BAB 3
METODOLOGI PENELITIAN

Penelitian ini menggunakan pendekatan kuantitatif dengan tahapan-tahapan sebagai berikut:

3.1 Rancangan Sistem

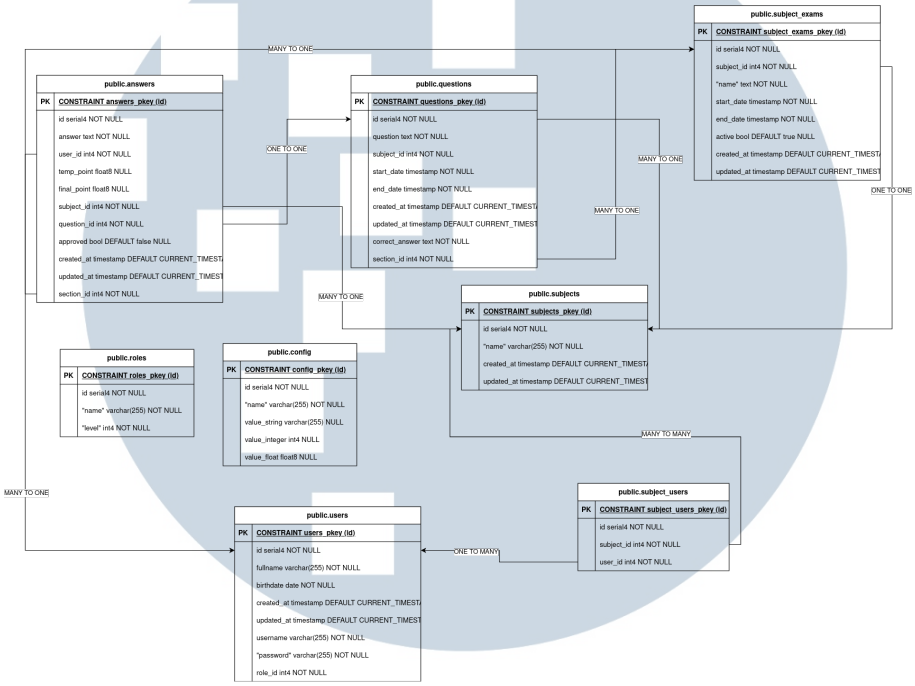
3.1.1 Analisa Fitur

Analisa fitur yang diperlukan untuk membantu implementasi algoritma pada penelitian ini.

Berikut daftar fitur utama yang dibuat pada sistem di penelitian ini:

No	Fitur	Deskripsi
1	Login	Fitur ini memungkinkan pengguna untuk masuk ke dalam sistem dengan menggunakan kredensial mereka.
2	Register	Fitur ini memungkinkan pengguna baru untuk mendaftar dan membuat akun baru di dalam sistem.
3	Melihat daftar kuis	Fitur ini memungkinkan pengguna untuk melihat semua kuis yang tersedia dalam sistem.
4	Menjawab kuis	Fitur ini memungkinkan pengguna untuk mengerjakan dan menjawab pertanyaan-pertanyaan dalam kuis yang dipilih.
5	Mengoreksi kuis	Fitur ini memungkinkan pengguna, biasanya pengajar, untuk memeriksa dan memberikan nilai terhadap jawaban kuis yang telah dikerjakan.
6	Membuat kuis	Fitur ini memungkinkan pengguna, biasanya pengajar, untuk membuat dan menambahkan kuis baru ke dalam sistem.

