**JURUSAN TEKNIK INFORMATIKA**

**FAKULTAS TEKNOLOGI INFORMASI**

**INSTITUT TEKNOLOGI SEPULUH NOPEMBER**

**USULAN TUGAS AKHIR**

# **IDENTITAS PENGUSUL**

**Nama : Galih Adi Paryanto**

**NRP : 5108100005**

**Dosen Wali : Anny Yuniarti, S.Kom, M.Comp.Sc**

# **JUDUL TUGAS AKHIR**

“Perancangan dan Pembangunan Sistem Pengelompokan Surat Elektronik secara Otomatis dengan Metode Klasifikasi K-Nearest Neighbour Berbasis Kerangka Kerja Spring pada Platform Android”

# **LATAR BELAKANG**

Dewasa ini, email sudah menjadi alat komunikasi umum yang digunakan oleh banyak orang. Setiap pengguna Internet paling tidak memiliki satu alamat email. Menurut riset yang diselenggarakan oleh The Radicati Group Inc pada tahun 2010 menunjukkan bahwa jumlah pengguna email di seluruh dunia mencapai 2.9 milyar dan diperkirakan akan meningkat lagi menjadi 3.8 milyar pada tahun 2014 [2]. Banyaknya jumlah pengguna email sekarang disebabkan manfaat yang begitu besar. Email dianggap sebagai sarana berkomunikasi dan bertukar informasi yang mudah, murah dan cepat. Dengan email, pengguna dapat saling berkomunikasi dan berbagi informasi kepada keluarga, teman, dan rekan bisnis. Selain itu, mereka juga bisa mendapatkan informasi, berita, pemberitahuan, promosi ataupun iklan terbaru dengan cepat dan mudah. Selain jumlah pengguna yang besar, intensitas lalu lintas pengiriman dan penerimaan email juga tinggi. Menurut riset yang sama, pengguna email pada lingkup perusahaan mengirim dan menerima sekitar 105 email setiap hari. Sebanyak 19 % dari email tersebut berupa spam maupun graymail (unwanted newsletter, pesan peringatan, dll) [2]. Intensitas email juga diperkirakan akan terus meningkat setiap tahunnya.

Dari data statistik sebelumnya menunjukkan bahwa jumlah pengguna dan intensitas penggunaan email semakin meningkat. Hal ini akan membawa dampak pada pengelolaan email pengguna, khususnya pada pengelolaan pesan masuk (*inbox*). *Inbox* dipenuhi dengan berbagai macam email yang berasal dari banyak pengirim dengan konten yang berbeda. Akibatnya, kumpulan email tidak tersusun dan terorganisasi dengan baik. Pengguna akan sulit untuk menentukan mana email yang penting dibaca dengan email yang tidak penting untuk dibaca.

Banyak email server yang sudah menyediakan fitur untuk menyaring email, misal *gmail*, *ymail* maupun email server yang lain. Namun penyaringan email yang dilakukan hanya sebatas pada pengirim email, tanggal pengiriman dan subject email. Penyaringan tersebut dirasa belum cukup membantu menyelesaikan masalah. Tidak semua pengguna tahu atau masih ingat nama pengirim, tanggal pengiriman maupun subject dari email yang diterima. Disamping itu, penyaringan yang dilakukan harus dilakukan secara manual oleh pengguna.

Untuk itu, diperlukan suatu sistem yang dapat mempermudah pengguna untuk membaca setiap email yang masuk . Dalam tugas akhir ini, penulis akan membangun suatu sistem yang mampu mengelompokkan email secara otomatis berdasarkan tingkat kemiripan antar email. Email yang masuk ke *inbox* akan disaring dan dikelompokkan ke dalam kategori sesuai tingkat kemiripan masing-masing. Dengan begitu, kumpulan email akan tersusun dan tertata dengan baik.

