BAB V. IMPLEMENTASI DAN PENGUJIAN

5.1. Implementasi Data

Data yang di implementasikan berasal dari data jawaban salah satu soal kuis mata kuliah *Business Inteligence* pada salah satu kelas di Jurusan Teknologi Informasi Politeknik Negeri Malang dimana data jawaban totalnya berjumlah 20 data dengan rentang kata setiap datanya antara 24 sampai 416 kata dengan jumlah keseluruhan kata sebanyak 1683 kata. Selain itu untuk rentang karakter setiap datanya ada di antara 165 sampai 3050 karakter dengan jumlah keseluruhan karakter sebanyak 12770 karakter. Contoh sampel data dapat dilihat di tabel 5.1.

Tabel 5. 1. Sampel Data

Soal	Label Mahasiswa	Jawaban
Sebutkan apa yang anda ketahui tentang data warehouse dan hubungannya dengan BI	Mhs1	Data warehouse adalah suatu sistem yang digunakan untuk menyimpan dan mengelola data yang terintegrasi dari berbagai sumber data yang berbeda, dan diorganisasikan sedemikian rupa sehingga dapat diakses dan digunakan untuk keperluan analisis dan pelaporan. Data warehouse biasanya digunakan untuk menyimpan data historis dalam jumlah besar dan beragam, serta dilengkapi dengan alat analisis data yang memungkinkan pengguna untuk memperoleh wawasan yang mendalam dan berharga dari data tersebut. Data warehouse dan Business Intelligence (BI) saling terkait, karena BI memanfaatkan data warehouse sebagai sumber datanya. BI merujuk pada kumpulan aplikasi, teknologi, dan praktik bisnis yang memungkinkan organisasi untuk menganalisis data bisnis dan informasi operasional untuk mendukung pengambilan keputusan yang lebih baik. Dalam konteks ini, data warehouse menyimpan dan menyediakan data yang diperlukan oleh BI untuk menganalisis dan melaporkan informasi bisnis yang penting bagi organisasi. Dengan menggunakan data warehouse sebagai basis datanya, BI dapat membantu organisasi untuk mengubah data menjadi informasi yang berguna dan bermanfaat, mempercepat waktu respons dalam pengambilan keputusan, meningkatkan

		efisiensi operasional, dan membantu mengidentifikasi peluang bisnis baru. Sebagai contoh, dengan memanfaatkan data warehouse, BI dapat membantu manajer dalam mengevaluasi kinerja bisnis, memprediksi tren dan pola bisnis, melakukan segmentasi pelanggan, dan melakukan analisis risiko bisnis.

5.1.1. *Preprocessing*

Preprocessing disini berguna untuk mengubah data kotor menjadi darta bersih. *Preprocessing* menggunakan bantuan *library* Sastrawi dan terdiri dari lima tahapan yang berkaitan satu sama lainnya yaitu *case folding, tokenization, stemming*, dan *stopword removal*.

1. *Case Folding*

Case folding akan mengubah huruf kapital menjadi huruf kecil.

Sampel dari proses *case folding* dapat dilihat pada tabel 5.2.

Tabel 5. 2. Sampel Data *Case Folding*

Data Kotor	*Case Folding*
Data warehouse adalah suatu sistem yang digunakan untuk menyimpan dan mengelola data yang terintegrasi dari berbagai sumber data yang berbeda, dan diorganisasikan sedemikian rupa sehingga dapat diakses dan digunakan untuk keperluan analisis dan pelaporan. Data warehouse biasanya digunakan untuk menyimpan data historis dalam jumlah besar dan beragam, serta dilengkapi dengan alat analisis data yang memungkinkan pengguna untuk memperoleh wawasan yang mendalam dan berharga dari data tersebut. Data warehouse dan Business Intelligence (BI) saling terkait, karena BI memanfaatkan data warehouse sebagai sumber datanya. BI merujuk pada kumpulan aplikasi, teknologi, dan praktik bisnis yang memungkinkan organisasi untuk menganalisis data bisnis dan informasi operasional untuk mendukung pengambilan keputusan yang lebih	data warehouse adalah suatu sistem yang digunakan untuk menyimpan dan mengelola data yang terintegrasi dari berbagai sumber data yang berbeda, dan diorganisasikan sedemikian rupa sehingga dapat diakses dan digunakan untuk keperluan analisis dan pelaporan. data warehouse biasanya digunakan untuk menyimpan data historis dalam jumlah besar dan beragam, serta dilengkapi dengan alat analisis data yang memungkinkan pengguna untuk memperoleh wawasan yang mendalam dan berharga dari data tersebut. data warehouse dan business intelligence (bi) saling terkait, karena bi memanfaatkan data warehouse sebagai sumber datanya. bi merujuk pada kumpulan aplikasi, teknologi, dan praktik bisnis yang memungkinkan organisasi untuk menganalisis data bisnis dan informasi operasional untuk mendukung pengambilan keputusan yang lebih