3.1.2 Diagram Database



Gambar 3.1. Diagram Database

Pada Diagram 3.1 memberikan gambaran relasi antar tabel yang digunakan. Tabel yang digunakan di penelitian ini adalah sebagai berikut:

1. Roles
2. Config
3. Questions
4. Answers
5. Subject Exams
6. Subject Users
7. Subjects
8. Users

A Relasi Antar Tabel

Dalam sistem basis data ini, terdapat beberapa tabel yang saling berelasi untuk mengelola data pengguna, peran, konfigurasi, mata pelajaran, ujian, pertanyaan, dan jawaban. Berikut adalah penjelasan relasi antar tabel tersebut:

Relasi Tabel Roles dan Tabel Users

Tabel `Roles` menyimpan data tentang peran (*roles*) yang dimiliki oleh pengguna. Tabel `Users` menyimpan data tentang pengguna, termasuk `role_id` yang merujuk ke kolom `id` pada tabel `Roles`. Relasi antara kedua tabel ini adalah satu-ke-banyak, di mana satu peran dapat dimiliki oleh banyak pengguna.

- `roles.id` → `users.role_id`

Relasi Tabel Users dan Tabel Subject_Users

Tabel `Subject_Users` berfungsi untuk menyimpan relasi antara pengguna (`Users`) dan mata pelajaran (`Subjects`). Tabel ini memiliki dua kolom `foreign key`, yaitu `user_id` yang merujuk ke kolom `id` pada tabel `Users` dan `subject_id` yang merujuk ke kolom `id` pada tabel `Subjects`.

- `users.id` → `subject_users.user_id`
- `subjects.id` → `subject_users.subject_id`

Relasi Tabel Subjects dan Tabel Subject_Exams

Tabel `Subject_Exams` menyimpan data tentang ujian (*exams*) yang terkait dengan mata pelajaran. Tabel ini memiliki kolom `foreign key` `subject_id` yang merujuk ke kolom `id` pada tabel `Subjects`. Relasi antara kedua tabel ini adalah satu-ke-banyak, di mana satu mata pelajaran dapat memiliki banyak ujian.

- `subjects.id` → `subject_exams.subject_id`

Relasi Tabel Subject_Exams dan Tabel Questions

Tabel `Questions` menyimpan data tentang pertanyaan yang terkait dengan ujian. Tabel ini memiliki dua kolom `foreign key`, yaitu `subject_id` yang

merujuk ke kolom `id` pada tabel `Subjects` dan `section_id` yang merujuk ke kolom `id` pada tabel `Subject_Exams`.

- `subjects.id` → `questions.subject_id`
- `subject_exams.id` → `questions.section_id`

Relasi Tabel Questions dan Tabel Answers

Tabel `Answers` menyimpan data tentang jawaban yang diberikan oleh pengguna. Tabel ini memiliki beberapa kolom `foreign key`: `question_id` yang merujuk ke kolom `id` pada tabel `Questions`, `user_id` yang merujuk ke kolom `id` pada tabel `Users`, `subject_id` yang merujuk ke kolom `id` pada tabel `Subjects`, dan `section_id` yang merujuk ke kolom `id` pada tabel `Subject_Exams`.

- `questions.id` → `answers.question_id`
- `users.id` → `answers.user_id`
- `subjects.id` → `answers.subject_id`
- `subject_exams.id` → `answers.section_id`

B Tabel Roles

Tabel ini berfungsi untuk menyimpan *roles* pada masing-masing pengguna, *roles* tersebut digunakan untuk mengatur akses pengguna terhadap menu berdasarkan tingkatan *roles* yang di berikan.

Tabel 3.2. Struktur Tabel Roles.

	Kolom	Tipe Data
id	serial4	NOT NULL
name	varchar(255)	NOT NULL
level	int4	NOT NULL

Penjelasan terhadap masing-masing kolom adalah sebagai berikut:

1. id: berfungsi untuk menyimpan id terhadap masing-masing *roles*
2. name: befungsi untuk menyimpan nama *roles*
3. level: berfungsi untuk menyimpan *level* atau tingkatan terhadap tiap *roles*

C Tabel Config

Tabel ini berfungsi untuk menyimpan konfigurasi terhadap *threshold*. Nilai tersebut disimpan untuk digunakan pada perhitungan skor pada proses penilaian otomatis.

Tabel 3.3. Struktur Tabel Config.