Sistem ini akan dikembangkan seperti *email client* dan berbasis *web application*. Sistem menggunakan metode K-Nearest Neighbour (KNN) sebagai metode klasifikasi email. Metode KNN dan Support Vector Machine (SVM) memiliki performa yang lebih baik daripada metode klasifikasi yang lain [3]. Akan tetapi, metode KNN sendiri cukup sederhana dan efektif dalam melakukan kategorisasi teks. Selain itu, proses klasifikasi KNN juga mudah untuk direpresentasikan dibandingkan dengan metode lainnya [4]. Sistem dibangun dengan menggunakan kerangka kerja Spring pada perangkat android yakni Spring Android. Spring Android merupakan ekstensi dari kerangka kerja Spring yang bertujuan untuk menyederhanakan pengembangan aplikasi native android.

# **RUMUSAN MASALAH**

Rumusan masalah yang diangkat dalam Tugas Akhir ini adalah:

1. Bagaimana membangun sistem *email client* pada perangkat android.
2. Bagaimana mengimplementasikan pemrosesan awal (*pre-processing*) email.
3. Bagaimana menentukan kelompok (*cluster*) terhadap email berdasarkan metode klasifikasi K-Nearest Neighbour.

# **BATASAN MASALAH**

Permasalahan yang dibahas dalam Tugas Akhir ini memiliki beberapa batasan, yakni sebagai berikut:

1. Sistem dibangun dengan menggunakan kerangka kerja Spring Android.
2. Sistem menggunakan protokol IMAP.
3. Sistem menangani email berbahasa Indonesia.

# **TUJUAN TUGAS AKHIR**

1. Merancang bangun aplikasi *email client* pada perangkat android yang mampu mengelompokkan email berdasarkan kemiripan secara otomatis.
2. Mengimplementasikan metode klasifikasi K-Nearest Neighbour sebagai metode pengelompokkan email.
3. Membangun aplikasi dengan menggunakan teknologi kerangka kerja Spring Android.

# **MANFAAT TUGAS AKHIR**

Manfaat dari aplikasi yang dibangun dalam tugas akhir ini adalah sebagai alat untuk mempermudah pengelolaan email secara otomatis pada perangkat android karena setiap email yang masuk sudah dikelompokkan berdasarkan tingkat kemiripan masing-masing.

1. **TINJAUAN PUSTAKA**
2. **Rancang Bangun Perangkat Lunak**

Rancang bangun perangkat lunak merupakan kumpulan beberapa tahapan teknis untuk membangun perangkat lunak yang meliputi tahap perencanaan, analisis, design, implementasi, pengujian serta pemeliharaan perangkat lunak.

Rancang bangun perangkat lunak digunakan untuk menentukan konsep dan strategi yang baik dalam membangun perangkat lunak yang handal, berkualitas tinggi, sesuai dengan anggaran biaya dan mudah pemeliharaannya serta tidak membutuhkan waktu lama dalam pembangunannya [1].

1. **Email Client**

*Email client* merupakan suatu aplikasi yang digunakan untuk mengelola email pengguna. Istilah ini dapat mengacu pada sistem yang mampu mengakses email dalam *inbox*, mendapatkan pesan terbaru, mengkomposisi dan membuat pesan baru. *Email client* juga bisa disebut sebagai agen yang bekerja sebagai *client* terhadap sebuah email server.

Untuk mendapatkan akses pesan dari *inbox*, *email client* harus terhubung dengan mail server melalui suatu protokol. Protokol yang digunakan bisa berupa Post Office Protocol (POP). POP memungkinkan pengguna untuk mengunduh pesan satu per satu dan menghapus pesan tersebut dari server setelah pesan berhasil disimpan di penyimpanan lokal. Hal itu dilakukan untuk menghindari agar pengguna lain tidak dapat mengakses pesan tersebut. Protokol lain yang saat ini banyak digunakan yakni Internet Message Access Protocol (IMAP). IMAP memungkinkan pengguna untuk mendapatkan pesan dari *inbox* tanpa harus menghapus pesan tersebut dari email server. Sehingga, jika pengguna login ke email server dengan *device* yang berbeda, maka pesan yang telah dibuka akan tetap tersimpan di email server.

