bagi pengguna untuk melihat riwayat pesan dari pengguna lain dan untuk membalas pesan dari pengguna tersebut. Pada halaman ini pengguna dapat memilih tim proyek dan sistem akan menampilkan daftar anggota dari tim tersebut. Kemudian pengguna dapat memilih pengguna dari daftar anggota tersebut dan sistem akan menampilkan riwayat pesan dari pengguna yang dipilih. Setelah itu pengguna dapat membalas pesan dari pengguna tersebut dengan memasukkan teks pesan pada *text.* Rancangan jendela *MessageBox* dapat dilihat pada Gambar 3.37. Penjelasan mengenai atribut-atribut yang terdapat pada halaman ini dapat dilihat pada Tabel 3.19.



Gambar 3.37 Rancangan Tampilan Halaman Jendela *MessageBox*

Tabel 3.19 Spesifikasi Atribut Antarmuka Halaman Jendela *MessageBox*

No	Nama Atribut Antarmuka	Jenis Atribut	Kegunaan	Jenis Masukan / Keluaran
1	comboBox Team	*JCombo Box*	Menampilkan daftar tim yang diikuti pengguna	*String*
2	comboBox Member	*JCombo Box*	Menampilkan daftar anggota tim yang dipilih	*String*
3	btnSelect Team	*Button*	Menjalankan perintah untuk melakukan proses pencarian anggota tim yang dipilih	*Selection Event*
4	btnCancel Team	*Button*	Menjalankan perintah untuk menutup antarmuka jendela *MessageBox*	*Selection Event*
5	btnSelect Member	*Button*	Menjalankan perintah untuk melakukan proses pencarian riwayat pesan	*Selection Event*

No	Nama Atribut Antarmuka	Jenis Atribut	Kegunaan	Jenis Masukan / Keluaran
			dengan anggota yang dipilih	
6	btnCancel Member	*Button*	Menjalankan perintah untuk tidak melakukan proses pencarian riwayat pesan dengan anggota yang dipilih	*Selection Event*
7	textArea Message	*Text*	Menampilkan riwayat pesan pengguna dengan anggota yang dipilih	*String*
8	textInput	*Text*	Memasukkan teks pesan dari pengguna	*String*
9	btnSend	*Button*	Menjalankan perintah untuk melakukan proses pengiriman pesan kepada anggota yang dipilih	*Selection Event*

BAB IV
IMPLEMENTASI

Bab ini membahas tentang implementasi dari perancangan sistem. Bab ini berisi proses implementasi dari setiap kelas pada arsitektur *client* dan *server*. Bahasa pemrograman yang digunakan adalah bahasa pemrograman Java.

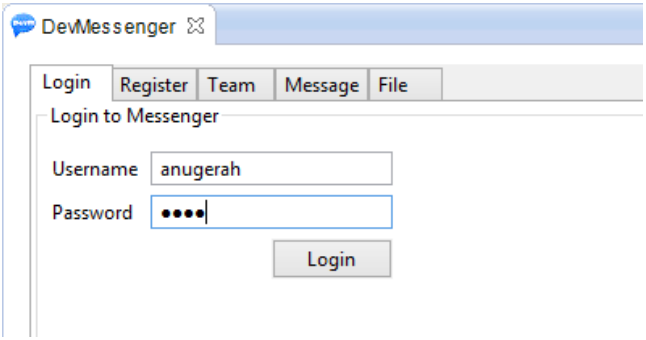
4.1. Implementasi Kelas *Boundary*

Kelas *Boundary* adalah kelas yang berfungsi untuk mengatur antarmuka sistem. Pada bagian ini akan dijelaskan implementasi dari rancangan kelas pada kelas *boundary.* Terdapat tiga kelas *boundary* pada sistem ini, yaitu kelas *MessengerView*, *JoinTeam* dan *MessageBox*.

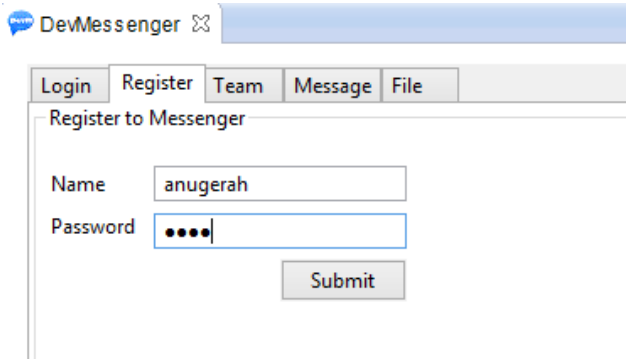
Kelas *MessengerView* merupakan kelas antarmuka yang digunakan untuk membuat *view* pada Eclipse. Pada kelas ini terdapat fungsi yang merupakan turunan dari kelas abstrak *ViewPart*, yaitu fungsi *createPartControl* dan *setFocus*. Fungsi *setFocus* digunakan untuk membuat kontrol tertentu menjadi fokus. Fungsi *createPartControl* digunakan untuk membangun desain pada tampilan *view* dan terdapat elemen-elemen untuk membangun tampilan *view,* yaitu *tab Login, Register, Team, Message* dan *File.* Elemen-elemen tersebut merupakan komponen dari *SWT widget*.

Tab Login adalah halaman untuk melakukan *login* pada sistem. Pada halaman ini pengguna memasukkan teks *username* dan *password* dan harus sesuai dengan *username* dan *password* saat melakukan *register* pada sistem. Tampilan halaman *tab Login* dapat dilihat pada Gambar 4.1.

Tab Register adalah halaman untuk melakukan *register* pada sistem. Pada halaman ini pengguna memasukkan teks *username* dan *password* dan sistem akan menyimpan username dan password yang dimasukkan pada *database* jika tidak ada *username* yang sama. Tampilan halaman *tab Register* dapat dilihat pada Gambar 4.2.



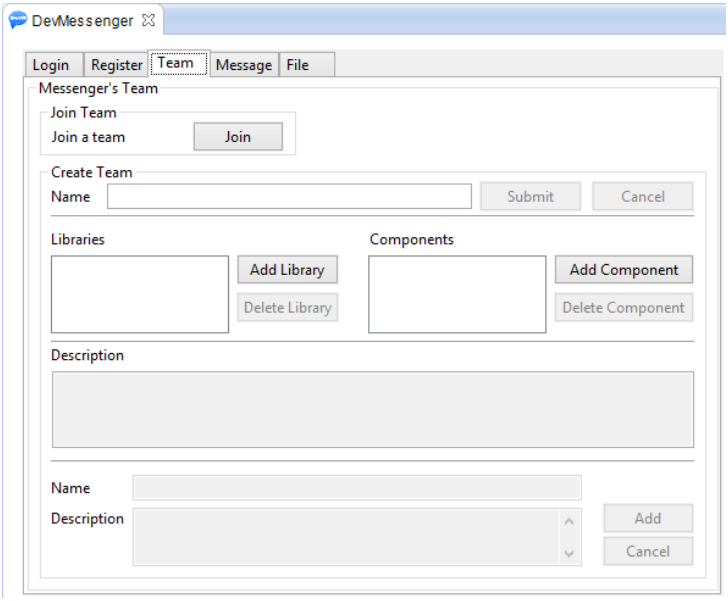
Gambar 4.1 Tampilan Halaman *Tab Login*



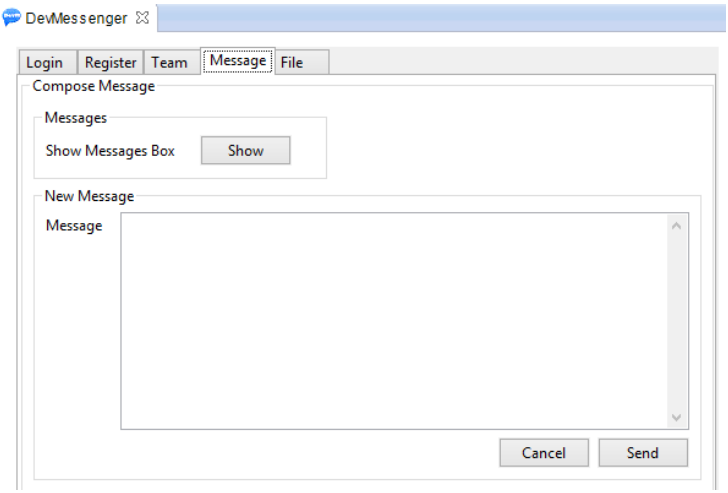
Gambar 4.2 Tampilan Halaman *Tab Register*

Tab Team adalah halaman untuk membuat tim proyek perangkat lunak dan untuk membuka jendela *JoinTeam*. Pada halaman ini pengguna dapat memasukkan nama tim yang belum pernah dibuat atau tidak sama dengan tim di *database* dan memasukkan daftar *library* dan *component* yang berhubungan dengan tim tersebut. Tampilan halaman *tab Team* dapat dilihat pada Gambar 4.3.

Tab Message adalah halaman untuk mengirimkan teks pesan, dan untuk menampilkan jendela *MessageBox*. Tampilan halaman *tab Message* dapat dilihat pada Gambar 4.4.

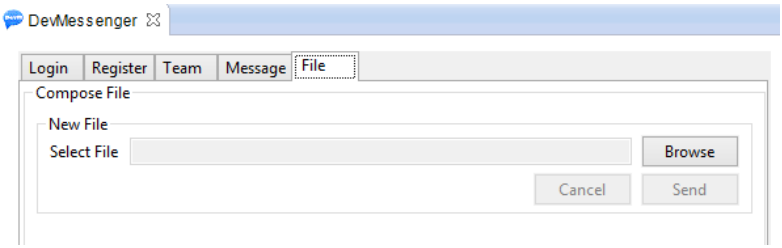


Gambar 4.3 Tampilan Halaman *Tab Team*



Gambar 4.4 Tampilan Halaman *Tab Message*

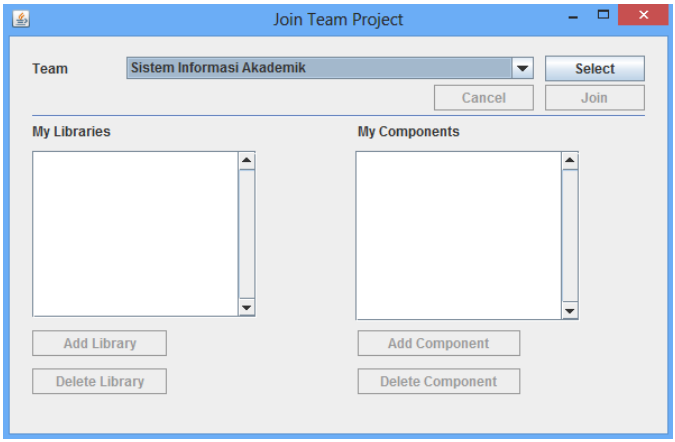
Tab File adalah halaman untuk mengirimkan *file*. Tampilan halaman *tab File* dapat dilihat pada Gambar 4.5.



Gambar 4.5 Tampilan Halaman Tab File

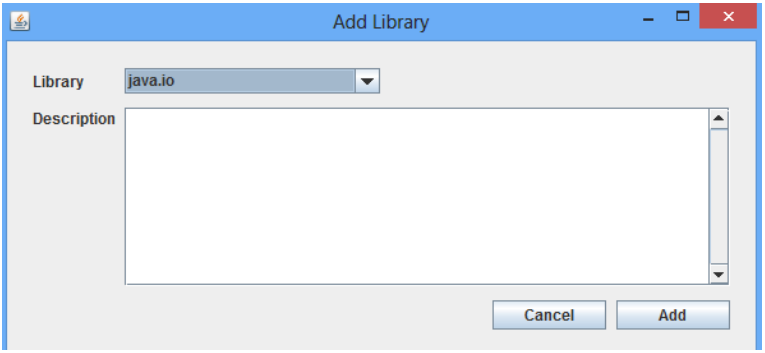
Kelas *JoinTeam* merupakan kelas antarmuka yang digunakan untuk membuat tampilan antarmuka jendela *JoinTeam*. Pada kelas ini terdapat fungsi *initialize* yang berguna untuk membangun desain pada tampilan jendela antarmuka *JoinTeam* dan membangun desain antarmuka tambahan, yaitu *panel library* dan *component*. Di dalam fungsi tersebut terdapat elemen-elemen untuk membangun tampilan yang merupakan komponen dari *SWING*.

Jendela *JoinTeam* merupakan halaman untuk mengikuti tim proyek perangkat lunak. Pada halaman ini pengguna dapat memilih tim proyek yang belum diikuti pada *combobox* tim. Setelah itu sistem akan menampilkan daftar *library* dan *component* dari tim tersebut pada *panel library* dan *component.* Untuk mengikuti tim, pengguna harus memilih *library* dan *component* terlebih dahulu. Tampilan halaman jendela *JoinTeam* dapat dilihat pada Gambar 4.6.



Gambar 4.6 Tampilan Halaman Jendela *JoinTeam*

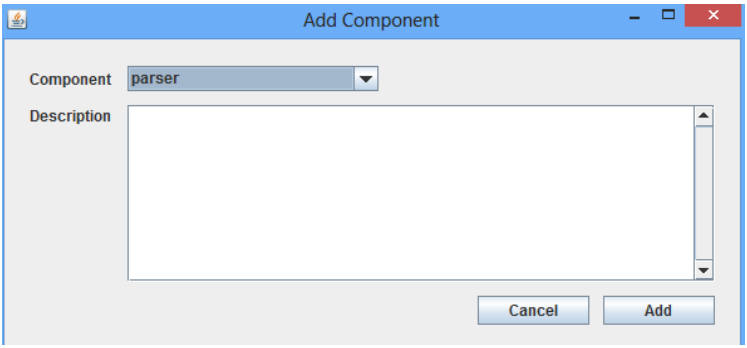
Panel Library adalah halaman untuk memilih *library* dari tim proyek perangkat lunak yang dipilih pada jendela *JoinTeam*. Pada halaman ini pengguna dapat memilih *library* yang belum dipilih sebelumnya dan melihat deskripsi dari *library* tersebut. Tampilan halaman *panel library* dapat dilihat pada Gambar 4.7.