	Kolom	Tipe Data
id	serial4	NOT NULL
name	varchar(255)	NOT NULL
value_string	varchar(255)	NULL
value_integer	int4	NULL
value_float	float8	NULL

Penjelasan terhadap masing-masing kolom adalah sebagai berikut:

1. id: berfungsi untuk menyimpan id terhadap nilai konfigurasi.
2. name: befungsi untuk menyimpan nama *config*
3. value_string: berfungsi untuk menyimpan nilai *config* dalam bentuk *string*
4. value_integer: berfungsi untuk menyimpan nilai *config* dalam bentuk *integer*
5. value_float: berfungsi untuk menyimpan nilai *config* dalam bentuk *float*

D Tabel Users

Tabel ini berfungsi untuk menyimpan informasi *user*. Tabel ini digunakan untuk fitur seperti: *login*, *register*, menjawab atau membuat pertanyaan.

Tabel 3.4. Struktur Tabel Users.

	Kolom	Tipe Data
id	serial4	NOT NULL
fullname	varchar(255)	NOT NULL
birthdate	date	NOT NULL
created_at	timestamp	DEFAULT CURRENT_TIMESTAMP NULL
updated_at	timestamp	DEFAULT CURRENT_TIMESTAMP NULL
username	varchar(255)	NOT NULL
password	varchar(255)	NOT NULL
role_id	int4	NOT NULL

Penjelasan terhadap masing-masing kolom adalah sebagai berikut:

1. id: berfungsi untuk menyimpan id dari setiap pengguna.
2. fullname: berfungsi untuk menyimpan nama lengkap pengguna.
3. birthdate: berfungsi untuk menyimpan tanggal lahir pengguna.
4. created_at: berfungsi untuk menyimpan waktu kapan data pengguna dibuat, dengan nilai default adalah waktu saat data dimasukkan.
5. updated_at: berfungsi untuk menyimpan waktu kapan data pengguna terakhir diperbarui, dengan nilai default adalah waktu saat data dimasukkan.
6. username: berfungsi untuk menyimpan nama pengguna yang unik.
7. password: berfungsi untuk menyimpan kata sandi pengguna.
8. role_id: berfungsi untuk menyimpan id peran dari pengguna, yang menghubungkan ke tabel *roles*.

E Tabel Subject

Tabel ini berfungsi untuk menyimpan daftar mata pelajaran yang tersedia.

Tabel 3.5. Struktur Tabel Subjects.

	Kolom	Tipe Data
id	serial4	NOT NULL
name	varchar(255)	NOT NULL
created_at	timestamp	DEFAULT CURRENT_TIMESTAMP NULL
updated_at	timestamp	DEFAULT CURRENT_TIMESTAMP NULL

Penjelasan terhadap masing-masing kolom adalah sebagai berikut:

1. id: berfungsi untuk menyimpan id dari setiap mata pelajaran.
2. name: berfungsi untuk menyimpan nama mata pelajaran.
3. created_at: berfungsi untuk menyimpan waktu kapan data mata pelajaran dibuat, dengan nilai default adalah waktu saat data dimasukkan.
4. updated_at: berfungsi untuk menyimpan waktu kapan data mata pelajaran terakhir diperbarui, dengan nilai default adalah waktu saat data dimasukkan.

F Tabel Subject User

Tabel ini berfungsi untuk menyimpan relasi antara tabel *User* dengan tabel *Subject*.

Tabel 3.6. Struktur Tabel Subject_Users.

	Kolom	Tipe Data
id	serial4	NOT NULL
subject_id	int4	NOT NULL
user_id	int4	NOT NULL

Penjelasan terhadap masing-masing kolom adalah sebagai berikut:

1. id: berfungsi untuk menyimpan id dari setiap data dalam tabel ‘subject_users‘.
2. subject_id: berfungsi untuk menyimpan id mata pelajaran, yang menghubungkan ke tabel *subjects*.

3. user_id: berfungsi untuk menyimpan id pengguna, yang menghubungkan ke tabel *users*.