1. **Javamail**

Javamail merupakan API (Application Programming Interface) untuk mengelola email yang digunakan dalam bahasa pemrograman Java. Hal ini memungkinkan bagi penyedia *service* untuk menyediakan *interface* standar ke sistem pemilik email. Dengan Javamail, *developer* mampu membangun aplikasi *email client* yang dapat mengakses lokasi pesan, serta membuat dan mengirim pesan ke pengguna lain dengan mudah.

Pada dasarnya, Javamail terdiri dari kumpulan *abstract class* dengan model yang bermacam-macam. Adapun kelas-kelas tersebut antara lain :

* Message

Message merupakan *abstract class* yang menggambarkan komponen dari email. Javamail mengimplementasikan standar RFC822 serta MIME (Multipurpose Internet Mail Extension).

* Store

Store merupakan *abstract class* yang menggambarkan database dari email yang dikelola oleh email server dan dikelompokkan oleh pengguna. Class store menggunakan akses protokol khusus untuk mengakses database email.

* Folder

Folder merupakan *abstract class* yang menyediakan hubungan hirarki organisasi pesan.

* Transport

Transport berupa *abstract class* yang menggambarkan transport protokol khusus. Dalam hal ini, object Transport akan menggunakan transport protokol khusus untuk mengirim email ke tujuan.

1. **Representational State Transfer (REST) Web Service**

Representational State Transfer (REST) merupakan prinsip arsitektur perangkat lunak untuk merancang suatu layanan web (*web service*) yang berfokus pada pendefinisian dan pengalamatan sumber daya sistem, termasuk bagaimana sebuah state sumber daya diatur dan ditransfer melalui HTTP oleh berbagai macam klien yang ditulis dalam berbagai macam bahasa. Jika diukur dalam jumlah layanan web yang menggunakannya, REST muncul sebagai layanan web yang paling banyak digunakan dalam beberapa tahun terakhir. Keberadaaan REST memilik dampak besar terhadap perkembangan web dan secara berangsur-angsur menggantikan desain layanan web SOAP dan WSDL.

REST *web service* memiliki beberapa karakteristik sebagai berikut :

* **Menggunakan method HTTP secara eksplisit**

Salah satu karakteristik kunci dari RESTful *web service* adalah penggunaan method HTTP secara eksplisit. REST *web service* memetakan *one-to-one* antara operasi create, read, update dan delete (CRUD) dengan 4 metode utama dalam HTTP sebagai berikut :

1. GET : digunakan untuk mendapatkan sumber daya.
2. POST : digunakan untuk membuat sumber daya di server.
3. PUT : digunakan untuk mengubah state sumber daya.
4. DELETE : digunakan untuk menghapus sumber daya.

* **Stateless**

Stateless web service membuat sebuah respon yang terhubung dengan halaman *resource*. Hal ini berbeda dengan stateful service yang menyimpan variabel dari *request* sebelumnya dan menggunakannya kembali dengan tambahan variabel baru untuk mengakses *resource* yang diminta oleh sebuah *request* baru. Ketika klien membuat sebuah HTTP *request*, semua informasi yang dibutuhkan server untuk memenuhi *request* harus dikirim. Server tidak bergantung pada informasi yang dikirim dari *request* sebelumnya.

* **Memiliki struktur direktori URI**

Untuk mengakses RESTful *web service* digunakan sebuah **URI**(Uniform Resource Identifiers). URI adalah nama dan alamat dari sebuah *resource*.

* **Pesan ditransfer dalam format XML, JavaScript Object Notation (JSON), atau keduanya**

RESTful *web service* tidak menggunakan WSDL, pesan yang dikirim dikemas dalam format XML dan JSON. Hal ini berbeda dengan SOAP *web service* yang menggunakan protokol khusus untuk pengiriman pesan.

Prinsip dan karakteristik inilah yang mendorong sistem yang menggunakan REST menjadi lebih sederhana, ringan dan memiliki performa yang tinggi.