Gambar 4.7 Tampilan Halaman *Panel Library*

Panel component adalah halaman untuk memilih *component* dari tim proyek perangkat lunak yang dipilih pada jendela

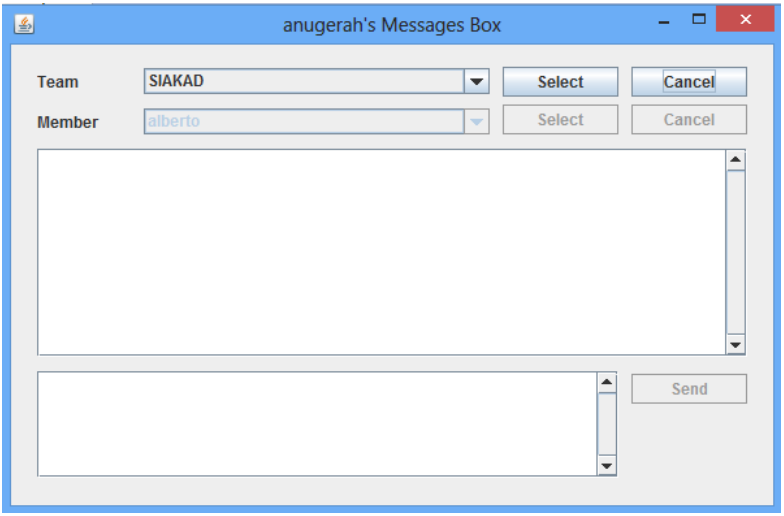
JoinTeam. Pada halaman ini pengguna dapat memilih *component* yang belum dipilih sebelumnya dan melihat deskripsi dari *component* tersebut. Tampilan halaman *panel component* dapat dilihat pada Gambar 4.8.



Gambar 4.8 Tampilan Halaman *Panel Component*

Kelas *MessageBox* merupakan kelas antarmuka yang digunakan untuk membuat tampilan antarmuka jendela *MessageBox*. Pada kelas ini terdapat fungsi *initialize* yang berguna untuk membangun desain pada tampilan jendela antarmuka *MessageBox*. Di dalam fungsi tersebut terdapat elemen-elemen untuk membangun tampilan yang merupakan komponen dari *SWING*.

Jendela *MessageBox* adalah halaman untuk melihat riwayat pesan dari anggota lain dalam sebuah tim dan untuk membalas pesan dari anggota tersebut. Pada halaman ini pengguna dapat memilih tim proyek dan sistem akan menampilkan daftar anggota dari tim tersebut. Kemudian pengguna dapat memilih pengguna dari daftar anggota dan sistem akan menampilkan riwayat pesan dari pengguna yang dipilih. Setelah itu pengguna dapat membalas pesan dari pengguna yang dipilih dengan memasukkan teks pesan. Tampilan halaman jendela *MessageBox* dapat dilihat pada Gambar 4.9.



Gambar 4.9 Tampilan Halaman Jendela *MessageBox*

4.2. Implementasi Kelas *Control*

Pada bagian ini akan dijelaskan implementasi dari rancangan kelas *control*. Kelas ini berhubungan dengan kelas *boundary* dan kelas *entity*. Terdapat enam kelas control, yaitu *SocketClient, Upload, Download, ThreadServer, SocketServer* dan *LatentSemanticIndexing.*

Kelas *SocketClient* merupakan kelas yang memiliki hubungan dengan semua kelas *boundary* dan kelas *entity.* Kelas ini mengimplementasikan antarmuka *Runnable* dari *thread*. Pada kelas ini terdapat fungsi *run* yaitu fungsi yang dijalankan dengan *thread* lain. Untuk menjalankan *thread client*, di kelas *SocketClient* dipanggil konstruktor untuk membuat objek *thread*. Kemudian, dipanggil fungsi *start* untuk melakukan konfigurasi *thread* dan menjalankan fungsi *run* yang berfungsi sebagai penerima *streaming* data dari *server*. Pada fungsi ini, setiap aliran data yang masuk dirubah menjadi objek data. Kemudian dilakukan klasifikasi data sesuai dengan tipe, seperti *login*, *register*, membuat tim, mengikuti tim, mengirimkan pesan dan *file*. Selain itu juga terdapat

fungsi *sendServer* yang berfungsi sebagai pengirim atau *streaming* data ke *server*. Fungsi ini memiliki parameter objek data untuk dikirimkan ke *server*.

Kelas *Upload* digunakan pengguna untuk mengirimkan *file* kepada pengguna lainnya. Pada kelas ini dibuat sebuah *thread* untuk melakukan *upload file*. Kelas ini memiliki parameter yaitu alamat IP penerima, nomor *port* penerima, *file* dan objek *SocketClient*. Terdapat fungsi *run* untuk melakukan *upload file* kepada penerima.

Kelas *Download* digunakan pengguna untuk mengunduh *file* yang dikirimkan oleh pengguna lainnya. Pada kelas ini dibuat sebuah *thread* untuk melakukan *download file*. Kelas ini memiliki parameter yaitu alamat untuk menyimpan *file* dan objek *SocketClient*. Terdapat fungsi *run* untuk melakukan *download file.*

Kelas *ThreadServer* merupakan kelas turunan dari *java.lang.Thread* atau kelas dengan *extend thread*. kelas ini memiliki semua fungsi untuk membuat dan menjalankan *thread.* Fungsi paling penting adalah *run*, yang bisa dibeban-lebihkan untuk melakukan tugas yang dibutuhkan. Fungsi *run* berfungsi sebagai penerima *streaming* data dari *client*. Selain itu juga terdapat fungsi *sendClient* yang berfungsi sebagai pengirim atau *streaming* data ke *client*.

Kelas *SocketServer* merupakan kelas yang mengimplementasikan *Runnable* dari *thread*. Pada kelas ini, *server* menerima *client* yang mengirimkan permintaan koneksi kepada *server. Client* tersebut kemudian dibuatkan *thread* dengan tipe *ThreadServer. Server* menerima permintaan koneksi dari *client* pada fungsi *run*. Kemudian memanggil fungsi *addThreadClient* untuk membuat *thread* pada *client*. Di kelas ini juga terdapat metode *handleClient* yang berfungsi untuk memproses data yang diterima dari *client* sesuai dengan tipe, seperti: *login, register*, membuat tim, mengikuti tim, dan mengirimkan pesan.

Di kelas ini juga berisi metode-metode untuk manipulasi data dari *database* MySQL. Terdapat fungsi *connectDatabase* yang berfungsi untuk melakukan koneksi kepada *database*

MySQL. Terdapat berbagai fungsi yang digunakan untuk manipulasi data pada *database*, seperti fungsi untuk proses login, menambahkan pengguna baru, menambahkan tim, mengikuti tim, mencari anggota tim, mencari detil tim, menyimpan dan mencari riwayat pesan pengguna.

Kelas *LatentSemanticIndexing* merupakan kelas yang berfungsi untuk melakukan klasifikasi pesan yang diterima oleh *server.* Dalam kelas ini pesan yang diterima dari *client* diuraikan kedalam sebuah matriks frekuensi yang berisi bobot dari setiap kata dan dokumen. Dokumen yang digunakan dalam kelas ini adalah deskripsi dari setiap *library* dan *component* sebuah tim proyek dimana pesan tersebut berhubungan dengan tim tersebut. Deskripsi dari tim tersebut dipotong untuk memisahkan setiap kata dan dilakukan proses *stemming* atau mengubah kata menjadi kata dasar. Proses *stemming* menggunakan pustaka dari *apache lucene*. Kemudian setiap kata tersebut dibandingkan dengan daftar kata yang termasuk *stopwords* atau kata yang sering keluar. Setelah itu daftar kata dimasukkan ke dalam *list*. Potongan fungsi ini dapat dilihat pada Kode Sumber 4.1.

```
ArrayList<String> doc = new ArrayList<String>();
String[] stopWords = stopWord();
int queryIndex = 0;

ArrayList<Double> similarityTemp = new
ArrayList<Double>();
ArrayList<Double> similarity = new ArrayList<Double>();

int indexReceiver = 0;
for(int i = 0; i < count; i++)
{
  String[] detail =
stem(detailTeam.get(i)).split("\\s+|,\\s*|\\.\\s*|:\\s*");
  for(int j = 0;j < detail.length;j++)
  {
   if(!doc.contains(detail[j].toLowerCase()))
   {
if(!Arrays.asList(stopWords).contains(detail[j].toLowerCas
e()))
    {
     doc.add(detail[j].toLowerCase());}
   }
```

```
   }
 }
 Collections.sort(doc);
```

Kode Sumber 4.1 Kode Untuk Memisahkan Kata Dalam Dokumen

Setelah setiap kata yang berbeda dimasukkan ke dalam *list*, setiap kata tersebut dibandingkan dengan setiap dokumen atau deskripsi dari tim. Apabila kata yang dibandingkan terdapat pada dokumen, maka diberi bobot nilai satu pada *array matrixSVD*. Potongan fungsi ini dapat dilihat pada Kode Sumber 4.2.

```
double[][] matrixSVD = new double[doc.size()][count];
double[][] query = new double[1][doc.size()];

for(int i = 0; i < doc.size(); i++)
{
 for(int j = 0; j < count; j++)
 {
  String[] detail =
stem(detailTeam.get(j)).split("\\s+|,\\s*|\\.\\s*|:\\s*");
   outerloop:
   for(int k = 0;k < detail.length;k++)
   {
    if(doc.get(i).equals(detail[k].toLowerCase()))
    {
      matrixSVD[i][j] = 1.0;
      break outerloop;
    }
    else
    {
     matrixSVD[i][j] = 0.0;
    }
   }
  }
}
```

Kode Sumber 4.2 Kode Untuk Memberi Bobot Nilai Dokumen dan Kata

Pesan yang dikirimkan oleh pengguna juga dipotong untuk memisahkan setiap kata. Potongan fungsi ini dapat dilihat pada Kode Sumber 4.3. Setiap kata dalam pesan dibandingkan dengan setiap kata pada *list*. Apabila kata yang dibandingkan sama dengan kata pada *list*, maka diberi bobot nilai satu pada *array query*. Potongan fungsi ini dapat dilihat pada Kode Sumber 4.4.

```
String[] message =
stem(_msg).toLowerCase().split("\\s+|,\\s*|\\.\\s*|:\\s*")
;
String[] messages = new String[message.length];
for(int i = 0; i < message.length; i++)
{
 if(!Arrays.asList(messages).contains(message[i]))
 {
  if(!Arrays.asList(stopWords).contains(message[i]))
  {
   messages[queryIndex] = message[i];
   queryIndex++;
  }
 }
}
```

Kode Sumber 4.3 Kode Untuk Memisahkan kata dalam *Query*

```
for(int i = 0;i<doc.size();i++)
{
 outerloop:
 for(int j = 0;j<queryIndex;j++)
 {
  if(doc.get(i).equals(messages[j]))
  {
   query[0][i] = 1.0;
   break outerloop;
  }
  else
  {
   query[0][i] = 0.0;
  }
 }
}
```

Kode Sumber 4.4 Kode Untuk Memberi Bobot Nilai *Query*

Kelas ini menggunakan *library JAMA* yang merupakan *library* untuk operasi aritmatika pada matriks. Dengan kelas ini dapat dibentuk matriks frekuensi dan kemudian didapatkan tiga buah matriks yang membentuk matriks *singular value decomposition* yaitu matriks U, matriks σ dan matriks V. Potongan fungsi ini dapat dilihat pada Kode Sumber 4.5.

```
Matrix matrix = new Matrix(matrixSVD);
SingularValueDecomposition svd = new
SingularValueDecomposition(matrix);

Matrix wordVector = svd.getU();
Matrix sigma = svd.getS();
Matrix documentVector = svd.getV();
```

Kode Sumber 4.5 Kode Untuk Membentuk Matriks SVD

Setelah terbentuk matriks U, σ dan V, dihitung nilai k untuk mengurangi dimensi ketiga matriks tersebut. Setelah menemukan nilai k, maka ketiga matriks tersebut dibangun kembali dengan mengurangi dimensi matriks sebesar k dan membentuk matriks frekuensi yang baru. Potongan fungsi ini dapat dilihat pada Kode Sumber 4.6.

```
int k = (int)
Math.floor(Math.sqrt(matrix.getColumnDimension()));

Matrix reducedWordVector = wordVector.getMatrix(0,
wordVector.getRowDimension() - 1, 0, k - 1);

Matrix reducedSigma = sigma.getMatrix(0, k - 1, 0, k - 1);

Matrix reducedDocumentVector = documentVector.getMatrix(
0, documentVector.getRowDimension() - 1, 0, k - 1);

Matrix weights = reducedWordVector.times(
reducedSigma).times(reducedDocumentVector.transpose());
```

Kode Sumber 4.6 Kode Untuk Membentuk Matriks SVD Dimensi Baru

Digunakan juga metode *cosine similarity* untuk menemukan kesamaan antara pesan dengan *library* atau *component* yang sesuai. Potongan fungsi ini dapat dilihat pada Kode Sumber 4.7.

```
public static double computeSimilarity(Matrix sourceDoc,
Matrix targetDoc)
{
 double dotProduct = 0;
 for(int i = 0;i<sourceDoc.getColumnDimension();i++)
```

```
 dotProduct += (sourceDoc.get(0, i) * targetDoc.get(0,
i));

double eucledianDist = sourceDoc.normF() *
targetDoc.normF();

return dotProduct / eucledianDist;
}
```

Kode Sumber 4.7 Fungsi Untuk Menghitung *Cosine Similarity*

Dari kesamaan tersebut sistem akan menentukan pengguna yang memiliki keahlian atau yang memiliki *library* atau *component* yang dikuasai sesuai dengan hasil klasifikasi. Untuk menggunakan fungsi tersebut dibutuhkan matriks *query* dari pesan pengguna dan matriks dokumen dari detil tim. Setelah membentuk kedua matriks tersebut, maka dilakukan perhitungan *cosine similarity* antara *query* dan dokumen. Potongan fungsi ini dapat dilihat pada Kode Sumber 4.8.

```
Matrix sigmaInverse = reducedSigma.inverse();
Matrix qt = new Matrix(query);
Matrix vq =
(qt.times(reducedWordVector)).times(sigmaInverse);

for (int j = 0; j < weights.getColumnDimension(); j++)
{
Matrix dok = (weights.getMatrix(0,
weights.getRowDimension()-1, j, j));
Matrix doks = dok.transpose();
Matrix vd =
(doks.times(reducedWordVector)).times(sigmaInverse);
similarityTemp.add(computeSimilarity(vq,vd));
similarity.add(computeSimilarity(vq,vd)); }
```

Kode Sumber 4.8 Kode Untuk Menghitung Kesamaan Antara *Query* dan Dokumen

4.3. Implementasi Kelas *Entity*

Pada bagian ini akan dijelaskan implementasi dari rancangan kelas *entity*. Kelas *entity* hanya dapat diakses oleh kelas *control* karena kelas ini hanya berhubungan dengan kelas *control.* Terdapat lima kelas entity, yaitu kelas *Library, Component, Team, Message* dan *Data*.