Tabel ‘subject_users‘ juga memiliki ‘foreign_key‘ sebagai berikut:

1. `subject_users_subject_id_fkey`: ‘foreign_key‘ yang menghubungkan kolom `subject_id` dengan kolom `id` pada tabel `subjects`.
2. `subject_users_user_id_fkey`: ‘foreign_key‘ yang menghubungkan kolom `user_id` dengan kolom `id` pada tabel `users`.

G Tabel Subject Exam

Tabel ini berfungsi untuk menyimpan daftar kuis essay yang tersedia.

Tabel 3.7. Struktur Tabel Subject_Exams.

	Kolom	Tipe Data
id	serial4	NOT NULL
subject_id	int4	NOT NULL
name	text	NOT NULL
start_date	timestamp	NOT NULL
end_date	timestamp	NOT NULL
active	bool	DEFAULT true NULL
created_at	timestamp	DEFAULT CURRENT_TIMESTAMP NULL
updated_at	timestamp	DEFAULT CURRENT_TIMESTAMP NULL

Penjelasan terhadap masing-masing kolom adalah sebagai berikut:

1. id: berfungsi untuk menyimpan id dari setiap dalam dalam tabel ‘subject_exams‘.
2. subject_id: berfungsi untuk menyimpan id mata pelajaran, yang menghubungkan ke tabel ‘subjects‘.
3. name: berfungsi untuk menyimpan nama ujian.
4. start_date: berfungsi untuk menyimpan tanggal dan waktu mulai ujian.
5. end_date: berfungsi untuk menyimpan tanggal dan waktu akhir ujian.

6. active: berfungsi untuk menyimpan status aktif dari ujian, dengan nilai default adalah true.
7. created_at: berfungsi untuk menyimpan waktu kapan data ujian dibuat, dengan nilai default adalah waktu saat data dimasukkan.
8. updated_at: berfungsi untuk menyimpan waktu kapan data ujian terakhir diperbarui, dengan nilai default adalah waktu saat data dimasukkan.

Tabel ‘subject_exams‘ juga memiliki ‘foreign_key‘ sebagai berikut:

1. `subject_exams_subject_id_fkey`: ‘foreign_key‘ yang menghubungkan kolom `subject_id` dengan kolom `id` pada tabel `subjects`.

H Tabel Question

Tabel ini berfungsi untuk menyimpan daftar pertanyaan serta daftar opsi jawaban yang benar.

Tabel 3.8. Struktur Tabel Questions.

	Kolom	Tipe Data
id	serial4	NOT NULL
question	text	NOT NULL
subject_id	int4	NOT NULL
start_date	timestamp	NOT NULL
end_date	timestamp	NOT NULL
created_at	timestamp	DEFAULT CURRENT_TIMESTAMP NULL
updated_at	timestamp	DEFAULT CURRENT_TIMESTAMP NULL
correct_answer	text	NOT NULL
section_id	int4	NOT NULL

Penjelasan terhadap masing-masing kolom adalah sebagai berikut:

1. id: berfungsi untuk menyimpan id dari setiap entri dalam tabel questions.
2. question: berfungsi untuk menyimpan teks pertanyaan.
3. subject_id: berfungsi untuk menyimpan id mata pelajaran, yang menghubungkan ke tabel `subjects`.

4. start_date: berfungsi untuk menyimpan tanggal dan waktu mulai pertanyaan dapat diakses.
5. end_date: berfungsi untuk menyimpan tanggal dan waktu akhir pertanyaan dapat diakses.
6. created_at: berfungsi untuk menyimpan waktu kapan data pertanyaan dibuat, dengan nilai default adalah waktu saat data dimasukkan.
7. updated_at: berfungsi untuk menyimpan waktu kapan data pertanyaan terakhir diperbarui, dengan nilai default adalah waktu saat data dimasukkan.
8. correct_answer: berfungsi untuk menyimpan jawaban benar dari pertanyaan.
9. section_id: berfungsi untuk menyimpan id bagian ujian, yang menghubungkan ke tabel `subject_exams`.