baik. Dalam konteks ini, data warehouse menyimpan dan menyediakan data yang diperlukan oleh BI untuk menganalisis dan melaporkan informasi bisnis yang penting bagi organisasi. Dengan menggunakan data warehouse sebagai basis datanya, BI dapat membantu organisasi untuk mengubah data menjadi informasi yang berguna dan bermanfaat, mempercepat waktu respons dalam pengambilan keputusan, meningkatkan efisiensi operasional, dan membantu mengidentifikasi peluang bisnis baru. Sebagai contoh, dengan memanfaatkan data warehouse, BI dapat membantu manajer dalam mengevaluasi kinerja bisnis, memprediksi tren dan pola bisnis, melakukan segmentasi pelanggan, dan melakukan analisis risiko bisnis.	baik. dalam konteks ini, data warehouse menyimpan dan menyediakan data yang diperlukan oleh bi untuk menganalisis dan melaporkan informasi bisnis yang penting bagi organisasi. dengan menggunakan data warehouse sebagai basis datanya, bi dapat membantu organisasi untuk mengubah data menjadi informasi yang berguna dan bermanfaat, mempercepat waktu respons dalam pengambilan keputusan, meningkatkan efisiensi operasional, dan membantu mengidentifikasi peluang bisnis baru. sebagai contoh, dengan memanfaatkan data warehouse, bi dapat membantu manajer dalam mengevaluasi kinerja bisnis, memprediksi tren dan pola bisnis, melakukan segmentasi pelanggan, dan melakukan analisis risiko bisnis.

2. *Tokenization*

Tokenization akan membagi kalimat menjadi kata. Sampel dari proses *tokenization* dapat dilihat pada tabel 5.3.

Tabel 5. 3. Sampel Data *Tokenization*

Hasil *Case Folding*	*Tokenization*
data warehouse adalah suatu sistem yang digunakan untuk menyimpan dan mengelola data yang terintegrasi dari berbagai sumber data yang berbeda, dan diorganisasikan sedemikian rupa sehingga dapat diakses dan digunakan untuk keperluan analisis dan pelaporan. data warehouse biasanya digunakan untuk menyimpan data historis dalam jumlah besar dan beragam, serta dilengkapi dengan alat analisis data yang memungkinkan pengguna untuk memperoleh wawasan yang mendalam dan berharga dari data tersebut. data warehouse dan business intelligence (bi) saling terkait, karena bi memanfaatkan data warehouse	['data', 'warehouse', 'adalah', 'suatu', 'sistem', 'yang', 'digunakan', 'untuk', 'menyimpan', 'dan', 'mengelola', 'data', 'yang', 'terintegrasi', 'dari', 'berbagai', 'sumber', 'data', 'yang', 'berbeda,', 'dan', 'diorganisasikan', 'sedemikian', 'rupa', 'sehingga', 'dapat', 'diakses', 'dan', 'digunakan', 'untuk', 'keperluan', 'analisis', 'dan', 'pelaporan.', 'data', 'warehouse', 'biasanya', 'digunakan', 'untuk', 'menyimpan', 'data', 'historis', 'dalam', 'jumlah', 'besar', 'dan', 'beragam,', 'serta', 'dilengkapi', 'dengan', 'alat', 'analisis', 'data', 'yang', 'memungkinkan', 'pengguna', 'untuk', 'memperoleh', 'wawasan', 'yang', 'mendalam', 'dan', 'berharga', 'dari',