Kelas *Library* berisi atribut-atribut yang diperlukan untuk merepresentasikan *library* pada saat membuat tim, mengikuti tim, dan mencari detil tim proyek perangkat lunak. Kelas ini berisi atribut nama *library* beserta deskripsinya dan jumlah dari *library* tersebut.

Kelas *Component* berisi atribut-atribut yang diperlukan untuk merepresentasikan *component* pada saat membuat tim, mengikuti tim, dan mencari detil tim proyek perangkat lunak. Kelas ini berisi atribut nama *component* beserta deskripsinya dan jumlah dari *component* tersebut.

Kelas *Team* berisi atribut-atribut yang diperlukan untuk merepresentasikan *team* pada saat membuat tim, mengikuti tim, dan mencari anggota tim proyek perangkat lunak. Kelas ini berisi atribut nama tim proyek perangkat lunak beserta *library* dan *component* seperti pada kelas *Library* dan *Component*.

Kelas *Message* berisi atribut-atribut yang diperlukan untuk merepresentasikan pesan pada saat pengguna mengirimkan pesan baru, membalas pesan dari pengguna lain dan untuk mencari riwayat pesan pengguna dengan anggota tim lain. Kelas ini berisi atribut nama tim, pengirim pesan, penerima pesan, waktu saat pesan dikirimkan, isi pesan, dan jumlah riwayat pesan pengguna.

Kelas *Data* berisi atribut-atribut yang diperlukan untuk merepresentasikan semua kelas *entity* karena kelas *data* merupakan objek yang dikirimkan oleh *client* kepada *server* begitu juga sebaliknya. Kelas ini berisi atribut *team, message*, pengirim, penerima, tipe dan status.

Kelas *entity* juga digunakan untuk menambahkan data atau mengambil data dari *database* MySQL dengan menjadi parameter untuk metode-metode yang ada pada aplikasi. Contoh untuk menambahkan data adalah menambahkan data pengguna, membuat tim baru, menambahkan pengguna dalam tim dan menambahkan riwayat pesan pengguna. Sedangkan untuk mengambil data adalah untuk mengambil data pengguna, tim, detail tim dan riwayat pesan pengguna. Digunakan perintah *query* untuk menambah dan mengambil data dari *database*. Contoh

potongan kode dan fungsi untuk menambahkan data dapat dilihat pada kode sumber 4.9. Contoh potongan kode untuk mengambil data dapat dilihat pada kode sumber 4.10.

```
else if(data.type.equals("replyMessage"))
{
insertHistory(data.sender,data.receiver,data.message.conte
nt,data.message.team);

Message msg = searchHistory(data.sender, data.receiver,
data.message.team);

clients[searchClient(clientID)].sendClient(new
Data(data.type, data.sender, data.receiver, msg,
"success"));

searchThreadClient(data.receiver).sendClient(new
Data(data.type, data.sender, data.receiver, msg,
"success"));
}

public void insertHistory(String _sender, String
_receiver, String _content, String _team) throws
ClassNotFoundException, SQLException
{
  String sender = _sender; String receiver = _receiver;
  String content = _content;String team = _team;
  Connection con = connectDatabase();
  ResultSet rs = null;
  int count = 0;
  Statement stat = con.createStatement();
rs = stat.executeQuery("SELECT id_history FROM history
ORDER BY id_history DESC LIMIT 1 ");
  if (rs.next())
     count = rs.getInt(1);
  count+=1;
  Statement st = con.createStatement();
  st.executeUpdate("INSERT INTO history VALUES('"+ count
+"','"+sender+"','"+receiver+"','"+team+"','"+content+"',N
OW())"); }
```

Kode Sumber 4.9 Kode Untuk Menambahkan Data Pada *Database*

```
Statement stat2 = con.createStatement();
rs2 = stat2.executeQuery("select sender_name,
receiver_name, contents, DATE_FORMAT (send_date,'%W %D %M
%Y %h:%i %p') from history where team = '"+team+"' and
(sender_name = '"+sender+"' and receiver_name =
'"+receiver+"') or (sender_name = '"+receiver+"' and
receiver_name = '"+sender+"')");

while (rs2.next())
{
  senders[i] = rs2.getString(1);
  receivers[i] = rs2.getString(2);
  contents[i] = rs2.getString(3);
  datetimes[i] = rs2.getString(4);
  i++;
}
return new Message(count, team, senders, receivers,
contents, datetimes);
```

Kode Sumber 4.10 Kode Untuk Mengambil Data Dari *Database*

BAB V
PENGUJIAN DAN EVALUASI

Bab ini menjelaskan tentang pengujian dan evaluasi pada aplikasi yang dikembangkan. Pengujian yang dilakukan adalah pengujian terhadap kebutuhan fungsionalitas sistem dan pengujian klasifikasi pesan. Pengujian fungsionalitas mengacu pada kasus penggunaan pada bab tiga. Pada bagian akhir bab ini akan dijelaskan tentang rangkuman hasil pengujian dan hasil evaluasi.

5.1. Lingkungan Pengujian

Lingkungan pengujian sistem pada pengerjaan Tugas Akhir ini dilakukan pada lingkungan dan alat kakas sebagai berikut:

Prosesor	: AMD A8-4500M APU Radeon HD *Graphics* @ 1.90GHz
Memori	: 4.00 GB
Jenis *Device*	: Laptop
Sistem Operasi	: Microsoft Windows 8 Pro 64 bit
IDE Eclipse	: Eclipse Kepler 4.3
JDK	: JDK 7

5.2. Skenario Pengujian

Bagian ini menjelaskan tentang skenario pengujian yang dilakukan. Pengujian yang dilakukan adalah pengujian kebutuhan fungsionalitas dan pengujian klasifikasi pesan. Pengujian fungsionalitass menggunakan metode *black box*. Metode ini menekankan pada kesesuaian hasil keluaran dari sistem. Pengujian terhadap hasil klasifikasi pesan dilakukan dengan membandingkan hasil deteksi sistem dengan hasil analisa yang dilakukan oleh pengembang perangkat lunak berdasarkan sumber data uji. Dengan membandingkan hasil klasifikasi pesan oleh sistem dan pengembang akan didapatkan kesimpulan terhadap hasil klasifikasi sistem.

Untuk pengujian ini digunakan studi kasus aplikasi dari *sourceforge*. Diambil tiga buah aplikasi sebagai bahan pengujian,

yaitu openCSV, aTunes, dan tuxGuitar. Dari ketiga aplikasi tersebut dilakukan pengolahan informasi dan menerjemahkan dari Bahasa Inggris ke Bahasa Indonesia. Daftar aplikasi beserta informasi yang dibutuhkan dapat dilihat pada Tabel 5.1

Tabel 5.1 Daftar Aplikasi untuk Data Uji

No	Nama Aplikasi	Jumlah Anggota	Jumlah *Library*	Jumlah *Component*	Jumlah Pesan
1	openCSV	6	4	3	10
2	aTunes	10	4	5	10
3	tuxGuitar	7	3	4	10

5.2.1. Pengujian Fungsionalitas

Pengujian fungsionalitas sistem dilakukan dengan menyiapkan sejumlah skenario sebagai pengukur keberhasilan dari pengujian. Pengujian fungsionalitas dilakukan dengan mengacu pada kasus penggunaan yang telah dijelaskan pada subbab 3.1.3.2. Pengujian kebutuhan fungsionalitas dapat dijabarkan pada subbab berikut.

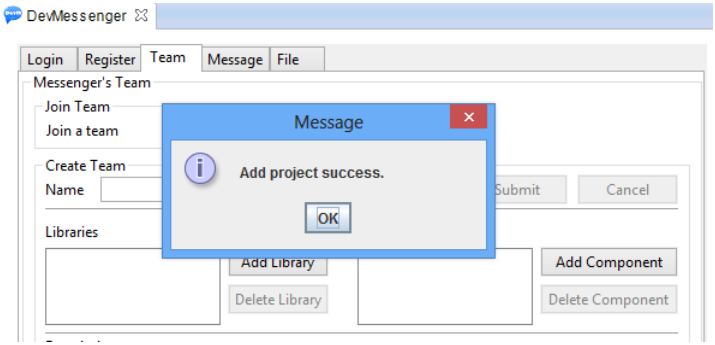
5.2.1.1. Pengujian Fitur Membuat Tim Proyek Perangkat Lunak

Pengujian fitur membuat tim proyek perangkat lunak dilakukan dengan menggunakan data uji aplikasi pertama. Pada pengujian ini data uji yang diperlukan adalah nama tim proyek, daftar *library* dan *component* yang digunakan. Skenario pengujian fungsionalitas sistem dapat dilihat pada Tabel 5.2. Terdapat dua skenario pengujian untuk program saat membuat tim. Hasil tampilan pada program untuk pengujian pertama dapat dilihat pada Gambar 5.1 dan untuk pengujian kedua dapat dilihat pada Gambar 5.2.

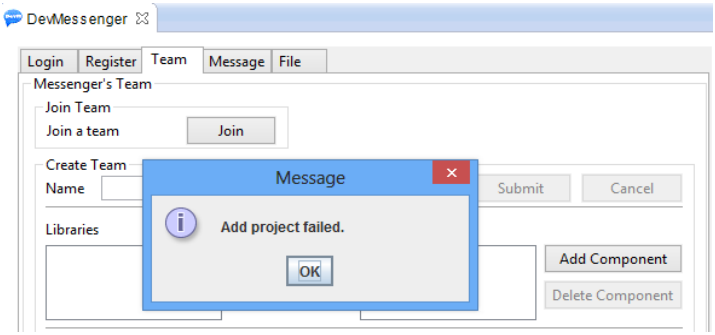
Tabel 5.2 Pengujian Fitur Membuat Tim Proyek Perangkat Lunak

ID	UJ.UC-0001
Referensi Kasus Penggunaan	UC-0001
Nama	Pengujian fitur membuat tim proyek perangkat lunak
Tujuan Pengujian	Menguji fitur untuk membuat sebuah tim proyek perangkat lunak dengan *library* dan *component* yang digunakan.
Skenario 1	**Membuat tim proyek dengan memasukkan nama tim proyek yang belum pernah dibuat sebelumnya.**
Kondisi Awal	*Plugin* dalam keadaan aktif. Belum terdapat nama tim yang akan dimasukkan pada *database*.
Data Uji	**Data uji diperoleh dari aplikasi pertama.**
Langkah Pengujian	1. Pengguna menekan *tab Team.* 2. Pengguna memasukkan nama tim pada *text name*. 3. Pengguna menekan tombol *Add Library.* 4. Pengguna memasukkan nama *library*. 5. Pengguna memasukkan deskripsi *library*. 6. Pengguna menekan tombol *Add*. 7. Pengguna menekan tombol *Submit*.
Hasil Yang Diharapkan	Muncul *message dialog* yang menampilkan pesan *error* gagal membuat tim baru.
Hasil Yang Didapat	Muncul *message dialog* yang menampilkan pesan berhasil membuat tim baru seperti yang terlihat pada Gambar 5.1.
Hasil Pengujian	Berhasil
Kondisi Akhir	*Message dialog* yang menampilkan pesan berhasil membuat tim baru seperti pada Gambar 5.1
Skenario 2	**Membuat tim proyek dengan memasukkan nama tim proyek sama dengan skenario 1.**
Kondisi Awal	*Plugin* dalam keadaan aktif. Sudah terdapat nama tim yang akan dimasukkan pada *database*.
Data Uji	**Data uji diperoleh dari aplikasi pertama.**

Langkah Pengujian	1. Pengguna menekan *tab Team.* 2. Pengguna memasukkan nama tim pada *text name*. 3. Pengguna menekan tombol *Add Library.* 4. Pengguna memasukkan nama *library*. 5. Pengguna memasukkan deskripsi *library*. 6. Pengguna menekan tombol *Add*. 7. Pengguna menekan tombol *Submit*.
Hasil Yang Diharapkan	Muncul *message dialog* yang menampilkan pesan *error* gagal membuat tim baru.
Hasil Yang Didapat	Muncul *message dialog* yang menampilkan pesan *error* gagal membuat tim baru seperti yang terlihat pada Gambar 5.2.
Hasil Pengujian	Berhasil
Kondisi Akhir	*Message dialog* yang menampilkan pesan *error* dapat dilihat pada Gambar 5.2.



Gambar 5.1 Hasil Pengujian Fitur Membuat Tim Untuk Skenario Pertama



Gambar 5.2 Hasil Pengujian Fitur Membuat Tim Untuk Skenario Kedua

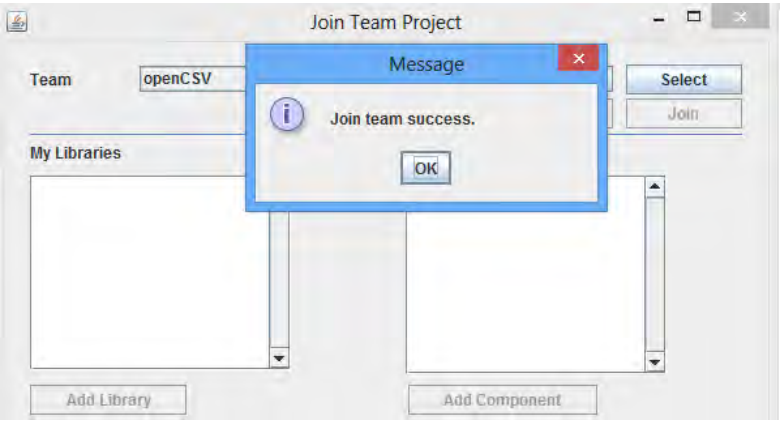
5.2.1.2. Pengujian Fitur Mengikuti Tim Proyek Perangkat Lunak

Pengujian fitur mengikuti tim proyek perangkat lunak dilakukan dengan menggunakan data uji aplikasi pertama. Pada pengujian ini data uji yang diperlukan adalah nama tim proyek, daftar *library* dan *component* yang digunakan. Skenario pengujian fungsionalitas sistem dapat dilihat pada Tabel 5.3. Hasil tampilan pada program saat pengujian dapat dilihat pada Gambar 5.3.