1. **Spring Android**

Spring merupakan salah satu kerangka kerja (*framework*) Java yang handal dan digunakan secara luas. Spring dianggap sebagai standard stack development untuk pembuatan aplikasi berbasis Java EE. Komponen utama dari Spring ini adalah *container* Invertion of Control (IoC) yang berdasarkan dari penggunaan JavaBean [5]. IoC merupakan suatu *pattern* yang mana programmer tidak perlu menciptakan object dengan menginisialisasi *class*-nya, tapi dengan mendeskripsikannya di dalam sebuah *interface* dalam bentuk xml. Deskripsi object tersebut bisa berupa inisiasi object, pemberian nilai properti maupun penghapusan object. Hal ini memungkinkan Spring dapat melakukan Dependency Injection pada suatu object. Dependency Injection merupakan implementasi dari IoC yang merupakan *pattern* dimana *container* mampu meng-*inject* *dependency* dari suatu object ke object lainnya.

Spring menyediakan banyak modul yang dapat memudahkan *developer* dalam membangun perangkat lunak. Salah satu modul yang berbagai modul untuk mengakses data. Spring dapat dengan mudah diintegrasikan dengan berbagai macam ORM dan *library* seperti Hibernate, Ibatis, JDO, JPA, OXM, maupun menggunakan JDBC biasa. Selain dengan ORM, Spring juga mudah diintegrasikan dengan *framework* Java lain seperti JSF, ZK, Strut, WebWork dan Tapestry.

Spring Android merupakan ekstensi dari kerangka kerja Spring yang bertujuan untuk menyederhanakan pengembangan aplikasi native android. Tidak semua modul dari Spring dapat diimplementasikan pada Spring Android. Spring Android hanya menerapkan komponen yang digunakan dalam sisi klien pada Spring ke dalam pengembangan aplikasi Android. Fitur utama dari Spring Android yakni dukungan terhadap RESTful client application dan otentifikasi untuk mengakses *secure* API. Spring Android menyediakan modul RestTemplate, yang berasal dari RestTemplate Spring yang terkenal sebagai REST klien berbasis Java paling handal dan paling banyak digunakan. RestTemplate bekerja dengan menyediakan metode callback dan mengatur HttpMessageConverter yang digunakan untuk menyusun objek-objek ke dalam *body* dari HTTP *request* dan memecah respon ke dalam bentuk objek kembali. Spring Android juga dapat mengatasi kebutuhan aplikasi pada android untuk mendapatkan otorarisasi layanan web sehingga sistem dapat mengakses data melalui layanan web dengan mudah.

1. **K-Nearest Neighbour**

K-Nearest Neighbour (KNN) merupakan metode untuk melakukan klasifikasi terhadap suatu object berdasarkan data latih yang jaraknya paling dekat atau tingkat kemiripannya paling tinggi. Secara sederhana, metode ini bekerja dengan cara membandingkan jarak data masukan dengan k data latih yang paling dekat [9].

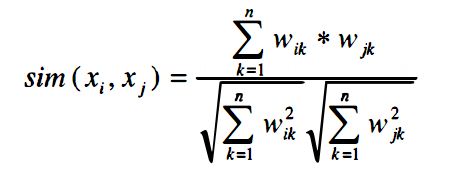
Metode K-Nearest Neigbour ini dibagi menjadi 2 fase :

1. Fase Pembelajaran

Pada fase ini, algoritma hanya akan melakukan proses penyimpanan vector-vektor fitur dan label dari data latih. Dalam klasifikasi teks, fitur ini bisa berupa term-term dengan bobot dari masing-masing term tersebut.

1. Fase Klasifikasi

Pada fase klasifikasi, semua data latih akan dibandingkan nilai kemiripannya dengan data uji. Perhitungan nilai kemiripan (*similarity*) menggunakan persamaan cosine similarity dengan rumus :



Dimana sim (xi, xj) merupakan *similarity* dari dokumen ke-i dan ke-j. Sedangkan w merupakan bobot dari masing-masing term pada dokumen.