sebagai sumber datanya. bi merujuk pada kumpulan aplikasi, teknologi, dan praktik bisnis yang memungkinkan organisasi untuk menganalisis data bisnis dan informasi operasional untuk mendukung pengambilan keputusan yang lebih baik. dalam konteks ini, data warehouse menyimpan dan menyediakan data yang diperlukan oleh bi untuk menganalisis dan melaporkan informasi bisnis yang penting bagi organisasi. dengan menggunakan data warehouse sebagai basis datanya, bi dapat membantu organisasi untuk mengubah data menjadi informasi yang berguna dan bermanfaat, mempercepat waktu respons dalam pengambilan keputusan, meningkatkan efisiensi operasional, dan membantu mengidentifikasi peluang bisnis baru. sebagai contoh, dengan memanfaatkan data warehouse, bi dapat membantu manajer dalam mengevaluasi kinerja bisnis, memprediksi tren dan pola bisnis, melakukan segmentasi pelanggan, dan melakukan analisis risiko bisnis.	'data', 'tersebut.', 'data', 'warehouse', 'dan', 'business', 'intelligence', '(bi)', 'saling', 'terkait,', 'karena', 'bi', 'memanfaatkan', 'data', 'warehouse', 'sebagai', 'sumber', 'datanya.', 'bi', 'merujuk', 'pada', 'kumpulan', 'aplikasi,', 'teknologi,', 'dan', 'praktik', 'bisnis', 'yang', 'memungkinkan', 'organisasi', 'untuk', 'menganalisis', 'data', 'bisnis', 'dan', 'informasi', 'operasional', 'untuk', 'mendukung', 'pengambilan', 'keputusan', 'yang', 'lebih', 'baik.', 'dalam', 'konteks', 'ini,', 'data', 'warehouse', 'menyimpan', 'dan', 'menyediakan', 'data', 'yang', 'diperlukan', 'oleh', 'bi', 'untuk', 'menganalisis', 'dan', 'melaporkan', 'informasi', 'bisnis', 'yang', 'penting', 'bagi', 'organisasi.', 'dengan', 'menggunakan', 'data', 'warehouse', 'sebagai', 'basis', 'datanya,', 'bi', 'dapat', 'membantu', 'organisasi', 'untuk', 'mengubah', 'data', 'menjadi', 'informasi', 'yang', 'berguna', 'dan', 'bermanfaat,', 'mempercepat', 'waktu', 'respons', 'dalam', 'pengambilan', 'keputusan,', 'meningkatkan', 'efisiensi', 'operasional,', 'dan', 'membantu', 'mengidentifikasi', 'peluang', 'bisnis', 'baru.', 'sebagai', 'contoh,', 'dengan', 'memanfaatkan', 'data', 'warehouse,', 'bi', 'dapat', 'membantu', 'manajer', 'dalam', 'mengevaluasi', 'kinerja', 'bisnis,', 'memprediksi', 'tren', 'dan', 'pola', 'bisnis,', 'melakukan', 'segmentasi', 'pelanggan,', 'dan', 'melakukan', 'analisis', 'risiko', 'bisnis.']

3. Stemming

Stemming akan mengubah kata menjadi kata dasarnya. Sampel dari *stemming* dapat dilihat di tabel 5.4.