Tabel ‘questions‘ juga memiliki *foreign_key* sebagai berikut:

1. `questions_subject_id_fkey`: ‘foreign_key‘ yang menghubungkan kolom `subject_id` dengan kolom `id` pada tabel `subjects`.
2. `questions_section_id_fkey`: ‘foreign_key‘ yang menghubungkan kolom `section_id` dengan kolom `id` pada tabel `subject_exams`.

I Tabel Answers

Tabel ini berfungsi untuk menyimpan jawaban yang telah di-*input* oleh siswa, beserta dengan hasil penilaian.

Tabel 3.9. Struktur Tabel Answers.

	Kolom	Tipe Data
id	serial4	NOT NULL
answer	text	NOT NULL
user_id	int4	NOT NULL
temp_point	float8	NULL
final_point	float8	NULL
subject_id	int4	NOT NULL
question_id	int4	NOT NULL
approved	bool	DEFAULT false NULL
created_at	timestamp	DEFAULT CURRENT_TIMESTAMP NULL
updated_at	timestamp	DEFAULT CURRENT_TIMESTAMP NULL
section_id	int4	NOT NULL

Penjelasan terhadap masing-masing kolom adalah sebagai berikut:

1. id: berfungsi untuk menyimpan id dari setiap entri dalam tabel answers.
2. answer: berfungsi untuk menyimpan teks jawaban dari pengguna.
3. user_id: berfungsi untuk menyimpan id pengguna, yang menghubungkan ke tabel ‘users‘.
4. temp_point: berfungsi untuk menyimpan nilai sementara dari jawaban.
5. final_point: berfungsi untuk menyimpan nilai final dari jawaban.
6. subject_id: berfungsi untuk menyimpan id mata pelajaran, yang menghubungkan ke tabel ‘subjects‘.
7. question_id: berfungsi untuk menyimpan id pertanyaan, yang menghubungkan ke tabel ‘questions‘.
8. approved: berfungsi untuk menyimpan status persetujuan jawaban, dengan nilai default adalah false.
9. created_at: berfungsi untuk menyimpan waktu kapan data jawaban dibuat, dengan nilai default adalah waktu saat data dimasukkan.

10. updated_at: berfungsi untuk menyimpan waktu kapan data jawaban terakhir diperbarui, dengan nilai default adalah waktu saat data dimasukkan.

11. section_id: berfungsi untuk menyimpan id bagian ujian, yang menghubungkan ke tabel ‘subject_exams‘.

Tabel ‘answers‘ juga memiliki `foreign key` sebagai berikut:

1. `answers_question_id_fkey`: `foreign key` yang menghubungkan kolom `question_id` dengan kolom `id` pada tabel `questions`.
2. `answers_subject_id_fkey`: `foreign key` yang menghubungkan kolom `subject_id` dengan kolom `id` pada tabel `subjects`.
3. `answers_user_id_fkey`: `foreign key` yang menghubungkan kolom `user_id` dengan kolom `id` pada tabel `users`.
4. `answers_section_id_fkey`: `foreign key` yang menghubungkan kolom `section_id` dengan kolom `id` pada tabel `subject_exams`.

3.2 Praproses Teks

Praproses teks adalah langkah awal yang sangat penting dalam analisis teks untuk memastikan data yang akan dianalisis bersih dan sesuai. Tahapan praproses teks dalam penelitian ini meliputi:

1. **Tokenisasi:** Memecah teks menjadi unit-unit terkecil seperti kata atau frasa.
2. **Normalisasi:** Mengubah semua teks menjadi bentuk standar, seperti mengubah semua huruf menjadi huruf kecil dan menghilangkan tanda baca.
3. **Penghapusan Stopwords:** Menghapus kata-kata umum yang tidak memiliki makna penting seperti ”dan”, ”yang”, ”di”, dan lain-lain.