Setelah didapatkan semua nilai kemiripan, langkah selanjutnya yakni mencari sebanyak k vector pada data latih yang memiliki nilai kemiripan paling tinggi. Label dari k data latih yang memiliki nilai kemiripan tertinggi akan menjadi label dari data uji. Sehingga, data uji akan diklasifikasikan ke dalam kategori sesuai dengan label tersebut.

# **RINGKASAN TUGAS AKHIR**

Pada tugas akhir ini, akan dibangun suatu sistem pengelompokkan email secara otomatis berdasarkan tingkat kemiripan email. Sistem dikembangkan seperti email client dan berjalan pada perangkat android. Sistem ini akan membantu pengguna email untuk mengelola email yang masuk karena email disaring dan dikelompokkan berdasarkan kemiripan masing-masing.

Secara garis besar, desain arsitektur dari sistem yang akan dibangun ditunjukkan pada gambar 2. Sistem terhubung ke email server dan menangani email berbahasa Indonesia. Dari desain arsitektur tersebut, sistem dapat dijelaskan ke dalam 3 bagian, yakni sebagai berikut :

1. Input

Sistem menerima input berupa email yang diperoleh dari mail server. Untuk mendapatkan email tersebut, sistem menggunakan Email Retriever Engine yang dibangun dengan memanfaatkan *library* Javamail. Selanjutnya, email akan diubah ke dalam bentuk file teks dan disimpan ke penyimpanan lokal sistem.

1. Proses

Teks email yang telah disimpan akan diolah dan diproses oleh sistem melalui Clasification Engine. Tidak semua bagian email digunakan dalam Clasification Engine. Sistem hanya akan mengambil subject dan body email. Clasification Engine terdiri dari beberapa proses yang saling berurutan untuk mengklasifikasikan email ke dalam suatu kelompok (cluster). Urutan proses-proses dalam Clasification Engine dapat dilihat pada gambar 3. Pada dasarnya, kumpulan proses tersebut dapat dikelompokkan menjadi 2 proses utama.. Adapun proses utama tersebut antara lain :

* Text processing

Text processing ini terdiri dari beberapa proses pengelolaan teks yang akan menghasilkan informasi yang dibutuhkan oleh proses selanjutnya. Proses – proses tersebut terdiri dari :

1. Token removal

Proses ini bertujuan untuk menghapus semua token yang ada pada text. Token bisa berupa tanda baca, tanda penghubung, spasi, angka dan semacamnya.

1. Case folding

Proses ini digunakan untuk mengubah semua huruf pada teks menjadi huruf kecil.

1. Remove stop words.

Remove stop words merupakan proses penghapusan kata-kata yang tidak berpengaruh terhadap makna dari sebuah teks, sehingga yang tersisa hanyalah bagian-bagian yang mewakili makna dari teks tersebut. Kata-kata tersebut biasanya berupa kata penghubung, kata ganti, dan kata keterangan lainnya yang dianggap tidak penting.

1. Stemming.

Stemming merupakan proses penghapusan imbuhan dari suatu kata sehingga didapatkan bentuk kata dasarnya atau yang disebut dengan term. Imbuhan kata dalam bahasa Indonesia bisa berupa awalan (prefix), akhiran (suffix) dan sisipan (infix).

1. Indexing

Proses terakhir dari text processing engine yakni proses indexing. Indexing merupakan proses perhitungan bobot dari setiap term.. Pada proses indexing, setiap term akan diberi bobot sesuai dengan frekuensi kemunculan di dalam teks. Perhitungan bobot pada tiap term menggunakan TF/IDF dengan rumus sebagai berikut :

Bobot (i) = frekuensi term \* (1 + log(N) - log(di))

Dimana :

Bobot(i) = bobot term i.

N = jumlah file keseluruhan

di = jumlah file yang mengandung term i

Term dengan bobot tertinggi menunjukkan bahwa term tersebut sangat berpengaruh terhadap makna dari sebuah teks. Data-data berupa term beserta bobotnya akan digunakan pada proses yang lebih lanjut.