Tabel 5. 4. Sampel Data *Stemming*

Hasil *Tokenization*	*Stemming*

['data', 'warehouse', 'adalah', 'suatu', 'sistem', 'yang', 'digunakan', 'untuk', 'menyimpan', 'dan', 'mengelola', 'data', 'yang', 'terintegrasi', 'dari', 'berbagai', 'sumber', 'data', 'yang', 'berbeda,', 'dan', 'diorganisasikan', 'sedemikian', 'rupa', 'sehingga', 'dapat', 'diakses', 'dan', 'digunakan', 'untuk', 'keperluan', 'analisis', 'dan', 'pelaporan.', 'data', 'warehouse', 'biasanya', 'digunakan', 'untuk', 'menyimpan', 'data', 'historis', 'dalam', 'jumlah', 'besar', 'dan', 'beragam,', 'serta', 'dilengkapi', 'dengan', 'alat', 'analisis', 'data', 'yang', 'memungkinkan', 'pengguna', 'untuk', 'memperoleh', 'wawasan', 'yang', 'mendalam', 'dan', 'berharga', 'dari', 'data', 'tersebut.', 'data', 'warehouse', 'dan', 'business', 'intelligence', '(bi)', 'saling', 'terkait,', 'karena', 'bi', 'memanfaatkan', 'data', 'warehouse', 'sebagai', 'sumber', 'datanya.', 'bi', 'merujuk', 'pada', 'kumpulan', 'aplikasi,', 'teknologi,', 'dan', 'praktik', 'bisnis', 'yang', 'memungkinkan', 'organisasi', 'untuk', 'menganalisis', 'data', 'bisnis', 'dan', 'informasi', 'operasional', 'untuk', 'mendukung', 'pengambilan', 'keputusan', 'yang', 'lebih', 'baik.', 'dalam', 'konteks', 'ini,', 'data', 'warehouse', 'menyimpan', 'dan', 'menyediakan', 'data', 'yang', 'diperlukan', 'oleh', 'bi', 'untuk', 'menganalisis', 'dan', 'melaporkan', 'informasi', 'bisnis', 'yang', 'penting', 'bagi', 'organisasi.', 'dengan', 'menggunakan', 'data', 'warehouse', 'sebagai', 'basis', 'datanya,', 'bi', 'dapat', 'membantu', 'organisasi', 'untuk', 'mengubah', 'data', 'menjadi', 'informasi', 'yang', 'berguna', 'dan', 'bermanfaat,', 'mempercepat', 'waktu', 'respons', 'dalam', 'pengambilan', 'keputusan,', 'meningkatkan', 'efisiensi', 'operasional,', 'dan', 'membantu', 'mengidentifikasi', 'peluang', 'bisnis', 'baru.', 'sebagai', 'contoh,', 'dengan', 'memanfaatkan',	['data', 'warehouse', 'adalah', 'suatu', 'sistem', 'yang', 'guna', 'untuk', 'simpan', 'dan', 'kelola', 'data', 'yang', 'integrasi', 'dari', 'bagai', 'sumber', 'data', 'yang', 'beda', 'dan', 'organisasi', 'demikian', 'rupa', 'sehingga', 'dapat', 'akses', 'dan', 'guna', 'untuk', 'perlu', 'analisis', 'dan', 'lapor', 'data', 'warehouse', 'biasa', 'guna', 'untuk', 'simpan', 'data', 'historis', 'dalam', 'jumlah', 'besar', 'dan', 'agam', 'serta', 'lengkap', 'dengan', 'alat', 'analisis', 'data', 'yang', 'mungkin', 'guna', 'untuk', 'oleh', 'wawas', 'yang', 'dalam', 'dan', 'harga', 'dari', 'data', 'sebut', 'data', 'warehouse', 'dan', 'business', 'intelligence', 'bi', 'saling', 'kait', 'karena', 'bi', 'manfaat', 'data', 'warehouse', 'bagai', 'sumber', 'data', 'bi', 'rujuk', 'pada', 'kumpul', 'aplikasi', 'teknologi', 'dan', 'praktik', 'bisnis', 'yang', 'mungkin', 'organisasi', 'untuk', 'analis', 'data', 'bisnis', 'dan', 'informasi', 'operasional', 'untuk', 'dukung', 'ambil', 'putus', 'yang', 'lebih', 'baik', 'dalam', 'konteks', 'ini', 'data', 'warehouse', 'simpan', 'dan', 'sedia', 'data', 'yang', 'perlu', 'oleh', 'bi', 'untuk', 'analis', 'dan', 'lapor', 'informasi', 'bisnis', 'yang', 'penting', 'bagi', 'organisasi', 'dengan', 'guna', 'data', 'warehouse', 'bagai', 'basis', 'data', 'bi', 'dapat', 'bantu', 'organisasi', 'untuk', 'ubah', 'data', 'jadi', 'informasi', 'yang', 'guna', 'dan', 'manfaat', 'cepat', 'waktu', 'respons', 'dalam', 'ambil', 'putus', 'tingkat', 'efisiensi', 'operasional', 'dan', 'bantu', 'identifikasi', 'peluang', 'bisnis', 'baru', 'bagai', 'contoh', 'dengan', 'manfaat', 'data', 'warehouse', 'bi', 'dapat', 'bantu', 'manajer', 'dalam', 'evaluasi', 'kerja', 'bisnis', 'prediksi', 'tren', 'dan', 'pola', 'bisnis', 'laku', 'segmentasi', 'langgan', 'dan', 'laku', 'analisis', 'risiko', 'bisnis']

'data', 'warehouse,', 'bi', 'dapat', 'membantu', 'manajer', 'dalam', 'mengevaluasi', 'kinerja', 'bisnis,', 'memprediksi', 'tren', 'dan', 'pola', 'bisnis,', 'melakukan', 'segmentasi', 'pelanggan,', 'dan', 'melakukan', 'analisis', 'risiko', 'bisnis.']	

4. Stopword Removal

Stopword Removal akan menghapus kata yang tidak relevan, imbuhan, dan kata yang sering muncul. Selain itu di proses ini juga dilakukan persatuan kembali data menjadi bentuk kalimat. Sampel dari proses stopword removal dilihat di tabel 5.5.