Tabel 5.3 Pengujian Fitur Mengikuti Tim Proyek Perangkat Lunak

ID	UJ.UC-0002
Referensi Kasus Penggunaan	UC-0002
Nama	Pengujian fitur mengikuti tim proyek perangkat lunak
Tujuan Pengujian	Menguji fitur untuk mengikuti sebuah tim proyek perangkat lunak dengan daftar *library* dan *component* yang digunakan oleh pengguna.
Skenario 1	**Mengikuti tim proyek dengan memilih nama tim proyek dan *library* serta *component*.**
Kondisi Awal	*Plugin* dalam keadaan aktif. Terdapat tim yang belum pernah diikuti sebelumnya.
Data Uji	**Data uji diperoleh dari aplikasi pertama.**

Langkah Pengujian	1. Pengguna menekan *tab Team.* 2. Pengguna menekan tombol *Join*. 3. Pengguna memilih tim pada *combobox* tim. 4. Pengguna menekan tombol *Select*. 5. Pengguna menekan tombol *Add Library.* 6. Pengguna memilih *library* pada *combobox library*. 7. Pengguna menekan tombol *Add*. 8. Pengguna menekan tombol *Join*.
Hasil Yang Diharapkan	Muncul *message dialog* yang menampilkan pesan berhasil mengikuti tim.
Hasil Yang Didapat	Muncul *message dialog* yang menampilkan pesan berhasil mengikuti tim seperti yang terlihat pada Gambar 5.3.
Hasil Pengujian	Berhasil
Kondisi Akhir	*Message dialog* yang menampilkan pesan berhasil dapat dilihat pada Gambar 5.3.



Gambar 5.3 Hasil Pengujian Fitur Mengikuti Tim

5.2.1.3. Pengujian Fitur Mengirimkan Pesan Baru

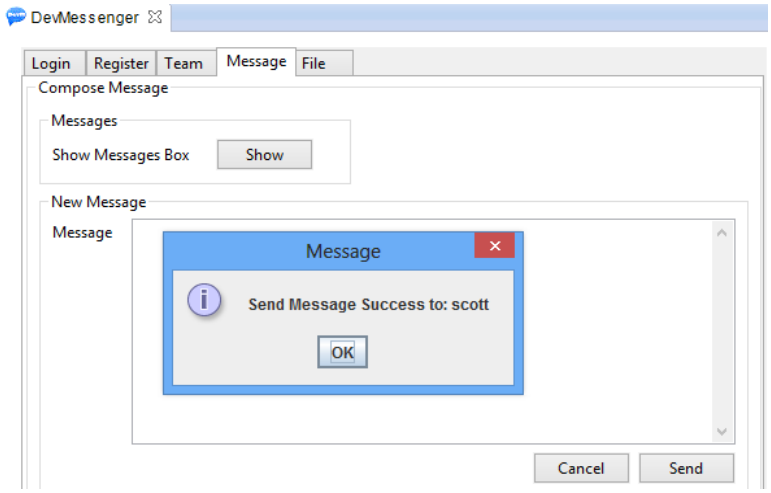
Pengujian fitur mengirimkan pesan baru dilakukan dengan menggunakan data uji aplikasi pertama. Pada pengujian ini data uji yang diperlukan adalah nama tim proyek dan teks pesan

yang berhubungan dengan tim tersebut. Skenario pengujian fungsionalitas sistem dapat dilihat pada Tabel 5.4. Hasil tampilan program saat pengujian dapat dilihat pada Gambar 5.4.

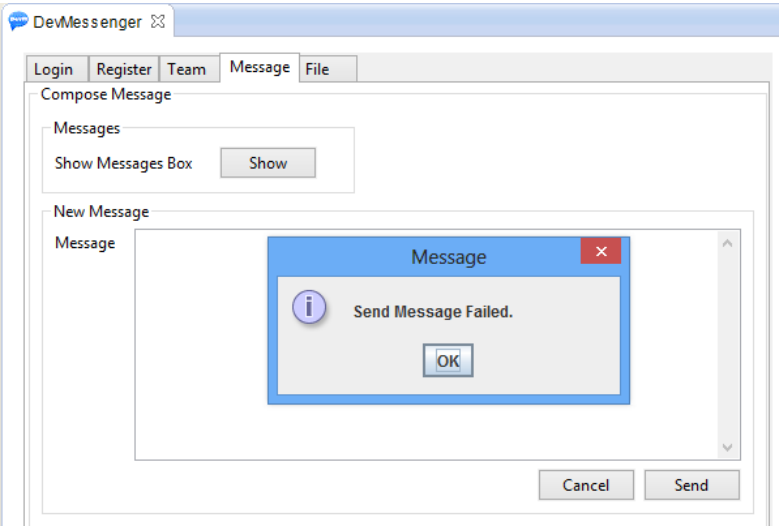
Tabel 5.4 Pengujian Fitur Mengirimkan Pesan Baru

ID	UJ.UC-0003
Referensi Kasus Penggunaan	UC-0003
Nama	Pengujian fitur mengirimkan pesan baru.
Tujuan Pengujian	Menguji fitur untuk mengirimkan pesan baru kepada pengguna lain.
Skenario 1	**Mengirimkan pesan baru dengan memilih nama tim proyek dan memasukkan teks pesan yang berhubungan dengan tim tersebut.**
Kondisi Awal	*Plugin* dalam keadaan aktif. Terdapat pengguna lain yang aktif menggunakan *plugin*.
Data Uji	**Data uji diperoleh dari aplikasi pertama.**
Langkah Pengujian	1. Pengguna menekan *tab Message.* 2. Pengguna memasukkan teks pesan pada *text message*. 3. Pengguna menekan tombol *Send*. 4. Pengguna memilih *tim*. 5. Pengguna menekan tombol *Ok*.
Hasil Yang Diharapkan	Muncul *message dialog* yang menampilkan pesan berhasil dikirim.
Hasil Yang Didapat	Muncul *message dialog* yang menampilkan pesan berhasil dikirim seperti yang terlihat pada Gambar 5.4.
Hasil Pengujian	Berhasil
Kondisi Akhir	*Message dialog* yang menampilkan pesan berhasil dikirim dapat dilihat pada Gambar 5.4.
Skenario 2	**Mengirimkan pesan baru dengan memilih nama tim proyek dan memasukkan teks pesan yang tidak berhubungan dengan tim tersebut.**
Kondisi Awal	*Plugin* dalam keadaan aktif. Terdapat pengguna lain yang aktif menggunakan *plugin*.

Data Uji	**Data uji diperoleh dari aplikasi pertama.**
Langkah Pengujian	1. Pengguna menekan *tab Message.* 2. Pengguna memasukkan teks pesan pada *text message*. 3. Pengguna menekan tombol *Send*. 4. Pengguna memilih *tim*. 5. Pengguna menekan tombol *Ok*.
Hasil Yang Diharapkan	Muncul *message dialog* yang menampilkan pesan tidak berhasil dikirim.
Hasil Yang Didapat	Muncul *message dialog* yang menampilkan pesan tidak berhasil dikirim seperti yang terlihat pada Gambar 5.5.
Hasil Pengujian	Berhasil
Kondisi Akhir	*Message dialog* yang menampilkan pesan tidak berhasil dikirim dapat dilihat pada Gambar 5.5.



Gambar 5.4 Hasil Pengujian Fitur Mengirimkan Pesan Baru Untuk Skenario Pertama



Gambar 5.5 Hasil Pengujian Fitur Mengirimkan Pesan Baru Untuk Skenario Kedua

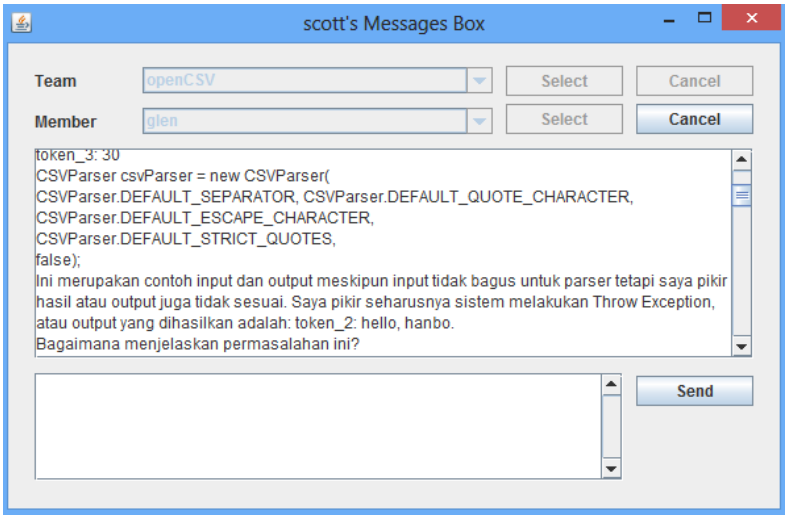
5.2.1.4. Pengujian Fitur Melihat Riwayat Pesan

Pengujian fitur melihat riwayat pesan dilakukan dengan menggunakan data uji aplikasi pertama. Pada pengujian ini data uji yang diperlukan adalah nama tim proyek dan nama anggota. Skenario pengujian fungsionalitas sistem dapat dilihat pada Tabel 5.5. Hasil tampilan program saat pengujian dapat dilihat pada Gambar 5.6.

Tabel 5.5 Pengujian Fitur Melihat Riwayat Pesan

ID	UJ.UC-0004
Referensi Kasus Penggunaan	UC-0004
Nama	Pengujian fitur melihat riwayat pesan.
Tujuan Pengujian	Menguji fitur untuk melihat riwayat pesan antara pengguna dengan pengguna lain.
Skenario 1	**Melihat riwayat pesan pengguna dengan memilih nama tim proyek dan anggota tim.**

Kondisi Awal	*Plugin* dalam keadaan aktif. Sudah terdapat riwayat pesan antara pengguna dengan anggota tim yang dipilih.
Data Uji	**Data uji diperoleh dari aplikasi pertama.**
Langkah Pengujian	1. Pengguna menekan *tab Message.* 2. Pengguna memilih tim pada *combobox* tim. 3. Pengguna menekan tombol *Select*. 4. Pengguna memilih anggota pada *combobox* anggota. 5. Pengguna menekan tombol *Select*.
Hasil Yang Diharapkan	Muncul riwayat pesan antara pengguna dengan anggota tim yang dipilih pada daftar riwayat pesan.
Hasil Yang Didapat	Muncul daftar pesan yang telah dikirimkan pada daftar riwayat pesan pengirim seperti yang terlihat pada Gambar 5.6.
Hasil Pengujian	Berhasil
Kondisi Akhir	Muncul daftar pesan yang telah dikirimkan pada daftar riwayat pesan pengirim yang dapat dilihat pada Gambar 5.6.



Gambar 5.6 Hasil Pengujian Melihat Riwayat Pesan

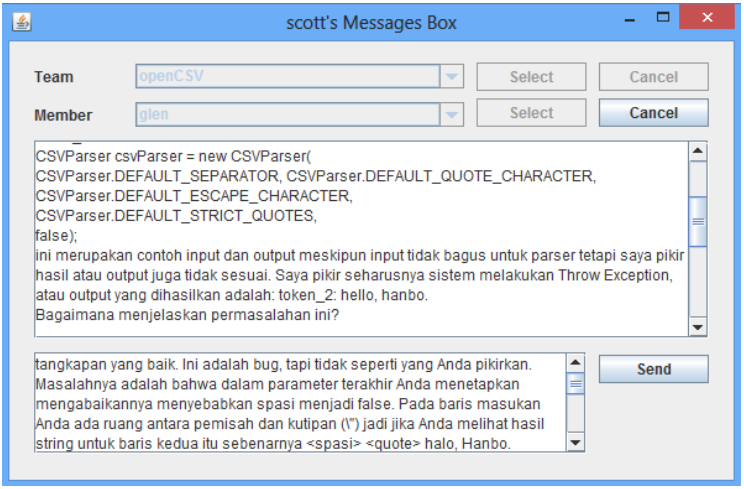
5.2.1.5. Pengujian Fitur Membalas Pesan

Pengujian fitur membalas pesan dilakukan dengan menggunakan data uji aplikasi pertama. Pada pengujian ini data uji yang diperlukan adalah nama tim proyek, nama anggota dan teks pesan. Skenario pengujian fungsionalitas sistem dapat dilihat pada Tabel 5.6. Hasil tampilan program saat pengujian untuk tampilan pengirim dapat dilihat pada Gambar 5.7 dan tampilan penerima dapat dilihat pada Gambar 5.8.

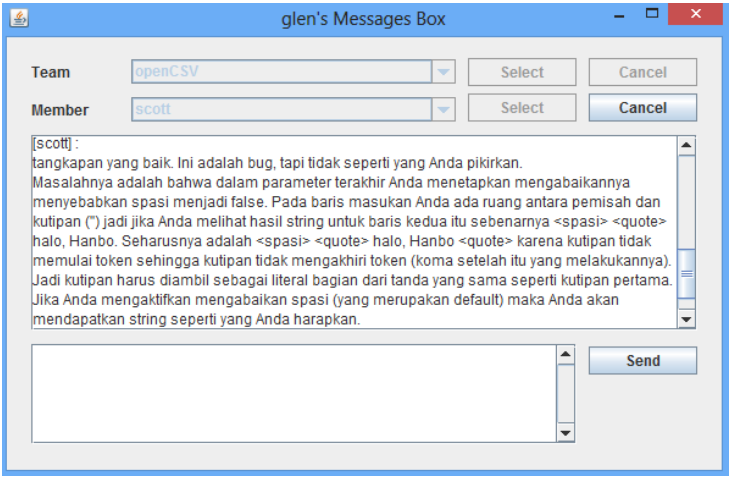
Tabel 5.6 Pengujian Fitur Membalas Pesan

ID	UJ.UC-0005
Referensi Kasus Penggunaan	UC-0005
Nama	Pengujian fitur membalas pesan.
Tujuan Pengujian	Menguji fitur untuk membalas pesan dari pengguna lain.
Skenario 1	**Mengirimkan pesan balasan dengan memilih nama tim proyek dan anggota tim.**
Kondisi Awal	*Plugin* dalam keadaan aktif. Pengguna yang akan dikirim pesan sedang aktif menggunakan *plugin*.
Data Uji	**Data uji diperoleh dari aplikasi pertama.**
Langkah Pengujian	1. Pengguna menekan *tab compose.* 2. Pengguna memilih tim pada *combobox* tim. 3. Pengguna menekan tombol *select*. 4. Pengguna memilih anggota pada *combobox* anggota. 5. Pengguna menekan tombol *select*. 6. Pengguna memasukkan teks pesan pada *textfield*. 7. Pengguna menekan tombol *send*.
Hasil Yang Diharapkan	Muncul pesan yang telah dikirimkan pada daftar riwayat pesan.
Hasil Yang Didapat	Muncul pesan yang telah dikirimkan pada daftar riwayat pesan pengirim seperti yang terlihat pada Gambar 5.7 dan pada daftar riwayat pesan penerima seperti pada Gambar 5.8.
Hasil Pengujian	Berhasil

Kondisi Akhir	Muncul pesan yang telah dikirimkan pada daftar riwayat pesan pengirim yang dapat dilihat pada Gambar 5.7 dan pada daftar riwayat pesan penerima pada Gambar 5.8.