Potongan Kode 3.1 menunjukkan fungsi untuk melakukan praproses teks. Fungsi ini mencakup langkah-langkah tokenisasi, normalisasi, dan penghapusan stopwords.

```
1 // Fungsi ini mengembalikan slice dari token-token yang dihasilkan
      dari pemrosesan teks.
2 func preprocessText(text string) []string {
```

```
    // Memisahkan teks menjadi kata-kata dan mengubahnya menjadi huruf kecil
    words := strings.Fields(strings.ToLower(text))
    var result []string

    // Menambahkan setiap kata yang sudah diproses ke dalam slice hasil
    stopWords, _ := loadStopWords("indonesian.txt")
    for _, word := range words {
        // Hanya tambahkan kata ke hasil jika bukan stopword
        if !stopWords[word] {
            result = append(result, word)
        }
    }

    // Mencetak teks yang telah diproses dan mengembalikan hasilnya
    fmt.Println("Text preprocessed:", result)
    return result
}
```

Kode 3.1: Potongan Kode Praproses Teks

Langkah pertama dalam fungsi ini adalah melakukan tokenisasi dan normalisasi, yang mencakup memisahkan teks menjadi kata-kata dan mengubah semua huruf menjadi huruf kecil. Potongan kode untuk proses ini ditunjukkan pada Kode 3.2.

```
// Memisahkan teks menjadi kata-kata dan mengubahnya menjadi huruf kecil
words := strings.Fields(strings.ToLower(text))
```

Kode 3.2: Implementasi Kode Tokenisasi dan Normalisasi

Selanjutnya, fungsi ini melakukan penghapusan *stopwords*. *Stopwords* adalah kata-kata umum yang tidak memiliki makna penting dalam analisis teks dan perlu dihapus. Implementasi kode untuk penghapusan *stopwords* ditunjukkan pada Kode 3.3.

```
// Menambahkan setiap kata yang sudah diproses ke dalam slice hasil
stopWords, _ := loadStopWords("indonesian.txt")
for _, word := range words {
    // Hanya tambahkan kata ke hasil jika bukan stopword
    if !stopWords[word] {
        result = append(result, word)
```

```
    }
}
```

Kode 3.3: Implementasi Kode Penghapusan Stopwords

Setelah semua tahap praproses selesai, fungsi ini mencetak teks yang telah diproses dan mengembalikan hasilnya sebagai slice dari token-token. Langkah ini ditunjukkan pada Kode 3.4.

```
// Mencetak teks yang telah diproses dan mengembalikan hasilnya
fmt.Println("Text preprocessed:", result)
return result
```

Kode 3.4: Implementasi Kode Output Hasil Praproses

Dengan langkah-langkah praproses ini, teks yang akan dianalisis menjadi lebih bersih dan sesuai untuk tahap analisis berikutnya. Tokenisasi memecah teks menjadi unit-unit yang lebih kecil, normalisasi memastikan konsistensi format, penghapusan *stopwords* menghilangkan kata-kata yang tidak relevan, dan stemming mengembalikan kata-kata ke bentuk dasarnya.

3.3 Implementasi algoritma

Setelah praproses teks dilakukan, langkah selanjutnya adalah mengimplementasikan algoritma *Cosine Similarity* dan *TF-IDF* untuk pengoreksian otomatis esai. Implementasi dilakukan dengan menggunakan bahasa pemrograman Go. Tahapan implementasi meliputi:

3.3.1 Ekstraksi Fitur

Ekstraksi fitur dilakukan dengan menggunakan algoritma *Term Frequency-Inverse Document Frequency* (TF-IDF) setelah teks menjalani proses *pra-processing*.