Tabel 5. 5. Sampel Data *Stopword Removal*

Hasil *Stemming*	*Stopword Removal*
['data', 'warehouse', 'adalah', 'suatu', 'sistem', 'yang', 'guna', 'untuk', 'simpan', 'dan', 'kelola', 'data', 'yang', 'integrasi', 'dari', 'bagai', 'sumber', 'data', 'yang', 'beda', 'dan', 'organisasi', 'demikian', 'rupa', 'sehingga', 'dapat', 'akses', 'dan', 'guna', 'untuk', 'perlu', 'analisis', 'dan', 'lapor', 'data', 'warehouse', 'biasa', 'guna', 'untuk', 'simpan', 'data', 'historis', 'dalam', 'jumlah', 'besar', 'dan', 'agam', 'serta', 'lengkap', 'dengan', 'alat', 'analisis', 'data', 'yang', 'mungkin', 'guna', 'untuk', 'oleh', 'wawas', 'yang', 'dalam', 'dan', 'harga', 'dari', 'data', 'sebut', 'data', 'warehouse', 'dan', 'business', 'intelligence', 'bi', 'saling', 'kait', 'karena', 'bi', 'manfaat', 'data', 'warehouse', 'bagai', 'sumber', 'data', 'bi', 'rujuk', 'pada', 'kumpul', 'aplikasi', 'teknologi', 'dan', 'praktik', 'bisnis', 'yang', 'mungkin', 'organisasi', 'untuk', 'analis', 'data', 'bisnis', 'dan', 'informasi', 'operasional', 'untuk', 'dukung', 'ambil', 'putus', 'yang', 'lebih', 'baik', 'dalam', 'konteks', 'ini', 'data', 'warehouse', 'simpan', 'dan', 'sedia', 'data', 'yang', 'perlu', 'oleh', 'bi', 'untuk', 'analis', 'dan', 'lapor', 'informasi',	data warehouse suatu sistem simpan kelola data integrasi bagai sumber data beda organisasi rupa akses perlu analisis lapor data warehouse biasa guna simpan data historis jumlah besar agam lengkap alat analisis data mungkin guna wawas harga data sebut data warehouse business intelligence bi saling kait bi manfaat data warehouse bagai sumber data bi rujuk kumpul aplikasi teknologi praktik bisnis mungkin organisasi analis data bisnis informasi operasional dukung ambil putus lebih baik konteks data warehouse simpan sedia data perlu bi untuk analis lapor informasi bisnis penting organisasi guna data warehouse bagai basis data bi bantu organisasi untuk ubah data jadi informasi guna manfaat cepat waktu respons ambil putus tingkat efisiensi operasional bantu identifikasi peluang bisnis baru bagai contoh manfaat data warehouse bi dapat bantu manajer dalam evaluasi kerja bisnis prediksi tren pola bisnis laku segmentasi langgan laku analisis risiko bisnis

'bisnis', 'yang', 'penting', 'bagi', 'organisasi', 'dengan', 'guna', 'data', 'warehouse', 'bagai', 'basis', 'data', 'bi', 'dapat', 'bantu', 'organisasi', 'untuk', 'ubah', 'data', 'jadi', 'informasi', 'yang', 'guna', 'dan', 'manfaat', 'cepat', 'waktu', 'respons', 'dalam', 'ambil', 'putus', 'tingkat', 'efisiensi', 'operasional', 'dan', 'bantu', 'identifikasi', 'peluang', 'bisnis', 'baru', 'bagai', 'contoh', 'dengan', 'manfaat', 'data', 'warehouse', 'bi', 'dapat', 'bantu', 'manajer', 'dalam', 'evaluasi', 'kerja', 'bisnis', 'prediksi', 'tren', 'dan', 'pola', 'bisnis', 'laku', 'segmentasi', 'langgan', 'dan', 'laku', 'analisis', 'risiko', 'bisnis']	

Adapun kode proses yang dijalankan pada sistem dapat dilihat sebagai dibawah.

```
def process_jawabans_sastrawi(jawabans):
    # Initialize Sastrawi stemmer and stopword remover
    stemmer = StemmerFactory().create_stemmer()
    stopword_remover =
StopWordRemoverFactory().create_stop_word_remover()

    processed_jawabans = []
    for jawaban in jawabans:
        jawaban = jawaban.lower()
        tokens = jawaban.split()
        jawaban = [stemmer.stem(token) for token in tokens]
        jawaban = stopword_remover.remove(' '.join(jawaban))
        jawaban = ' '.join(jawaban.split())
        processed_jawabans.append(jawaban)

    return processed_jawabans
```

Pada potongan kode diatas dapat dilihat bahwa semua proses *preprocessing* ada pada perulangan jawaban, hal ini menunjukkan bahwa *preprocessing* dijalankan kepada setiap data jawaban.