Gambar 5.7 Hasil Pengujian Membalas Pesan Tampilan Pengirim



Gambar 5.8 Hasil Pengujian Membalas Pesan Tampilan Penerima

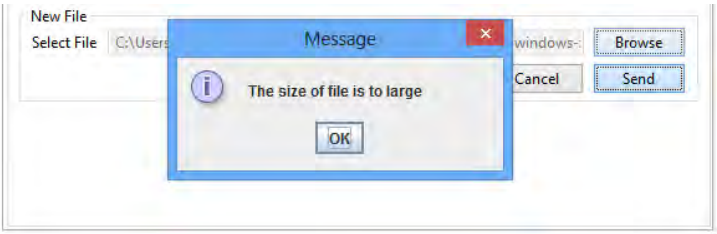
5.2.1.6. Pengujian Fitur Mengirimkan *File*

Pengujian fitur mengirimkan *file* dilakukan dengan menggunakan data uji yaitu *file* dengan ukuran yang berbeda. Pada pengujian ini data uji yang diperlukan adalah nama penerima dan *file*. Skenario pengujian fungsionalitas sistem dapat dilihat pada Tabel 5.7. Hasil tampilan pada aplikasi dapat dilihat pada Gambar 5.9.

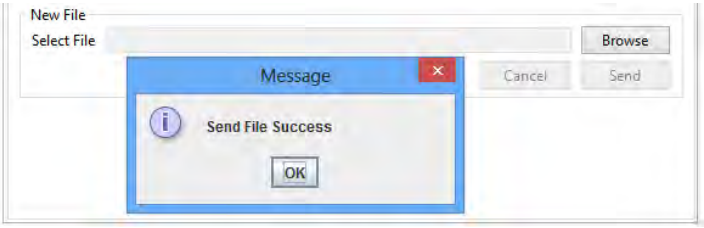
Tabel 5.7 Pengujian Fitur Mengirimkan *File*

ID	UJ.UC-0006
Referensi Kasus Penggunaan	UC-0006
Nama	Pengujian fitur mengirimkan *file*.
Tujuan Pengujian	Menguji fitur untuk mengirimkan *file* kepada pengguna lain.
Skenario 1	**Mengirimkan *file* dengan ukuran 30 Mb dan memilih anggota tim.**
Kondisi Awal	*Plugin* dalam keadaan aktif. Pengguna yang akan dikirim file sedang aktif menggunakan *plugin*.
Data Uji	**Data uji diperoleh dari aplikasi pertama.**
Langkah Pengujian	1. Pengguna menekan *tab File.* 2. Pengguna menekan tombol *Browse*. 3. Pengguna memilih *file* dengan ukuran 30 Mb. 4. Pengguna menekan tombol *Save*. 5. Pengguna menekan tombol *Send*. 6. Pengguna memilih anggota. 7. Pengguna menekan tombol *Ok*.
Hasil Yang Diharapkan	Muncul *message dialog* yang menampilkan pesan ukuran *file* terlalu besar.
Hasil Yang Didapat	Muncul *message dialog* yang menampilkan pesan ukuran *file* terlalu besar seperti yang terlihat pada Gambar 5.9.
Hasil Pengujian	Berhasil
Kondisi Akhir	Muncul *message dialog* yang menampilkan pesan ukuran *file* terlalu besar yang dapat dilihat pada Gambar 5.9.

Skenario 2	**Mengirimkan file dengan ukuran 5 Mb dan memilih anggota tim.**
Kondisi Awal	*Plugin* dalam keadaan aktif. Pengguna yang akan dikirim *file* sedang aktif menggunakan *plugin*.
Data Uji	**Data uji diperoleh dari aplikasi pertama.**
Langkah Pengujian	1. Pengguna menekan *tab File.* 2. Pengguna menekan tombol *Browse*. 3. Pengguna memilih *file* dengan ukuran 5 Mb. 4. Pengguna menekan tombol *Save*. 5. Pengguna menekan tombol *Send*. 6. Pengguna memilih anggota. 7. Pengguna menekan tombol *Ok*.
Hasil Yang Diharapkan	Muncul *message dialog* yang menampilkan pesan *file* berhasil dikirim.
Hasil Yang Didapat	Muncul *message dialog* yang menampilkan pesan *file* berhasil dikirim seperti yang terlihat pada Gambar 5.10.
Hasil Pengujian	Berhasil
Kondisi Akhir	Muncul *message dialog* yang menampilkan pesan *file* berhasil dikirim yang dapat dilihat pada Gambar 5.10.



Gambar 5.9 Hasil Pengujian Fitur Mengirimkan *File* Untuk Skenario Pertama



Gambar 5.10 Hasil Pengujian Fitur Mengirimkan *File* Untuk Skenario Kedua

5.2.2. Pengujian Klasifikasi Pesan

Pengujian fitur klasifikasi pesan dilakukan untuk menguji teks pesan sebagai masukan dan penerima pesan sebagai keluaran. Pada pengujian ini data uji yang diperlukan adalah tim proyek, deskripsi dari *library* dan *component* dari tim proyek, anggota tim dan teks pesan.

Untuk pengujian ini digunakan studi kasus aplikasi dari *sourceforge* [17],[18],[19]. Diambil tiga buah aplikasi sebagai bahan pengujian seperti yang dapat dilihat pada Tabel 5.1. Dari ketiga aplikasi tersebut dilakukan pengolahan informasi dan menerjemahkan deskripsi atau pesan dari Bahasa Inggris ke Bahasa Indonesia. Informasi yang diambil sebagai bahan pengujian adalah daftar pengembang aplikasi, daftar *library* dan *component*, pembagian *library* dan *component* setiap pengembang serta daftar pesan yang berkaitan dengan aplikasi.

Untuk aplikasi pertama, daftar pengembang dan keahlian dapat dilihat pada Tabel 5.8, daftar *library* dapat dilihat pada Tabel 5.9, daftar *component* dapat dilihat pada Tabel 5.10 dan daftar pesan dapat dilihat pada Tabel 5.11. Hasil pengujian klasifikasi pesan untuk aplikasi pertama dapat dilihat pada Tabel 5.12.

Untuk aplikasi kedua, daftar pengembang dan keahlian dapat dilihat pada Tabel A.1, daftar *library* dapat dilihat pada Tabel A.2, daftar *component* dapat dilihat pada Tabel A.3 dan daftar pesan dapat dilihat pada Tabel A.4. Hasil pengujian klasifikasi pesan untuk aplikasi kedua dapat dilihat pada Tabel A.5.

Untuk aplikasi ketiga, daftar pengembang dan keahlian dapat dilihat pada Tabel A.6, daftar *library* dapat dilihat pada Tabel A.7, daftar *component* dapat dilihat pada Tabel A.8 dan daftar pesan dapat dilihat pada Tabel A.9. Hasil pengujian klasifikasi pesan untuk aplikasi kedua dapat dilihat pada Tabel A.10.

Tabel 5.8 Daftar Pengembang pada Aplikasi Pertama

No	Nama	Pembagian Keahlian
1	Scott	Writer, Reader, Parser, java.io, java.util,
2	Glen	Writer, java.io, java.sql, java.beans
3	Sean	Writer, java.util, java.sql
4	Synszatana	Reader, Parser, java.io, java.util
5	Romanda	Parser, java.util
6	Kyle	Reader, java.beans

Tabel 5.9 Daftar dan Deskripsi *Library* pada Aplikasi Pertama

No	Nama	Deskripsi *Library*
1	*Java. io*	Menyediakan masukan dan keluaran melalui aliran data, serialisasi dan sistem *file*. Digunakan *FileReader* untuk membaca *file* karakter, *IOException* untuk pengecualian dari I/O, *StringWriter* untuk membangun sebuah *string*, *BufferedReader* untuk membaca teks dari aliran karakter yang masuk.
2	*java. util*	Berisi kerangka kerja koleksi, kelas koleksi warisan, model kejadian, tanggal dan waktu, internasionalisasi, dan kelas utilitas lainnya. Digunakan juga *arraylist* dan *list* untuk menyimpan data dalam koleksi yang urut dan *iterator* dalam koleksi. *Map* untuk mengimplementasikan antarmuka *Set* yang didukung oleh tabel *hash* atau *HashMap*.
3	*java. sql*	Mengakses dan memproses data yang tersimpan dalam *database* relasional. Digunakan *ResultSet* atau tabel data yang mewakili hasil *set database*, yang biasanya dihasilkan setelah mengeksekusi pernyataan *database*. *SQLException* menyediakan informasi tentang kesalahan akses *database*.
4	*java. beans*	Memudahkan untuk menggunakan kembali komponen perangkat lunak. Digunakan *PropertyDescriptor* yang

No	Nama	Deskripsi *Library*
		menggambarkan salah satu properti yang mengekspor Java Bean melalui sepasang metode pengakses. Sebuah *implementor bean* yang ingin memberikan informasi eksplisit tentang *bean* dapat mengimplementasikan *BeanInfo* untuk memberikan informasi eksplisit tentang *method, properties*, dan *event* dari *bean*.

Tabel 5.10 Daftar dan Deskripsi *Component* pada Aplikasi Pertama

No	Nama	Deskripsi *Component*
1	*Parser*	Untuk menguraikan *file* csv digunakan kelas *CSVParser* yang mengimplementasikan pemecahan baris tunggal kedalam bidang. *CSVParser* menggunakan koma sebagai pemisah. Digunakan *CSVParserBuilder* untuk membangun *CSVParser* seperti untuk pemisah, tanda kutip, dan spasi atau jarak. Terdapat beberapa konstruktor jika ingin menggunakan pemisah sendiri dan karakter tanda kutip.
2	*Reader*	Untuk membaca dan menguraikan *file* csv digunakan kelas *CSVReader* dengan pola perulangan yang membaca setiap baris atau dengan membaca seluruh *file* kedalam sebuah *list* dengan setiap elemen menjadi *token string* dengan metode *readAll.* Terdapat beberapa konstruktor jika ingin menggunakan pemisah sendiri dan karakter tanda kutip. Digunakan *CSVReaderBuilder* untuk membangun *CSVReader* seperti mengatur *reader*, nomor baris untuk dilewati dan *parser* yang digunakan. Selain itu juga ada kelas *BeanToCSV* yang digunakan untuk ekspor isi dari java beans menjadi *file* csv dan sebaliknya dengan kelas *CSVToBean*.
3	*Writer*	Terdapat kelas *CSVWriter* untuk menulis kedalam *file* csv dengan menulis setiap baris atau menulis seluruh daftar *string* kedalam *file* csv dengan metode *writeAll*. Terdapat beberapa konstruktor jika ingin menggunakan pemisah sendiri dan karakter tanda kutip. Hasil dari pernyataan *database* dapat

No	Nama	Deskripsi *Component*
		dimasukkan kedalam *file* csv dengan menambahkan parameter *resultset* pada kelas *CSVWriter*.

Tabel 5.11 Daftar Pesan Untuk Aplikasi Pertama

No	Deskripsi Pesan
1	Saya menulis sebuah *file* csv menggunakan *CSVWriter*. Salah satu bidang memiliki teks seperti: Joe mengatakan, *'Blah \n blah blah.'*. Untuk hasil pada baris baru, dengan *CSVWriter* dapat menulis dengan benar dan dengan *CSVReader* dan Excel dapat membaca dengan benar. Tapi dengan *CsvToBean* salah memotong teks di baris baru. Saya sudah mensiapkan *file* yang menunjukkan kesalahan.
2	Dalam *output* CSV melewatkan baris pertama dari *ResultSet*, kecuali anda menambahkan pernyataan *rs.beforeFirst()*. Sebelum memanggil *csv.writeAll*, *output* csv melewatkan baris pertama dari *ResultSet rs*. Hal ini berlaku meskipun nilai *tf Boolean*, termasuk menghilangkan baris *header*, benar atau salah.
3	*input: String line = "hi, \"hello, hanbo\", 30";* *output:* *token_1:hi* *token_2: "hello, hanbo* *token_3: 30* *CSVParser csvParser = new CSVParser(* *CSVParser.DEFAULT_SEPARATOR,* *CSVParser.DEFAULT_QUOTE_CHARACTER,* *CSVParser.DEFAULT_ESCAPE_CHARACTER,* *CSVParser.DEFAULT_STRICT_QUOTES,* *false);* Ini merupakan contoh *input* dan *output* meskipun *input* tidak bagus untuk *parser* tetapi saya pikir hasil atau *output* juga tidak sesuai. Saya pikir seharusnya sistem melakukan *Throw Exception*, atau *output* yang dihasilkan adalah: *token_2: hello, hanbo*. Bagaimana menjelaskan permasalahan ini?
4	Aku tidak bisa mendapatkan *CSVReader* untuk menampilkan teks yang dikodekan dengan benar. Saya mencoba untuk menambahkan sesuatu seperti berikut di kelas saya dan itu tidak

No	Deskripsi Pesan
	bekerja. Ada gagasan, saya bukan ahli Java dan diperparah oleh ini yang tampaknya cukup mudah? *public CSVReader ReadFile() throws IOException {* *List<String[]> rawList = new ArrayList<String[]>();* *mlsFile = MLSFileName.GetFileName(mlsFileType, fileLanguage);* *fullFilePath = Util.DoSetFullPath(this.filePath, mlsFile);* *this.headers.clear();* *this.reader = new CSVReader(new InputStreamReader(new FileInputStream(fullFilePath), "UTF-8"));* *rawList = reader.readAll();* *doSetListData(rawList);* *return reader;}*
5	Saya menggunakan OpenCSV 2.3. Menangani penguraian lebih dari 600.000 catatan dalam sebuah *file* csv semua baik-baik saja kecuali untuk rekaman pertama di bawah ini: *"id","email","profile_id","first_name","last_name"* *"475592","rbst218@yahoo.com","bc1er1163","\""CHia Sia Ta","""* *"12345","kazem.naderi@tt.com","bc234a","kazem","naderi"* Tampaknya OpenCSV jadi bingung dengan nilai *first_name* (*disymmetric* tanda kutip ganda di awal nilai). Anda dapat menyimpan di atas dua baris dalam *file* katakanlah *badProfiles.csv* dan kemudian menggunakan kode di bawah ini mencoba untuk mengurai catatan. Anda akan melihat bahwa itu tidak bisa mengurai dengan benar. *public static List<UpakneeUserBean> parseCvSIntoBeans2(String filePath)* *throws FileNotFoundException {* *CSVReader reader = new CSVReader(new FileReader(filePath));* *HeaderColumnNameMappingStrategy<UserBean> strat = new HeaderColumnNameMappingStrategy<UserBean>();* *strat.setType(UserBean.class);* *CsvToBean<UserBean> csv = new CsvToBean<UserBean>();* *return csv.parse(strat, reader);}*