Potongan Kode 3.5 menunjukkan fungsi untuk melakukan ekstraksi fitur menggunakan TF-IDF. Fungsi ini menerima dua *input*, yaitu `tokens`, sebuah *array string* dari kata-kata dalam dokumen yang akan diproses, dan `allDocuments`, sebuah kumpulan dokumen yang tersedia.

```
func calculateTFIDFVector(tokens []string, allDocuments [][]string
    ) map[string]float64 {
    tf := make(map[string]float64)
    idf := make(map[string]float64)

```

```
    // Menghitung frekuensi kemunculan token dalam dokumen (TF)
    for _, token := range tokens {
        tf[token]++
    }

    // Menghitung frekuensi kemunculan dokumen yang mengandung token (IDF)
    for _, document := range allDocuments {
        for _, token := range tokens {
            if containsToken(document, token) {
                idf[token]++
            }
        }
    }

    // Menghitung vektor TF-IDF
    tfidf := make(map[string]float64)
    for token, tfValue := range tf {
        tfidf[token] = (tfValue / float64(len(tokens))) * math.Log(float64(len(allDocuments))/(1+idf[token]))
    }

    // Mencetak vektor TF-IDF yang dihasilkan dan mengembalikan hasilnya
    fmt.Println("TF-IDF vector calculated:", tfidf)
    return tfidf
}
```

Kode 3.5: Potongan Kode Proses Ekstraksi Fitur

Proses pertama dalam fungsi ini adalah menghitung frekuensi kemunculan token dalam dokumen, yang dikenal sebagai *Term Frequency* (TF). Potongan kode untuk proses ini dapat dilihat pada Kode 3.6.

```
// Menghitung frekuensi kemunculan token dalam dokumen (TF)
for _, token := range tokens {
    tf[token]++
}
```

Kode 3.6: Implementasi Kode Perhitungan TF

Selanjutnya, fungsi ini menghitung frekuensi kemunculan dokumen yang mengandung token, yang dikenal sebagai *Inverse Document Frequency* (IDF). Implementasi kode ini ditunjukkan pada Kode 3.7.

```
// Menghitung frekuensi kemunculan dokumen yang mengandung token (IDF)
for _, document := range allDocuments {
    for _, token := range tokens {
        if containsToken(document, token) {
            idf[token]++
        }
    }
}
```

Kode 3.7: Implementasi Kode Perhitungan IDF

Terakhir, fungsi ini menghitung nilai TF-IDF untuk setiap token dengan menggunakan formula yang menggabungkan nilai TF dan IDF. Potongan kode ini ditunjukkan pada Kode 3.8.

```
// Menghitung vektor TF-IDF
tfidf := make(map[string]float64)
for token, tfValue := range tf {
    tfidf[token] = (tfValue / float64(len(tokens))) * math.Log(float64(len(allDocuments))/(1+idf[token]))
}
```

Kode 3.8: Implementasi Kode Perhitungan TF-IDF

Dengan mengimplementasikan langkah-langkah di atas, kita dapat memperoleh vektor TF-IDF yang mencerminkan pentingnya setiap token dalam konteks seluruh kumpulan dokumen. Vektor ini kemudian dapat digunakan untuk berbagai tujuan analisis teks dan pemrosesan bahasa alami.

3.3.2 Perhitungan Kesamaan

Perhitungan kesamaan antara teks yang diprediksi dan teks referensi dilakukan dengan menggunakan algoritma *Cosine Similarity*. Algoritma ini mengukur kesamaan antara dua vektor dengan menghitung sudut kosinus di antara mereka. Semakin kecil sudutnya, semakin besar kesamaannya.

Potongan Kode 3.9 menunjukkan fungsi yang digunakan untuk menghitung nilai kesamaan kosinus antara dua vektor.

```
// Fungsi ini mengembalikan nilai kesamaan kosinus antara dua vektor.
func cosineSimilarity(vector1, vector2 map[string]float64) float64 {
    dotProduct := 0.0
```

```
    magnitude1 := 0.0
    magnitude2 := 0.0

    // Menghitung dot product dan magnitudo masing-masing vektor
    for token, value1 := range vector1 {
        if value2, exists := vector2[token]; exists {
            dotProduct += value1 * value2
        }
        magnitude1 += math.Pow(value1, 2)
    }

    for _, value2 := range vector2 {
        magnitude2 += math.Pow(value2, 2)
    }

    // Menghitung magnitudo akhir dan mengembalikan nilai kesamaan kosinus
    magnitude1 = math.Sqrt(magnitude1)
    magnitude2 = math.Sqrt(magnitude2)

    if magnitude1 == 0 || magnitude2 == 0 {
        return 0.0
    }

    // Mencetak hasil kesamaan kosinus yang dihitung
    fmt.Println("Cosine similarity calculated:", dotProduct/(magnitude1*magnitude2))
    return dotProduct / (magnitude1 * magnitude2)
}
```