5.2. Implementasi Algoritma

Implementasi algoritma dilakukan untuk mengubah data bersih menjadi data klaster yang diinginkan. Untuk proses ini peneliti menggunakan bantuan

library sklearn. Terdapat tiga algoritma yang akan di implementasikan yaitu *vectorizing, clustering,* dan *testing.*

5.2.1. *Vectorizing*

Proses *vectorizing* menggunakan *tf-idf* dimana data perlu dijadikan vektor terlebih dahulu agar bisa dilanjutkan ke proses *clustering*. Seperti yang ditunjukan pada kode dibawah.

```
def perform_tfidf(processed_jawabans):
    vectorizer = TfidfVectorizer()
    tfidf_matrix = vectorizer.fit_transform(processed_jawabans)

    return tfidf_matrix()
```

Pada potongan kode diatas *tf-idf* menggunakan data bersih hasil *preprocessing* yang bernama *processed_jawabans* yang kemudian akan mengeluarkan output berupa data vektor yang ditunjukkan dengan *tfidf_matrix*.

5.2.2. *Clustering*

Proses *clustering* akan menggunakan algoritma *kmeans* yang berguna untuk mengelompokkan data terhadap data vektor dari proses *vectorizing*, seperti yang ditunjukkan pada potongan kode dibawah.

```
length_cluster = len(jawabans)
max_clusters = length_cluster - 1
def perform_kmeans_clustering(tfidf_matrix, max_clusters):
    min_clusters = 2
    for num_clusters in range(min_clusters, max_clusters + 1):
    # KMeans clustering
        kmeans = KMeans(n_clusters=num_clusters, random_state=42)
        kmeans.fit(tfidf_matrix)
        cluster_labels = kmeans.labels_ + 1
        print("Number of Clusters:", num_clusters)
        print("Cluster Labels:", cluster_labels)
```

Pada potongan kode diatas dapat dilihat proses *clustering* yang digunakan berada dalam perulangan, hal ini terjadi agar sistem mengulang proses sebanyak jumlah data yang diproses dan agar lebih mudah untuk proses *testing* nanti. Proses *clustering* sendiri menggunakan data vektor *tfidf_matrix* sebagai *input* dan hasil klaster berupa *cluster_labels* sebagai *output.*

5.2.3. *Testing*

Untuk proses *testing* dilakukan menggunakan *silhouette coefficient*, proses ini berguna untuk mengetahui klaster mana yang paling bagus berdasarkan nilai *silhouette*, seperti yang dapat dilihat pada potongan kode dibawah.

```
length_cluster = len(jawabans)
max_clusters = length_cluster - 1
def perform_kmeans_clustering(tfidf_matrix, max_clusters):

    min_clusters = 2
    silhouette_scores = []

    for num_clusters in range(min_clusters, max_clusters + 1):
    # KMeans clustering
        kmeans = KMeans(n_clusters=num_clusters, random_state=42)
        kmeans.fit(tfidf_matrix)
        cluster_labels = kmeans.labels_ + 1
        print("Number of Clusters:", num_clusters)
        print("Cluster Labels:", cluster_labels)

        # Silhouette score
        silhouette_avg = silhouette_score(tfidf_matrix, cluster_labels)
        silhouette_scores.append(silhouette_avg)
        print("sil:", silhouette_avg)

    # Find the index of the highest silhouette score
    max_silhouette_index = silhouette_scores.index(max(silhouette_scores))

    # Get the number of clusters corresponding to the highest silhouette score
    best_num_clusters = max_silhouette_index + min_clusters

    # Perform KMeans clustering with the best number of clusters
    best_kmeans = KMeans(n_clusters=best_num_clusters, random_state=42)
    best_kmeans.fit(tfidf_matrix)
    best_cluster_labels = best_kmeans.labels_ + 1
    print("Best Number of Clusters:", best_num_clusters)
    print("BEst Clusters label:", best_cluster_labels)

    return best_cluster_labels.tolist(), silhouette_scores, best_num_clusters  #
Convert NumPy array to Python list
```

Pada potongan kode diatas dapat kita lihat bahwa perhitungan *silhouette* yang ditandai dengan # *Silhouette score* ada didalam perulangan klaster, hal ini menunjukkan bahwa setiap dilakukan proses *clustering* maka akan dilakukan juga proses perhitungan *silhouette*. Setelah semua nilai *silhouette* terhadap setiap jumlah

klaster didapatkan, akan dihitung nilai terbesar ada pada jumlah klaster berapa yang akhirnya jumlah klaster itulah yang akan menjadi klaster rekomendasi sistem yang ditandai dengan *best_num_clusters*. Selain itu dengan nilai *silhouette* kita dapat mengetahui seberapa bagus persebaran data dan seberapa cocok data yang diolah pada algoritma.