No	Deskripsi Pesan
6	Ketika bidang berisi karakter pemisah dan ketika *quote* diatur ke *NO_QUOTE_CHARACTER*, separator tersebut tidak keluar dengan OpenCSV. Idealnya, saya ingin mengutip bidang hanya ketika diperlukan atau ketika mereka mengandung karakter khusus seperti pemisah kolom.
7	Dengan *CSVWriter.writeAll, rs ResultSet* bekerja baik untuk kebutuhan saya kecuali bahwa itu tidak mengenali nama kolom alias. Saat menulis pernyataan SQL saya menentukan alias kolom seperti dalam *"column_name SELECT AS column_alias FROM table_name"* tapi *CSVWriter* mengabaikan alias ketika menulis *file* CSV pada baris *header*. Perbaikan untuk ini adalah perubahan kecil pada baris 47 dari *ResultSetHelperService*: *names.add (metadata.getColumnName (i +1));* seharusnya *names.add (metadata.getColumnLabel (i +1));* Ini akan bekerja tanpa memperhatikan apakah pengguna menentukan alias kolom atau tidak karena jika tidak ada alias ditentukan, *getColumnLabel()* mengembalikan nama kolom sebagai gantinya.
8	Saya menggunakan mysql *"select into outfile"* untuk membuat gaya csv *textfile.* Contoh: *SQL: mysql>* *select* ***** *into outfile '/tmp/records.out'* *FIELDS TERMINATED BY ',' ENCLOSED BY '"' ESCAPED BY '\\' LINES TERMINATED BY '\n'* *from data2unload;* *Saya sudah instansiasi OpenCSV sebagai p = new CSVParser (',', '"', '\\');* Semuanya baik sampai tabel sumber memiliki *NULLs*. Ini dikodekan sebagai *\N* tanpa karakter penutup dalam *file* csv. contoh: *"varchar\", simple\\","N","2011-06-06 00:06:38", \N \N \N* saat ini OpenCSV mengembalikan string *'N'* sebagai nilai untuk bidang tersebut. Apakah sudah ada penanganan *NULL* yang diimplementasikan? Aku tidak bisa menemukan apa-apa. Seberapa susah itu untuk menambahkan penanganan khusus atau mengembalikan objek *null* untuk *\N* tanpa karakter penutup?

No	Deskripsi Pesan
9	Karakter *escape* standar berbeda antara *CsvParser* dan *CsvWriter* yang dapat menyebabkan masalah jika anda hanya menggunakan pengaturan *default* pada baca atau tulis tanpa menyesuaikannya. Untuk konsistensi dan kegunaan ini harus disesuaikan. - *CSVWriter* *public static final char DEFAULT_ESCAPE_CHARACTER = '"';* *public static final char DEFAULT_QUOTE_CHARACTER = '"';* - *CSVParser* *public static final char DEFAULT_ESCAPE_CHARACTER = '\';* *public static final char DEFAULT_QUOTE_CHARACTER = '"';*
10	Jika karakter *escape* diatur ke *NULL_CHARACTER*, yaitu *\x0*, seharusnya tidak boleh ada karakter escape menurut *https://sourceforge.net/p/opencsv/support-requests/5/.* Sebaliknya *\x0* menjadi karakter *escape*. Lihat pada *CSVParser.java* baris 216. Hal ini menyebabkan masalah jika data masukan berisi nol *byte* diikuti dengan kutipan: *\ x0"* Jika urutan ini ditulis ke *file* CSV, bidang akan dikutip dan kutipan akan menjadi karakter *escape*, menghasilkan urutan *\x0""*. Ketika *\x0""* dibaca dengan *escape = NULL_CHARACTER*, kutipan pertama lolos, tapi yang kedua bidang dikutip sebelum waktunya, mengakibatkan kesalahan penguraian.

Tabel 5.12 Hasil Pengujian Klasifikasi Pesan Untuk Aplikasi Pertama

Pesan	*Library* atau *component* yang terpilih	Pengguna Penerima Pesan
1	*Writer*	Scott, glen, sean
2	*Writer*	Scott, glen, sean
3	*Java.io*	Scott, glen, synzsatana
4	*Java.io*	Scott, glen, synzsatana
5	*Writer*	Scott, glen, sean
6	*Parser*	Scott, synzsatana
7	*Java.io*	Scott, glen, synzsatana

Pesan	*Library* atau *component* yang terpilih	Pengguna Penerima Pesan
8	*Java.io*	Scott, glen, synzsatana
9	*Java.io*	Scott, glen, synzsatana
10	*writer*	Scott, glen, sean

5.3. Evaluasi Pengujian

Pada subbab ini akan diberikan hasil evaluasi dari pengujian-pengujian yang telah dilakukan. Evaluasi yang diberikan meliputi evaluasi pengujian kebutuhan fungsional dan evaluasi hasil klasifikasi pesan.

5.3.1. Evaluasi Pengujian Fungsionalitas

Rangkuman mengenai hasil pengujian fungsionalitas dapat dilihat pada Tabel 5.13. Berdasarkan data pada tabel tersebut, semua skenario pengujian berhasil dan program berjalan dengan baik. Sehingga bisa disimpulkan bahwa fungsionalitas dari program bisa bekerja sesuai dengan yang diharapkan.

Tabel 5.13 Rangkuman Hasil Pengujian

ID	Nama	Skenario	Hasil
UJ.UC-0001	Pengujian fitur membuat tim proyek perangkat lunak	Skenario 1	Berhasil
		Skenario 2	Berhasil
UJ.UC-0002	Pengujian fitur mengikuti tim proyek perangkat lunak	Skenario 1	Berhasil
UJ.UC-0003	Pengujian fitur mengirimkan pesan baru	Skenario 1	Berhasil
		Skenario 2	Berhasil
UJ.UC-0004	Pengujian fitur melihat riwayat pesan	Skenario 1	Berhasil
UJ.UC-0005	Pengujian fitur membalas pesan	Skenario 1	Berhasil
UJ.UC-0006	Pengujian fitur mengirimkan *file*	Skenario 1	Berhasil
		Skenario 2	Berhasil

5.3.2. Evaluasi Pengujian Klasifikasi Pesan

Sistem merupakan kakas bantu atau sebuah *plugin* untuk Eclipse versi 3.5 hingga versi 3.5 ke atas. Kakas telah diuji dengan versi Eclipse 3.4 dan menunjukkan bahwa *plugin* tidak dapat dijalankan. Sedangkan untuk Eclipse versi terbaru, *plugin* dapat berjalan dan berfungsi dengan baik.

Sistem melakukan klasifikasi pesan atau *query* berdasarkan pada dokumen atau deskripsi *library* dan *component* dari sebuah tim proyek perangkat lunak. Jika sebuah *query* memiliki kesamaan dengan sebuah dokumen, hal ini menunjukkan bahwa kata dalam *query* terdapat pada dokumen tersebut.

Setiap dokumen memiliki tingkat kesamaan yang berbeda, oleh karena itu sistem mengambil tingkat kesamaan tertinggi dari seluruh dokumen. Dari hasil tersebut, sistem mengirimkan pesan kepada pengguna atau pengembang yang memiliki keahlian sesuai dengan *library* atau *component* yang terpilih. Tetapi bisa saja sebuah *query* tidak memiliki kesamaan dengan semua dokumen sehingga sistem tidak dapat melakukan klasifikasi. Jika tidak terdapat kesamaan, maka sistem tidak akan mengirimkan pesan dari pengguna.

Untuk mengukur tingkat *reliability* dari sistem, digunakan metode *cohen's kappa* [16]. Penjelasan lebih terperinci dapat dilihat pada bab 2.7. Hasil perhitungan kemungkinan yang didapat antara sistem dan pengembang untuk aplikasi pertama dapat dilihat pada Tabel 5.14, untuk aplikasi kedua dapat dilihat pada Tabel 5.15 dan untuk aplikasi ketiga dapat dilihat pada Tabel 5.16.

Tabel 5.14 Hasil Pengujian Aplikasi Pertama Dalam Daftar Hasil Pengamatan

		Pengembang							
		Library				*Component*			Jumlah
Sistem		1	2	3	4	1	2	3	
Library	1	3	0	0	0	0	0	2	5
	2	0	0	0	0	0	0	0	0
	3	0	0	0	0	0	0	0	0
	4	0	0	0	0	0	0	0	0
Component	1	0	0	0	0	1	0	0	1
	2	0	0	0	0	0	0	0	0
	3	0	0	0	0	0	0	4	4
Jumlah		3	0	0	0	1	0	6	10

Tabel 5.15 Hasil Pengujian Aplikasi Kedua Dalam Daftar Hasil Pengamatan

		Pengembang									
		Library				*Component*					Jumlah
Sistem		1	2	3	4	1	2	3	4	5	
Library	1	0	0	0	0	0	0	0	0	0	0
	2	0	0	0	0	0	0	0	0	0	0
	3	0	0	1	0	0	0	0	0	0	1
	4	0	0	0	0	0	0	0	0	0	0
Com-ponent	1	0	0	0	0	0	0	0	0	0	0
	2	0	0	0	0	0	0	1	0	1	2
	3	0	0	0	0	0	0	2	0	0	2
	4	0	0	0	0	0	0	2	2	0	4
	5	0	0	0	0	0	0	0	0	1	1
Jumlah		0	0	1	0	0	0	5	2	2	10

Tabel 5.16 Hasil Pengujian Aplikasi Ketiga Dalam Daftar Hasil Pengamatan

		Pengembang							
		Library			*Component*				Jumlah
Sistem		1	2	3	1	2	3	4	
Library	1	1	0	0	0	0	0	0	1
	2	0	1	0	0	0	0	0	1
	3	0	0	2	0	0	0	0	2
Component	1	0	0	0	0	0	0	0	0
	2	1	0	1	0	4	0	0	6
	3	0	0	0	0	0	0	0	0
	4	0	0	0	0	0	0	0	0
Jumlah		2	1	3	0	4	0	0	10

Jumlah yang sesuai dari hasil yang didapat antara pengembang dan sistem atau kesepakatan yang diamati, kesepakatan yang diharapkan dan nilai *Kappa* dapat dilihat pada Tabel 5.17.

Tabel 5.17 Daftar Tingkat *Reliability* Sistem Untuk Tiga Data Uji

Nomor Aplikasi	*Observed Agreement*	*Expected Agreement*	*Kappa*
1	0,8	0,40	0,66
2	0,6	0,21	0,49
3	0,8	0,33	0,70

Setelah dilakukan percobaan pada ketiga aplikasi didapatkan rata-rata hasil *Kappa* 0,61. Berdasarkan tabel interpretasi *Kappa* pada Tabel 2.2 bisa disimpulkan bahwa hasil klasifikasi pesan yang didapatkan oleh sistem atau tingkat *reliability* dari sistem adalah baik.

BAB VI
KESIMPULAN DAN SARAN

Bab ini menjelaskan kesimpulan yang diambil selama pengerjaan Tugas Akhir serta saran-saran tentang pengembangan yang dapat dilakukan terhadap Tugas Akhir ini di masa yang akan datang.

6.1. Kesimpulan

Dari hasil yang didapat selama proses perancangan, implementasi dan pengujian perangkat lunak yang dilakukan, dapat diambil kesimpulan sebagai berikut:

1. Sistem dapat terintegrasi dengan IDE Eclipse versi 3.5 hingga versi 3.5 ke atas. Kompatibilitas untuk versi sebelumnya pada *plug-in* ini tidak bisa berjalan dengan baik karena pada waktu pengembangan diharuskan untuk memilih versi tertentu. Berdasarkan hasil pengujian fungsionalitas dapat disimpulkan bahwa fungsionalitas sistem berjalan dengan baik.
2. Sistem dapat melakukan klasifikasi pesan berdasarkan keahlian pengembang lain. Berdasarkan hasil pengujian klasifikasi pesan menggunakan metode *cohen's kappa* dapat disimpulkan bahwa sistem memiliki tingkat *reliability* yang baik.

6.2. Saran

Untuk pengembangan perangkat lunak dimasa mendatang, terdapat beberapa saran agar perangkat lunak ini dapat diperbaiki dan dikembangkan dengan lebih baik lagi. Saran tersebut diantaranya adalah:

1. Menambahkan fitur untuk mendeteksi aktivitas dari pengguna sehingga pada saat mengirimkan pesan sistem melihat aktivitas dari penerima terlebih dahulu. Jika tingkat aktivitas penerima rendah, pesan akan dikirimkan kepada penerima tersebut. Sedangkan jika tingkat aktivitas penerima tinggi, maka pesan

akan ditahan terlebih dahulu oleh *server* sampai tingkat aktivitas penerima rendah.

2. Menggunakan algoritma atau metode pencarian informasi selain metode *Latent Semantic Indexing* untuk melakukan klasifikasi pesan. Sehingga dapat dibandingkan antara metode baru dengan metode *Latent Semantic Indexing* dan didapatkan metode terbaik dalam melakukan klasifikasi pesan.
3. Koneksi atau komunikasi antara *client* dengan *server* dapat diimplementasikan tanpa menggunakan *socket*.

DAFTAR PUSTAKA

[1] Nishinaka, Y., Asada, M., Yamamoto, Y., and Ye, Y., "Please STeP_IN: A Socio-Technical Platform for in situ Networking".

[2] Manning, C., D., Raghavan, P., Schutze, H., Introduction to Information Retrieval, Cambridge: Cambridge University Press, 2008.

[3] L. Vogel, "Eclipse IDE – Tutorial," 10 April 2013. [Online]. Available: http://www.vogella.com/tutorials/Eclipse/article.html. [Accessed 3 March 2014].

[4] J. Weinstein, "Rich Client Platform/FAQ," 8 February 2013. [Online]. Available: http://wiki.eclipse.org/Rich_Client_Platform/FAQ. [Accessed 20 June 2014].

[5] E. Foundation, "JDT - Java development tools," 2014. [Online]. Available: http://projects.eclipse.org/projects/eclipse.jdt. [Accessed 21 June 2014].

[6] E. Foundation, "PDE," 2014. [Online]. Available: http://www.eclipse.org/pde/. [Accessed 21 June 2014].