* Pengelompokkan dengan metode K-Nearest Neighbour

Pada proses pengelompokkan, term-term beserta bobot dari hasil text processing akan dijadikan sebagai data uji dan diklasifikasikan berdasarkan data latih yang sudah disediakan oleh sistem. Data latih tersimpan dalam database lokal sistem dan tiap data latih memiliki label yang akan menjadi label (kategori) dari email yang masuk. Langkah – langkah pengelompokkan email berdasarkan metode K-Nearest Neighbour adalah sebagai berikut :

* Menentukan data latih beserta label masing-masing.
* Menghitung nilai kemiripan (similarity) antara data uji dengan data latih.
* Mengambil sebanyak k data latih yang memiliki nilai kemiripan paling besar
* Label pada k data latih yang paling banyak (mayoritas) akan menjadi label dari email.

Email dikelompokkan ke dalam kategori berdasarkan label yang diperoleh dari data latih. Terdapat 4 kategori yang digunakan dalam sistem yakni personal, work, hobby dan notification.

1. Output

Output yang dihasilkan oleh sistem berupa pengelompokkan email ke dalam folder kategori. Penentuan kategori email dilakukan berdasarkan hasil dari proses sebelumnya.

Fitur utama dari aplikasi ini yakni mengelompokkan email ke dalam kelompok kategori berdasarkan kemiripan terhadap kelompok tersebut. Email akan dikelompokkan ke dalam kategori yang memiliki nilai kemiripan yang lebih tinggi. Selain fitur utama tersebut, aplikasi yang dibangun pada tugas akhir ini juga memiliki fitur-fitur lain. Fitur - fitur tersebut antara lain :

1. Fitur Email History

Fitur ini digunakan untuk mencatat semua email baru yang masuk ke dalam *inbox*. Email tersebut akan ditampilkan dalam suatu halaman *history*, dimana pengguna dapat melakukan Training Email.

1. Fitur Tambah Kategori

Dengan menggunakan fitur ini, pengguna dapat menambah kategori email sesuai dengan keinginan sendiri. Karena sifat email yang dinamis dan tidak mudah ditebak, maka diperlukan fitur tambah kategori untuk membuat kategori baru yang sesuai dengan karakteristik email pengguna. Fitur ini menjadi solusi apabila sistem kurang akurat dalam mengelompokkan email.

1. Fitur Training Email

Selain dengan fitur tambah kategori, sistem juga menyediakan fitur training email untuk mengantisipasi apabila sistem kurang tepat dan akurat dalam mengelompokkan email. Training email mengijinkan pengguna untuk mengubah kategori email yang telah diklasifikasikan oleh sistem. Setiap perubahan pada kategori email, sistem akan mengupdate data uji sehingga klasifikasi email menjadi lebih tepat dan akurat.

1. Fitur Tambah Stop Words

Fitur ini digunakan untuk menambah perbendaharaan kata yang dianggap sebagai stop words dan tidak mempengaruhi makna dari email. Setiap penambahan stop words, sistem akan mengupdate list stop word yang tersimpan di penyimpanan lokal sistem.

Pada awal penggunaan sistem, setiap email yang masuk akan ditampung ke dalam folder kategori *unclassified.* Hal ini disebabkan sistem belum memiliki data latih, dimana data latih hanya bisa ditambahkan oleh pengguna. Penggunaan email masuk sebagai data latih bertujuan agar sistem mampu menyesuaikan karakteristik email pengguna. Penambahan data latih dilakukan dengan cara mengubah kategori *unclassified* dengan kategori yang telah disediakan oleh sistem. Sistem akan memindahkan email ke dalam folder kategori yang dipilih dan menambahkannya ke dalam database data latih. Apabila pengguna sering memasukkan data latih baru, maka klasifikasi sistem akan menjadi lebih akurat dan tepat.

Karena sistem menggunakan protokol IMAP, maka setiap perubahan yang terjadi pada email di dalam sistem juga akan terjadi pada email server. Ketika sistem membuat folder kategori baru, maka di email server juga akan terdapat folder yang sama. Begitu juga ketika sistem mengelompokkan email ke dalam suatu folder, maka email server akan mengelompokkan email ke dalam folder yang sama.