5.3. Implementasi Sistem

Pada implementasi sistem peneliti akan memaparkan cara mengimplementasikan kebutuhan fungsional kepada sistem. Implementasi sistem terdiri dari dua bagian yaitu implementasi *database*, dan implementasi antarmuka.

5.3.1. Implementasi *Database*

Pada implementasi *database* disini peneliti menggunakan *database mysql* dengan bahasa pemrograman *javascript*. Berdasarkan analisis kebutuhan fungsional setidaknya ada lima tabel yang diperlukan oleh sistem yaitu tabel untuk *user*, kategori, soal, jawaban, dan *session*.

1. Tabel Users

   Tabel users berguna untuk menyimpan data *user* atau pengguna dari sistem. Disini pengguna membutuhkan *id* dengan tipe data *integer*, dan *uuid* dengan tipe data *varchar* sebagai kode unik, *name* dengan tipe data *varchar* sebagai identitas utama, *username* dan *password* dengan tipe data *varchar* untuk keperluan *login*, *role* dengan tipe data *varchar* untuk membedakkan jenis pengguna dan *createdAt*, *updatedAt* dengan tipe data *datetime* untuk dokumentasi waktu ketika ada perubahan data. Untuk rincian tabel userd dapat dilihat pada tabel 5.6.

Tabel 5. 6. Implementasi Tabel Users

users	
Kolom	Tipe Data
id	*int*
uuid	*varchar*

name	*varchar*
username	*varchar*
password	*varchar*
role	*varchar*
createdAt	*datetime*
updatedAt	*datetime*

```
const Users = db.define('users',{
    uuid:{
        type: DataTypes.STRING,
        defaultValue: DataTypes.UUIDV4,
        allowNull: false,
        validate:{
            notEmpty: true
        }
    },
    name:{
        type: DataTypes.STRING,
        allowNull: false,
        validate:{
            notEmpty: true,
            len: [1, 100]
        }
    },
    username:{
        type: DataTypes.STRING,
        allowNull: false,
        validate:{
            notEmpty: true
        }
    },
    password:{
        type: DataTypes.STRING,
        allowNull: false,
        validate:{
            notEmpty: true
        }
    },
    role:{
        type: DataTypes.STRING,
        allowNull: false,
        validate:{
            notEmpty: true
        }
```

```
    }
},{
    freezeTableName: true
});

export default Users;
```

Pada potongan kode diatas dapat dilihat proses pembuatan tabel users pada *database* dengan segala kolom beserta atributnya yang ditandai dengan kode *db.define*.

2. Tabel Kategoris

Tabel kategoris berguna untuk menyimpan data kategori soal. Disini kategori membutuhkan *id* dengan tipe data *integer*, dan *uuid* dengan tipe data *varchar* sebagai kode unik, *name* dengan tipe data *varchar* sebagai atribut utama, *userId* dengan tipe data *integer* sebagai *foreign key* untuk *id* dari tabel users dan *createdAt*, *updatedAt* dengan tipe data *datetime* untuk dokumentasi waktu ketika ada perubahan data. Untuk rincian tabel kategoris dapat dilihat pada tabel 5.7.

Tabel 5. 7. Implementasi Tabel Kategoris

kategoris	
Kolom	Tipe Data
id	*int*
uuid	*varchar*
name	*varchar*
userId	*int*
createdAt	*datetime*
updatedAt	*datetime*

```
const Kategoris = db.define('kategoris',{
    uuid:{
        type: DataTypes.STRING,
        defaultValue: DataTypes.UUIDV4,
        allowNull: false,
        validate:{
            notEmpty: true
        }
```

```
    },
    name:{
        type: DataTypes.STRING,
        allowNull: false,
        validate:{
            notEmpty: true,
            len: [1, 100]
        }
    },
    userId:{
        type: DataTypes.INTEGER,
        allowNull: false,
        validate:{
            notEmpty: true,
            len: [1, 100]
        }
    }
},{
    freezeTableName: true
});

export default Kategoris;
```

Pada potongan kode diatas dapat dilihat proses pembuatan tabel kategoris pada *database* dengan segala kolom beserta atributnya yang ditandai dengan kode *db.define*.