[7] Buyya, R., Selvi, S. T., Chu, X., "Socket Programming," in *Object-Oriented Programming with JAVA*, Noida, Tata McGraw-Hill, 2009, pp. 346-363.

[8] "History of the Java™ programming language," 13 June 2014. [Online]. Available: http://en.wikibooks.org/wiki/Java_Programming/History. [Accessed 22 June 2014].

[9] tutorialspoint, "Java - Multithreading," 2014. [Online]. Available: http://www.tutorialspoint.com/java/java_multithreading.htm. [Accessed 22 June 2014].

[10] Oracle, "I/O Streams," 2014. [Online]. Available: http://docs.oracle.com/javase/tutorial/essential/io/streams.html. [Accessed 21 June 2014].

[11] Oracle, "Byte Streams," 2014. [Online]. Available: http://docs.oracle.com/javase/tutorial/essential/io/bytestreams.html. [Accessed 21 June 2014].

[12] Oracle, "Character Streams," 2014. [Online]. Available: http://docs.oracle.com/javase/tutorial/essential/io/charstreams.html. [Accessed 21 June 2014].

[13] tutorialspoint, "Java - Serialization," 2014. [Online]. Available: http://www.tutorialspoint.com/java/java_serialization.htm. [Accessed 22 June 2014].

[14] tutorialspoint, "Java - Networking (Socket Programming)," 2014. [Online]. Available: http://www.tutorialspoint.com/java/java_networking.htm. [Accessed 22 June 2014].

[15] J. Hicklin, "JAMA : A Java Matrix Package," 23 November 2012. [Online]. Available: http://math.nist.gov/javanumerics/jama/. [Accessed 14 June 2014].

[16] A. J. Viera and J. M. Garret, "Understanding Interobserver Agreement: The Kappa Statistic," *Family Medicine,* vol. 37, no. 5, pp. 360-363, 2005.

[17] SourceForge, "opencsv," 10 July 2014. [Online]. Available: http://sourceforge.net/projects/opencsv/. [Accessed 14 July 2014].

[18] SourceForge, "aTunes," 22 June 2014. [Online]. Available: http://sourceforge.net/projects/atunes/. [Accessed 14 July 2014].

[19] SourceForge, "TuxGuitar," 20 July 2013. [Online]. Available: http://sourceforge.net/projects/tuxguitar/. [Accessed 14 July 2014].

BIODATA PENULIS

Penulis, Anugerah Firdaus, lahir di kota Martapura pada tanggal 10 Mei 1992. Penulis adalah anak ketiga dari tiga bersaudara dan dibesarkan di kota Sidoarjo, Jawa Timur.

Penulis menempuh pendidikan formal di SDN Landasan Ulin Timur 4 (1998-2000), SDN Pabean 1 (2000-2004), SMPN 1 Waru (2004-2007), SMA Muhammadiyah 2 Surabaya (2007-2010). Pada tahun 2010, penulis memulai pendidikan sarjana di Jurusan Teknik Informatika Fakultas Teknologi Informasi Institut Teknologi Sepuluh Nopember Surabaya, Jawa Timur.

Di jurusan Teknik Informatika, penulis mengambil bidang minat Rekayasa Perangkat Lunak dan memiliki ketertarikan di bidang *Web Programming*, *Game Development*, dan *Desktop Software Development*. Penulis juga aktif dalam organisasi kemahasiswaan seperti Himpunan Mahasiswa Teknik Computer (HMTC). Penulis dapat dihubungi melalui alamat *email* anugerahfirdaus92@gmail.com

LAMPIRAN A. DATA DAN HASIL PENGUJIAN

Tabel A. 1 Daftar Pengembang pada Aplikasi Kedua

No	Nama	Pembagian *Library* dan *Component*
1	Alex	*Java.awt, java.net, player, internet*
2	Sylvain	*Java.awt, playlist*
3	Thomas	*Java.awt, audio cd*
4	Bertrand	*Java fx, player*
5	philip	*Java.net, internet*
6	roeland	*Java fx, audio cd*
7	aekold	*Javax.swing, GUI*
8	stefan	*Java fx, playlist*
9	laurent	*Javax.swing, GUI*
10	tobias	*Javax.swing, GUI*

Tabel A. 2 Daftar dan Deskripsi *Library* Aplikasi Kedua

No	Nama	Deskripsi *Library*
1	*Java. awt*	Berisi semua kelas untuk membuat antarmuka dan komponen awt. Kelas yang digunakan adalah *component* atau sebuah objek yang ditampilkan pada layar dan dapat berinteraksi dengan pengguna, *frame* untuk menampilkan jendela, *color* untuk enkapsulasi warna sesuai dengan RGB, *font* digunakan untuk membuat teks. Terdapat juga *awt.event* yang berisi kelas untuk menangani berbagai jenis *event* dari komponen AWT seperti *actionEvent* jika *event* dari komponen telah terjadi, *keyEvent* jika *keystroke* terjadi pada komponen, *mouseEvent* dan *windowEvent* jika *mouse action* dan *window action* terjadi pada komponen.
2	*java. net*	Menyediakan kelas untuk mengimplementasikan aplikasi jaringan. Digunakan kelas URL yang merepresentasikan *Uniform Resource Locator*, yaitu pointer ke sumber daya di *World Wide Web*.
3	*Javax. swing*	Menyediakan satu set komponen yang dapat bekerja sama pada semua *platform*. Kelas yang digunakan adalah *actionMap* untuk memetakan dari objek ke *action, keystroke* yang merepresentasikan *key action*

No	Nama	Deskripsi *Library*
		dari *keyboard, swingUtilities* merupakan koleksi metode utilitas untuk *swing*, dan *JComponent* atau kelas dasar dari komponen *swing* yang lain. Komponen dari *swing* yang digunakan adalah *JComboBox, JList, JFrame, JLabel, JProgressBar, JSplitPane, JDialog, JTable.*
4	*Java FX*	Seperangkat paket grafis dan media yang memungkinkan untuk merancang, membuat, menguji, *debug*, dan menyebarkan aplikasi *rich client* yang beroperasi di seluruh *platform* yang beragam dan menampilkan informasi dalam antarmuka pengguna yang *modern* yang menampilkan *audio, video*, grafik, dan animasi. Kelas yang digunakan adalah *Scene, collection, event, fxml, application, embed*, dan *util*.

Tabel A. 3 Daftar dan Deskripsi *Component* pada Aplikasi Kedua

No	Nama	Deskripsi *Component*
1	*GUI*	Antarmuka menggunakan komponen *SWING* dengan *frame* tunggal dimana *playlist* berada dibawah *navigator*. Layar tampilan memiliki mode untuk layar penuh dan tampilan standar dengan semua kontrol dan fitur. Setiap elemen *window*, seperti: *navigator, playlist*, informasi konteks, ditampilkan sebagai jendela terpisah dan setiap jendela dapat diatur seperti yang diinginkan. Warna pada tampilan aTunes dapat dirubah menggunakan tema. Sistem *tray icon* untuk mengontrol pemutar atau *player*. *Navigator* memungkinkan untuk melihat musik yang dikategorikan berdasarkan artis, album, atau *genre* dalam *folder*. Kemudian juga dapat mengakses cepat ke lagu dan album serta terdapat fungsi *Option Filter* untuk menemukan dengan mudah artis, album, atau *genre.*
2	*Player*	Dapat membaca lagu dengan *tag mp3, ogg, flac, wma, mp4, ra, rm*. Menulis *tag mp3, ogg, flac, wma, mp4. Window Tag Editor* dan *auto tag* yaitu set nomor trek, *genre*, lirik secara otomatis. Aplikasi ini juga dilengkapi dengan *radio online,* kontrol volume, fungsi

No	Nama	Deskripsi *Component*
		mute atau diam, fungsi karaoke, *equalizer*, pilihan *shuffle* dan mengulangi lagu. Mesin pemutar menggunakan mplayer untuk semua lingkungan sistem dan xine untuk sistem Linux.
3	*Play-list*	Dapat membuat *playlist* atau daftar lagu yang dimainkan. Menyediakan dukungan untuk ribuan lagu dan untuk beberapa *playlist* sekaligus. *Playlist* ditunjukkan dengan beberapa kolom, seperti: judul, artis, album, *genre*, panjang, dan nomor trek. *Playlist* tersebut dapat diurutkan berdasarkan salah satu kolom dan terdapat mode *Drag and Drop* dimana lagu-lagu dapat ditarik dari *navigator* atau dari sistem *file*. Terdapat juga dukungan untuk membuka atau menyimpan *m3u*.
4	*Inter-net*	Memiliki fitur internet untuk melakukan pencarian artis dalam halaman web yang berbeda: YouTube, Google Video, Wikipedia, dll. Kemudian mendapatkan informasi dari radio *online*, seperti Last.fm, ketika lagu sedang dimainkan, yaitu informasi album, album artis dan artis yang serupa serta pembaruan profil Last.fm ketika terdapat lagu yang sedang dimainkan akan diserahkan ke profil Last.fm pengguna. Terdapat juga informasi lirik saat lagu sedang dimainkan maka secara otomatis aTunes akan menunjukkan lirik lagu tersebut. Terdapat juga fitur *podcast* dimana pengguna dapat berlangganan *feed podcast* favorit dan mendengarkan *podcast* tersebut dalam aTunes.
5	*Audio CD*	Terdapat alat untuk ekstrak lagu dari cd menggunakan cdda2wav, flac dan oggenc. Untuk ekstrak cd diperlukan untuk menempatkan sebuah *cd audio* ke komputer. Pengguna dapat memilih trek lagu apa yang akan di ekstrak dan mendapatkan nama lagu otomatis dari Amazon. Terdapat juga dukungan tambahan untuk ekstrak cd dengan Nero AAC, FAAC kecuali Windows OS dan CDParanoia pada Mac OS X dan Linux.

Tabel A. 4 Daftar Pesan Untuk Aplikasi Kedua

No	Pesan
1	Saya sedang menjalankan WinXP pro dan memiliki Java 1.6.0 terpasang. Antarmuka pemutar terkunci setelah memainkan di mana saja dari satu menit sampai satu jam. Musik terus bermain, tapi layar antarmuka beku, begitu juga dengan tombol *toolbar*. Satu-satunya cara saya bisa menonaktifkan aTunes adalah untuk memaksa mengakhiri proses javaw.exe di *Task Manager*. Apakah ada orang lain punya masalah ini? Jika demikian, bagaimana saya bisa mencegah hal itu? Apakah ada konflik perangkat lunak yang bisa saya coba untuk hindari?
2	Ketika mengimpor CD, sekarang saya bisa memilih beberapa format selain WAV. Tapi itu tidak akan bekerja apapun format yang dipilih. Ketika saya pergi impor CD, kesalahan dicatat dalam *file*. Jika saya pilih pengaturan dan klik OK, tidak ada yang terjadi.
3	Saya memiliki *playlist* yang berisi *file* m3u. Yang harus saya lakukan adalah membuka di WMP atau winamp dan ini akan bekerja baik, tapi saya punya masalah ketika mencoba untuk memuatnya di aTunes. aTunes mencoba untuk memuat *playlist* sebagai: lokasi *file* m3u/isi di M3U. Sehingga bukan memuat musik sebagai: /192.168.1.3/music/blahblah/ Tetapi mencoba memuat musik saya sebagai: Users-Paul-Music-Playlist-192.168.1.3-musik-blahblah, dan jelas saja musik tidak ada dalam direktori di atas, sehingga tidak bisa memutar musik. Ada cara untuk memperbaiki ini?
4	Setiap kali aTunes dimatikan, tidak menyimpan *playlist* saat ini. Kemudian ketika aTunes dimulai lagi *playlist* hilang. Kemudian jika *playlist* disimpan secara manual, *playlist* itu harus dimuat secara manual. Jadi permintaan saya adalah: 1. Simpan *playlist* saat dimatikan. 2. Muat semua *playlist* yang dibuka sebelumnya pada saat memulai.
5	Saya sering mendengarkan *podcast* dan stasiun *radio internet.* Pesan berikut muncul ketika *podcast* berakhir, ketika saya berhenti memainkan dan menghapus *playlist*. Hal ini juga muncul ketika *podcast* yang tidak diunduh diputar.

No	Pesan
	net.sourceforge.atunes.kernel.modules.podcast.PodcastFeedEntry tidak dapat dilemparkan ke net.sourceforge.atunes.kernel.modules.repository variasi lain: net.sourceforge.atunes.kernel.modules.radio.Radio tidak dapat dilemparkan ke net.sourceforge.atunes.kernel.modules.repository.AudioFile. Hal ini muncul saat mengosongkan *playlist* setelah mendengarkan stasiun *radio*.
6	Aku sedang menjalankan aTunes pada Win7 dan saya mencoba untuk menambahkan semua stasiun Soma FM ke daftar radio. Saya telah mengikuti petunjuk di http://www.atunes.org/wiki/index.php?title=Propose_new_radio_stations, secara manual menambahkan rincian tetapi stasiun radio baru tidak muncul dalam daftar radio. Saya mencari dan tidak ada 'radio.xml' di folder pengaturan, tapi ada *file* 'preset_radios.xml'. Saya mencoba menambahkan stasiun baru secara langsung ke *file* tersebut tapi stasiun radio tersebut masih belum ditambahkan.
7	Jika anda sedang mendengarkan *file audio* yang panjang, seperti *podcast*, maka tidak dapat melompat ke waktu setelah sekitar 72 menit, bila menekan pada *timeline* setelah 72 menit, akan memutar *file* dari awal. Saya mendapatkan masalah ini dengan mendengarkan *podcast* Chaosradio, dengna 2 sampai 3 jam.
8	CD dengan karakter aksen, misalnya é, dalam artis atau nama album tidak dapat diekstrak. Mencoba dengan CD yang berbeda, beberapa bekerja beberapa tidak. Keberadaan é tampaknya menjadi titik utama. Saya melakukan dua pekerjaan untuk melakukan ekstrak CD: Saya menggunakan CDex untuk ekstrak CD dan CDex berhasil melakukan ekstrak CD yang oleh aTunes gagal. Saya melepaskan kabel jaringan dan berhasil,aTunes gagal mendapatkan info album sehingga tidak mendapatkan karakter aksen tetapi sukses ekstrak CD. Saya menggunakan aTunes pada Windows XP. Terlampir adalah *log file* saya. Menjelang akhir, anda akan melihat info *rip* ketika aTunes gagal ekstrak 'Journée d'Amerique' dari 'Richard Séguin'. Setelah informasi tersebut,