Sistem dibangun dengan menggunakan pola Model-View-Controller (MVC) pada teknologi berbasis layanan web (*web service*). Model dan controller berada pada sisi server sistem sedangkan view ditampilkan pada sisi klien berupa aplikasi yang berjalan pada perangkat android. Desain arsitektur client-server sistem dapat dilihat pada gambar 1. Berdasarkan gambar tersebut, segala proses bisnis sistem yang telah dijelaskan sebelumnya terletak pada sisi server. Server berhubungan langsung dengan email server untuk mendapatkan email. Server menyediakan layanan (*service*) pada controller kepada klien berupa data-data email yang sudah dikelompokkan oleh sistem. Sedangkan klien hanya bertugas untuk menangkap data yang dikirim oleh server melalui layanan web tersebut. Kemudian, data yang diperoleh akan ditampilkan ke dalam aplikasi android yang berinteraksi langsung dengan pengguna. Fitur lain yang terdapat dalam sistem juga akan ditampilkan dalam aplikasi android.



**Gambar 1 Desain Arsitektur Client-Server Sistem**



**Gambar 2 Desain Arsitektur Sistem**



**Gambar 3 Alur Proses Sistem**



**Gambar 3 Use Case Diagram Sistem**

# **METODOLOGI**

1. Penyusunan proposal tugas akhir

Tahap awal dalam pengerjaan tugas akhir ini adalah penyusunan proposal tugas akhir. Pada proposal ini, penulis mengajukan gagasan mengenai pembangunan suatu sistem yang mampu mengelompokkan email berdasarkan kemiripan dari email tersebut terhadap kelompok email.

1. Studi literatur

Pada tahap ini dilakukan pencarian, pengumpulan, penyaringan, pembelajaran dan pemahaman literatur yang diperlukan dalam pembangunan sistem yang penulis ajukan. Literatur yang dibutuhkan berhubungan dengan :

1. Metode K-Nearest Neighbour sebagai metode klasifikasi email.
2. Kerangka kerja Spring Android untuk mempermudah pembangunan aplikasi pada perangkat android.
3. Proses *text processing* berbahasa Indonesia.

Literatur yang digunakan dalam pengerjaan tugas akhir ini sebagian besar berasal dari internet berupa makalah ilmiah, artikel, paper, materi kuliah dan juga buku referensi yang terkait.

1. Analisis dan desain perangkat lunak

Analisis kebutuhan sistem dilakukan untuk merumuskan solusi yang tepat dalam pembuatan aplikasi serta kemungkinan yang dapat dilakukan dalam pengimplementasian rancangan tersebut. Di samping itu, permasalahan teknologi yang digunakan juga dibahas pada fase ini. Fase desain meliputi arsitektur perangkat lunak yang akan digunakan, desain *class* aplikasi, desain basis data, dan lain-lain.

1. Implementasi perangkat lunak

Implementasi merupakan tahap dalam membangun perangkat lunak, dimana hasil dari fase desain perangkat lunak diwujudkan ke dalam bentuk kode program. Pembangunan perangkat lunak menggunakan IDE Netbeans, sedangkan bahasa pemrograman yang dipakai yakni Java. Guna membantu kemudahan dalam membangun sistem, penulis menggunakan framework Spring yang digunakan pada perangkat android yakni Spring Android. Penulis juga mengimplementasikan metode klasifikasi K-Nearest Neigbour (k-NN) sebagai metode klasifikasi yang akan digunakan untuk mengelompokkan email.

Arsitektur yang digunakan untuk membangun tugas akhir ini adalah konsep Model-View-Controller (MVC) yang memisahkan antara lapisan model, tampilan dan controller. Dengan memakai konsep MVC ini, modularitas aplikasi dapat dijaga, sehingga usaha dan waktu yang dibutuhkan untuk uji coba maupun perbaikan dan perawatan aplikasi dapat ditekan.