Kode 3.9: Potongan Kode Perhitungan Kesamaan Kosinus

Proses pertama dalam fungsi ini adalah menghitung *dot product* dan magnitudo masing-masing vektor. *Dot product* merupakan hasil penjumlahan perkalian elemen-elemen yang bersesuaian dari kedua vektor. Potongan kode ini ditunjukkan pada Kode 3.10.

```
// Menghitung dot product dan magnitudo masing-masing vektor
for token, value1 := range vector1 {
    if value2, exists := vector2[token]; exists {
        dotProduct += value1 * value2
    }
    magnitude1 += math.Pow(value1, 2)
}
```

Kode 3.10: Implementasi Kode Perhitungan Dot Product

Selanjutnya, fungsi ini menghitung magnitudo dari vektor kedua. Magnitudo vektor adalah akar kuadrat dari jumlah kuadrat semua elemennya. Implementasi kode ini ditunjukkan pada Kode 3.11.

```
for _, value2 := range vector2 {
    magnitude2 += math.Pow(value2, 2)
}

// Menghitung magnitudo akhir dan mengembalikan nilai kesamaan
    kosinus
magnitude1 = math.Sqrt(magnitude1)
magnitude2 = math.Sqrt(magnitude2)
```

Kode 3.11: Implementasi Kode Perhitungan Magnitudo

Setelah magnitudo dari kedua vektor dihitung, langkah terakhir adalah menghitung nilai kesamaan kosinus dengan membagi *dot product* dengan hasil perkalian magnitudo kedua vektor. Potongan kode ini ditunjukkan pada Kode 3.12.

```
if magnitude1 == 0 || magnitude2 == 0 {
    return 0.0
}

// Mencetak hasil kesamaan kosinus yang dihitung
fmt.Println("Cosine similarity calculated:", dotProduct/(
    magnitude1*magnitude2))
return dotProduct / (magnitude1 * magnitude2)
```

Kode 3.12: Implementasi Kode Perhitungan Kesamaan Kosinus

Dengan begitu algoritma tersebut dapat menghitung nilai kesamaan kosinus yang mencerminkan derajat kesamaan antara dua vektor teks. Nilai ini berkisar antara -1 hingga 1, dimana nilai 1 menunjukkan kesamaan sempurna, nilai 0 menunjukkan tidak ada kesamaan, dan nilai -1 menunjukkan kesamaan yang berlawanan.

3.4 Pengukuran Efektivitas

Pengukuran efektivitas implementasi algoritma dilakukan dengan menggunakan metrik RMSE (Root Mean Square Error). Tahapan pengukuran meliputi:

1. **Pengumpulan Data:** Mengumpulkan data esai dari siswa yang telah dinilai oleh pengajar.

2. **Perbandingan Hasil:** Membandingkan hasil penilaian otomatis dengan hasil penilaian manual menggunakan RMSE.

3.5 Evaluasi Kinerja Sistem

Evaluasi kinerja sistem dilakukan untuk memastikan sistem berjalan sesuai dengan yang diharapkan. Tahapan evaluasi meliputi:

1. **Uji Coba:** Melakukan uji coba sistem dengan berbagai dataset untuk melihat kinerja sistem.
2. **Analisis Hasil:** Menganalisis hasil uji coba untuk mengidentifikasi kekuatan dan kelemahan sistem.