No	Pesan
	anda akan melihat info rip sukses. Ini adalah album yang sama tetapi dengan kabel jaringan dilepaskan sehingga Atune tidak bisa mendapatkan info album! Dengan salah satu CD dari saya dengan 'aksen', saya membuat gambar dan memasang gambar dengan Alkohol 52%. Atune juga gagal ekstrak gambar terpasang seperti gagal dengan CD yang sebenarnya.
9	Meningkatkan pilihan, baik *Shuffle* dan *Repeat*, sebagai berikut: *Shuffle Albums*: Ketika lagu terakhir dari album saat ini selesai, pilih lagu yang pertama atau nomor terendah dari album lain dalam *playlist*. *Repeat Album*: Ketika lagu terakhir dari album saat ini selesai, pilih lagu yang pertama atau nomor terendah dari album yang sama dalam *playlist.* *Repeat Playlist*: Ketika lagu terakhir dari *playlist* selesai bermain, pilih lagu pertama dari *playlist.*
10	Kebanyakan orang hanya meninggalkan *Equalizer* mereka pada pengaturan *default* yang terdengar tepat untuk *genre* tertentu musik yang dimainkan, atau menonaktifkan sepenuhnya, yang membuat *equalizer* menjadi barang yang tidak digunakan dalam setiap pemutar media. Idenya adalah sederhana; di satu sisi kita memiliki label *genre*. Di sisi lain preset *equalizer*. Apakah mungkin untuk memetakan preset untuk genre lagu yang sedang diputar? Jadi, jika sebuah lagu dengan *genre Rock* diputar, *equalizer* yang telah ditetapkan *Rock* secara otomatis diterapkan. Tentu, kesulitan dari fitur ini terletak pada pemetaan itu sendiri. Sebuah percobaan pertama bisa saja menduga kesamaan, contohnya: memiliki *genre* yang disebut '*HardRock*' tapi tidak ada preset dengan nama itu, kita bisa memuat standar preset *'Rock'*. Dan mungkin membuat atau memuat preset generik jika tidak ada genre yang tersedia dalam *tag*. Saya kira yang paling akurat adalah membuat *panel* di mana pengguna dapat menentukan pemetaan sendiri; menampilkan daftar preset yang saat ini tersedia di sebelah kiri, daftar *genre* yang dikenal di sebelah kanan dan pilih pemetaan anda sendiri seperti itu di mana preset tertentu dapat dipetakan ke beberapa *genre* atau *subgenre*.

Tabel A. 5 Hasil Pengujian Klasifikasi Pesan Untuk Aplikasi Kedua

Pesan	*Library* atau *component* yang terpilih	Pengguna Penerima Pesan
1	*Javax.Swing*	Aekold, laurent, tobias
2	*Audio CD*	Thomas, roeland
3	*Playlist*	Sylvain, stefan
4	*Playlist*	Sylvain, stefan
5	*Internet*	Alex, philip
6	*Internet*	Alex, philip
7	*Playlist*	Sylvain, stefan
8	*Playlist*	Sylvain, stefan
9	*Internet*	Alex, philip
10	*Internet*	Alex, philip

Tabel A. 6 Daftar Pengembang pada Aplikasi Ketiga

No	Nama	Pembagian *Library* dan *Component*
1	julian	*General, song player, edit song*
2	Hernan	*Sound, effects*
3	Nahuel	*File format, edit song*
4	Nikola	*General, interface*
5	Aaron	*File format, interface*
6	Herak	*Sound, song player*
7	auria	*Sound, effects*

Tabel A. 7 Daftar dan Deskripsi *Library* pada Aplikasi Ketiga

No	Nama	Deskripsi *Library*
1	*Gene-ral*	Merupakan *plugin* atau paket yang berisi format *file converter*. Memungkinkan untuk mengkonversi *file* dari berbagai format *file* yang didukung oleh aplikasi ke dalam format tujuan, dengan menjaga nama tetapi mengubah format dan ekstensi.
2	*Sound*	Merupakan *plugin* suara yang berisi JSA, ALSA, OSS dan *core audio*. JSA menghubungkan TuxGuitar dengan API suara pada Java sistem MIDI dan dapat memuat MIDI *sequencer* dan MIDI *Port*. Alsa menghubungkan *output* suara MIDI ke ALSA, *Advanced Linux Sound Architecture*. OSS menghubungkan *output* suara MIDI ke OSS, *Open*

No	Nama	Deskripsi *Library*
		Sound System, salah satu sistem *output* suara yang paling populer untuk Linux, FreeBSD dan sistem operasi nix lainnya. CoreAudio menghubungkan *output* suara MIDI ke CoreAudio.
3	*File Format*	Merupakan *plugin* format *file* yang berisi compat, GTP, PTB, TEF, Lilypond dan MusicXML. Compat memungkinkan untuk membuka *file* yang disimpan oleh versi lama dari TuxGuitar. GTP memungkinkan untuk membuka dan menyimpan *file* Guitar Pro, untuk format yang dapat dibaca yaitu gtp, gp3, gp4, gp5, sedangkan untuk format yang dapat ditulis adalah gp3, gp4, gp5. PTB memungkinkan untuk membuka *file* ptb dari Power Tab. TEF memungkinkan untuk mengimpor file TEF dari TablEdit. Lilypond memungkinkan untuk mengekspor hasil dari TuxGuitar sebagai file Lilypond. MusicXML untuk mengekspor hasil TuxGuitar sebagai file MusicXML.

Tabel A. 8 Daftar dan Deskripsi *Component* pada Aplikasi Ketiga

No	Nama	Deskripsi *Component*
1	*Interface*	Antarmuka dari tuxguitar menggunakan komponen dari java swt. Pada bagian *toolbar* terdapat berbagai ikon dengan fungsi yang berbeda, seperti manajemen *file*, *undo/redo*, mode *edit*, properti lagu, tambah/hapus lagu, durasi, notasi musik, komposisi, kontrol pemutar, tampilan *view* dan efek.
2	*Song Player*	TuxGuitar mendukung format *file* selain format internal Tg, yaitu semua format Guitar Pro, GP3, GP4, GP5, dan juga format PowerTab ptb. Selain membuka *file* lokal bisa juga membuka *file* dari sumber *online* dengan menggunakan *dialog Open* URL. Menggunakan kontrol pemutaran atau *playback*, dapat memutar, berhenti dan jeda pemutaran, tetapi juga dapat bernavigasi melalui lagu, awal atau akhir lagu. Selama pemutaran, notasi yang sedang diputar akan berwarna merah, sehingga lebih mudah untuk melihat bagaimana notasi saat dimainkan. Instrumen saat ini dapat diubah dengan menekan pada nama instrumen

		pada jendela. Terdapat dua mode pemutar, yaitu *simple mode* yang memungkinkan tempo diubah selama pemutaran dan *training mode* untuk berlatih dengan tempo yang naik secara bertahap.
3	*Edit Song*	Setiap lagu terdiri dari satu atau lebih *tracks* dan setiap *track* ditetapkan untuk satu instrumen. Dalam setiap *track* dapat diberi notasi baru dan harus menyesuaikan beberapa pengaturan, seperti waktu ketukan, tempo, kunci dan *triplet*. Dapat juga diberi *chord* yang sudah didefinisikan sebelumnya atau *chord* baru. Selain itu juga dapat diberi lirik dalam *track*. Untuk edit lagu secara penuh sama dengan edit setiap *track* tetapi pada aplikasi ini dapat edit seluruh lagu dengan menambahkan atau menghapus *track* dan menambahkan instrumen.
4	*Effec-ts*	Dalam edit sebuah lagu dapat diberi efek instrumen seperti efek pada gitar atau *bass*. *Dead note* adalah efek bermain yang menghasilkan efek perkusi. *Ghost* dan *accentuated note* adalah efek yang mengubah dinamika notasi. *Harmonics, slide* dan *tremolo* adalah efek dari gitar. *Vibrato* dan *bend* adalah efek yang mempengaruhi nada.

Tabel A. 9 Daftar Pesan Untuk Aplikasi Ketiga

No	Pesan
1	Dalam *file* ini, di beat 6, tempo melambat secara bertahap dari 60-20, tetapi di beat 7 tempo kembali ke tempo asli dari 20 sampai 60. Guitar pro bermain dengan cara ini. TuxGuitar hanya melihat perubahan terakhir sehingga dalam TuxGuitar perubahan tempo secara drastis dan tidak bertahap dan untuk seluruh lagu. Dalam beat 7 dengan TuxGuitar, tempo tidak beralih kembali ke tempo aslinya, tetap 20 bpm, seharusnya beralih kembali ke 60.
2	Saya telah mengkonversikan file GPX ke GP5 menggunakan WebTabPlayer (http://www.webtabplayer.com). *File* yang dihasilkan adalah: http://www.webtabplayer.com/Download/146343/gp5. Tampaknya bahwa *file* tersebut memiliki UTF-8 BOM (http://stackoverflow.com/questions/7297888/ufeff-character-

	showing-up-in-files-how-to-remove-them). Kebetulan, TuxGuitar tidak berhasil membuka *file*. Saya menggunakan utilitas bomstrip (http://www.ueber.net/who/mjl/projects/bomstrip/) untuk menghapus BOM, yang memungkinkan TuxGuitar untuk membuka *file*. Aku tidak pernah memiliki masalah sebelumnya dan saya tidak tahu apakah itu adalah *bug* dari WebTabPlayer. Mungkin akan lebih baik untuk memiliki TuxGuitar menangani berkas GP5 dengan BOM?
3	*Ada sebuah file GP5 yang menyebabkan TuxGuitar 1.2 crash ketika membukanya. Ini adalah pesan kesalahan saya:* *Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: -1* *at java.util.ArrayList.elementData(ArrayList.java:338)* *at java.util.ArrayList.get(ArrayList.java:351)* *at org.herac.tuxguitar.song.models.TGTrack.getString(Unknown Source)* *at org.herac.tuxguitar.gui.editors.tab.edit.MouseKit.mouseMove(Unknown Source)* *at org.herac.tuxguitar.gui.editors.tab.edit.EditorKit.mouseMove(Unknown Source)* *at org.eclipse.swt.widgets.TypedListener.handleEvent(TypedListener.java:212)* *at org.eclipse.swt.widgets.EventTable.sendEvent(EventTable.java:84)* *at org.eclipse.swt.widgets.Widget.sendEvent(Widget.java:1276)* *at org.eclipse.swt.widgets.Display.runDeferredEvents(Display.java:3554)* *at org.eclipse.swt.widgets.Display.readAndDispatch(Display.java:3179)*

	at org.herac.tuxguitar.gui.TuxGuitar.displayGUI(Unknown Source) *at org.herac.tuxguitar.gui.TGMain.main(Unknown Source)*
4	Buka lagu dan memainkannya untuk sementara waktu. Kemudian berhenti bermain. Pergi ke *Tools* -> Pengaturan dan konfigurasi ke Midiport lainnya. Cobalah untuk memainkan lagu yang tadi dimainkan lagi. Hasilnya adalah bahwa tidak memiliki suara. Anda harus menutup aplikasi untuk mendapatkan suara lagi.
5	Aku mencoba *track* sederhana dan diekspor ke lilypond. Itu tampak baik kecuali judul dan informasi komposer tidak tercetak ketika dikompilasi dengan lilypond 2.12.2. Menurut saya masalahnya adalah bahwa *header* bait tertutup dalam skor dan menggerakkan keluar akan memperbaiki masalah.
6	Ketika TuxGuitar ekspor ke LilyPond, saya mendapatkan *error: 'Wrong type argument in position 1 (expecting Pitch):'* Saya kira ini adalah kode yang *error:* *TrackATabStaff = new TabStaff with { stringTunings = #'(9 4 0 7) } <<* Bagaimana untuk memperbaiki ini?
7	Saya sedang menulis sebuah lagu. Pengukuran pertama, bermain baik-baik saja, tapi setelah beberapa pengukuran, ia berhenti bermain pada salah satu *track*. Saya memeriksa untuk melihat apa yang salah dan memeriksa campuran, saya juga memeriksa jika catatan ditempatkan pada tempat yang tepat di jalur yang benar, tapi mereka semua baik-baik saja. Saya tidak mengerti. Pada awalnya ini memainkan semua track, dan setelah beberapa saat berhenti bermain pada salah satu *track*.
8	Saya punya *plugin* alsa terinstal yang bekerja tanpa *port* alsa, Saya bisa memainkan midifiles di Rhythmbox atau *video-player* namun ketika memainkan melalui konsol dengan aplaymidi saya tidak mendengar apa-apa. Tidak ada kesalahan lain, TuxGuitar sendiri tampaknya bekerja dengan baik.
9	Ketika saya mengekspor file GP5 ke midi, kemudian diimpor ke Reaper itu tidak membawa nama trek, seperti gitar, perkusi, dll. Ini memanggil semua *track* nama dari file. Dengan Gpro5 membawa nama *track* untuk masing-masing lagu.
10	Saya melihat masalah dengan koding di ujung alternatif dalam *file* LilyPond yang diekspor. TuxGuitar memberikan akhiran

	alternatif dari bagian yang diulang tetapi informasi itu tidak benar pada *output* dalam kode LilyPond. Bagian diulang datang dengan benar, tetapi tidak pada akhiran alternatif. Saat ini, kode LilyPond termasuk alternatif pertama berakhir dalam *repeat* {} kurung dan hanya kode akhir kedua sebagai ukuran baru setelah ulangi. Saya mendapatkan *output* LilyPond seperti ini ais83 b82 c'82 cis'82 d'82 c'82 ais83 b82 g43 4 g,46 f'81 fis'81 } g43 4 g,46 d44 8 g83~ g83 g83 4 g83 a83 ketika saya harus melihat ujung alternatif seperti ini: ais83) b82 c'82 cis'82 d'82 c'82 ais83 b82 } alternative {{g43 4 g,46 f'81 fis'81 } {g43 4 g,46 d44} } 8 g83~ g83 g83 4 g83 a83 LilyPond memahami bahwa bar ulangi menutup ukuran alternatif pertama dan bukan ukuran terakhir di blok ulangi.

Tabel A. 10 Hasil Pengujian Klasifikasi Pesan Untuk Aplikasi Ketiga

Pesan	*Library* atau *component* yang terpilih	Pengguna Penerima Pesan
1	*Song player*	Julian, herak
2	*General*	Julian, nicola
3	*Song player*	Julian, herak
4	*Song player*	Julian, herak
5	*File format*	Nahuel, aaron
6	*File format*	Nahuel, aaron
7	*Song player*	Julian, herak
8	*Sound*	Hernan, herak, auria
9	*Song player*	Julian, herak
10	*Song player*	Julian, herak