1. Uji coba dan evaluasi

Pada tahap ini, perangkat lunak yang sudah dibuat akan diuji coba. Uji coba dilakukan dengan cara membandingkan pengelompokkan email yang dilakukan dengan sistem (menggunakan metode K-Nearest Neighbour) secara otomatis dengan pengelompokkan yang dilakukan oleh pengguna email secara manual. Uji coba ini dilakukan untuk mendapatkan tingkat akurasi dari pengelompokkan email yang dilakukan oleh sistem. Data yang digunakan dalam uji coba sistem berjumlah 100 yang terdiri dari 10 data uji dan 90 data latih. Uji coba akan dilakukan sebanyak 10 kali. Selain itu, uji coba perangkat lunak juga bertujuan untuk menemukan kesalahan-kesalahan (bug) pada kode program sedini mungkin sehingga dapat diperbaiki sesegera mungkin [8].

1. Penyusunan buku tugas akhir

Tahap terakhir dalam pengerjaan tugas akhir ini adalah penyusunan laporan yang memuat dokumentasi mengenai pembuatan perangkat lunak serta hasil dari implementasi perangkat lunak yang telah dibuat. Sistematika penulisan buku tugas akhir secara garis besar antara lain:

1. Pendahuluan
2. Latar Belakang
3. Rumusan Masalah
4. Batasan Tugas Akhir
5. Tujuan
6. Metodologi
7. Sistematika Penulisan
8. Tinjauan Pustaka
9. Desain dan Implementasi
10. Uji Coba dan Evaluasi
11. Kesimpulan dan Saran
12. Daftar Pustaka

# **JADWAL PENGERJAAN TUGAS AKHIR**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Tahapan | 2012 | | | | | | | | | | | | | | | | | | | | |
| Februari | | | | Maret | | | | April | | | | Mei | | | | | Juni | | | |
| Penyusunan Proposal |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Studi Literatur |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Perancangan Sistem |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Implementasi |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Uji coba dan Evaluasi |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Penyusunan Buku |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

# **DAFTAR PUSTAKA**

1. **Sommerville, Ian.** *Software Engineering.* Boston : Addison-Wesley, 2007. 9780321313799.
2. **Radicati, Sara.** *Email Statistics Report,2010.* The Radicati Group, Inc., April 2004. [Dikutip: 1 Maret 2012.] <http://www.radicati.com>.
3. **Y. Yang and X. Liu**, *A Re-examination of Text Categorization Methods*, dalam Proceedings 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR-99), Berkeley, 1999
4. **S. Jiang, G. Pang, M. Wu and L. Kuang**, *An improved K-nearest-neighbor algorithm for text categorization*, Expert Systems with Applications, vol. 39, pp. 1503-1509, 2011
5. **Johnson, R**. *Introduction to Spring Framework*, Mei 2005.[Dipetik 1 Maret 2011], dari TheServerSide.com: http://www.theserverside.com/news/1364527/Introduction-to-the-Spring-Framework
6. **Adriani, Mirna, Jelita Asian, Bobby Nazief, Seyed Mohammad Tahaghoghi and Hugh Williams,***Stemming Indonesian: A Confix-Stripping Approach*, ACM Transactions on Asian Language Information Processing, Vol. 6, No. 4, 2007.
7. **Purwarianti, Ayu**. *A Non Deterministic Indonesian Stemmer*, International Conference on Electrical Engineering and Informatics, Bandung, 2011.
8. **Godbole, Nina.** *Software Quality Assurance: Principles and Practice.* Oxford : Alpha Science, 2007. 1-84265-176-5.
9. **P.-N. Tan, M. Steinbach dan V. Kumar**, *Introduction to Data Mining*, 1st penyunt., Boston: Pearson Addison Wesley, 2006

# **LEMBAR PENGESAHAN**

Surabaya, 6 Maret 2012

Menyetujui,

|  |  |
| --- | --- |
| Dosen Pembimbing I,  **Dwi Sunaryono, S.Kom, M.Kom**  **NIP.** **197205281997021001** | Dosen Pembimbing II,  **Abdul Munif, S.Kom, M.Sc.** |