3. Tabel Soals

Tabel soals berguna untuk menyimpan data soal dosen. Disini soal membutuhkan *id* dengan tipe data *integer*, dan *uuid* dengan tipe data *varchar* sebagai kode unik, *name* dengan tipe data *varchar* sebagai atribut utama, *userId* dengan tipe data *integer* sebagai *foreign key* dari tabel users yang berguna untuk mengambil data *user*, *kategoriId* dengan tipe data *integer* sebagai *foreign key* dari tabel kategoris yang berguna untuk mengambil data kategori, dan *createdAt*, *updatedAt* dengan tipe data *datetime* untuk dokumentasi waktu ketika ada perubahan data. Untuk rincian tabel soals dapat dilihat pada tabel 5.8.

Tabel 5. 8. Implementasi Tabel Soals

soals	
Kolom	Tipe Data

id	*int*
uuid	*varchar*
name	*varchar*
userId	*int*
kategoriId	*int*
createdAt	*datetime*
updatedAt	*datetime*

```
const Soals = db.define('soals',{
    uuid:{
        type: DataTypes.STRING,
        defaultValue: DataTypes.UUIDV4,
        allowNull: false,
        validate:{
            notEmpty: true
        }
    },
    name:{
        type: DataTypes.STRING,
        allowNull: false,
        validate:{
            notEmpty: true,
            len: [1, 300]
        }
    },
    userId:{
        type: DataTypes.INTEGER,
        allowNull: false,
        validate:{
            notEmpty: true,
            len: [1, 100]
        }
    },
    kategoriId:{
        type: DataTypes.INTEGER,
        allowNull: false,
        validate:{
            notEmpty: true,
            len: [1, 100]
        }
    }
},{
    freezeTableName: true
});
```

```
export default Soals;
```

Pada potongan kode diatas dapat dilihat proses pembuatan tabel soals pada *database* dengan segala kolom beserta atributnya yang ditandai dengan kode *db.define*.

4. Tabel Jawabs

Tabel jawabs berguna untuk menyimpan data jawaban mahasiswa. Disini jawaban membutuhkan *id* dengan tipe data *integer*, dan *uuid* dengan tipe data *varchar* sebagai kode unik, *name* dengan tipe data *text* sebagai atribut utama, *userId* dengan tipe data *integer* sebagai *foreign key* dari tabel users yang berguna untuk mengambil data *user*, *kategoriId* dengan tipe data *integer* sebagai *foreign key* dari tabel kategoris yang berguna untuk mengambil data kategori, *soalId* dengan tipe data *integer* sebagai *foreign key* dari tabel soals yang berguna untuk mengambil data soal, dan *createdAt, updatedAt* dengan tipe data *datetime* untuk dokumentasi waktu ketika ada perubahan data. Untuk rincian tabel jawabs dapat dilihat pada tabel 5.9.

Tabel 5. 9. Implementasi Tabel Jawabs

jawabs	
Kolom	Tipe Data
id	*int*
uuid	*varchar*
name	*text*
userId	*int*
kategoriId	*int*
soalId	*int*
createdAt	*datetime*
updatedAt	*datetime*

```
const Jawabs = db.define('jawabs',{
   uuid:{
      type: DataTypes.STRING,
```

```
      defaultValue: DataTypes.UUIDV4,
      allowNull: false,
      validate:{
         notEmpty: true
      }
   },
   name:{
      type: DataTypes.TEXT,
      allowNull: false,
      validate:{
         notEmpty: true
      }
   },
   userId:{
      type: DataTypes.INTEGER,
      allowNull: false,
      validate:{
         notEmpty: true,
         len: [1, 100]
      }
   },
   kategoriId:{
      type: DataTypes.INTEGER,
      allowNull: false,
      validate:{
         notEmpty: true,
         len: [1, 100]
      }
   },
   soalId:{
      type: DataTypes.INTEGER,
      allowNull: false,
      validate:{
         notEmpty: true,
         len: [1, 100]
      }
   }
},{
   freezeTableName: true
});

export default Jawabs;
```

Pada potongan kode diatas dapat dilihat proses pembuatan tabel jawabs pada *database* dengan segala kolom beserta atributnya yang ditandai dengan kode *db.define*.

5. Tabel Sessions

Tabel sessions berguna untuk menyimpan data *user* yang sedang *login* atau aktif didalam sistem. Disini sessions membutuhkan *sid* dengan tipe data *varchar* sebagai kode unik, *expires* dengan tipe data *datetime* sebagai waktu berakhirnya seorang pengguna aktif, *data* dengan tipe data *text* sebagai atribut utama, dan *createdAt, updatedAt* dengan tipe data *datetime* untuk dokumentasi waktu ketika ada perubahan data. Untuk rincian tabel sessions dapat dilihat pada tabel 5.10.

Tabel 5. 10. Implementasi Tabel Sessions

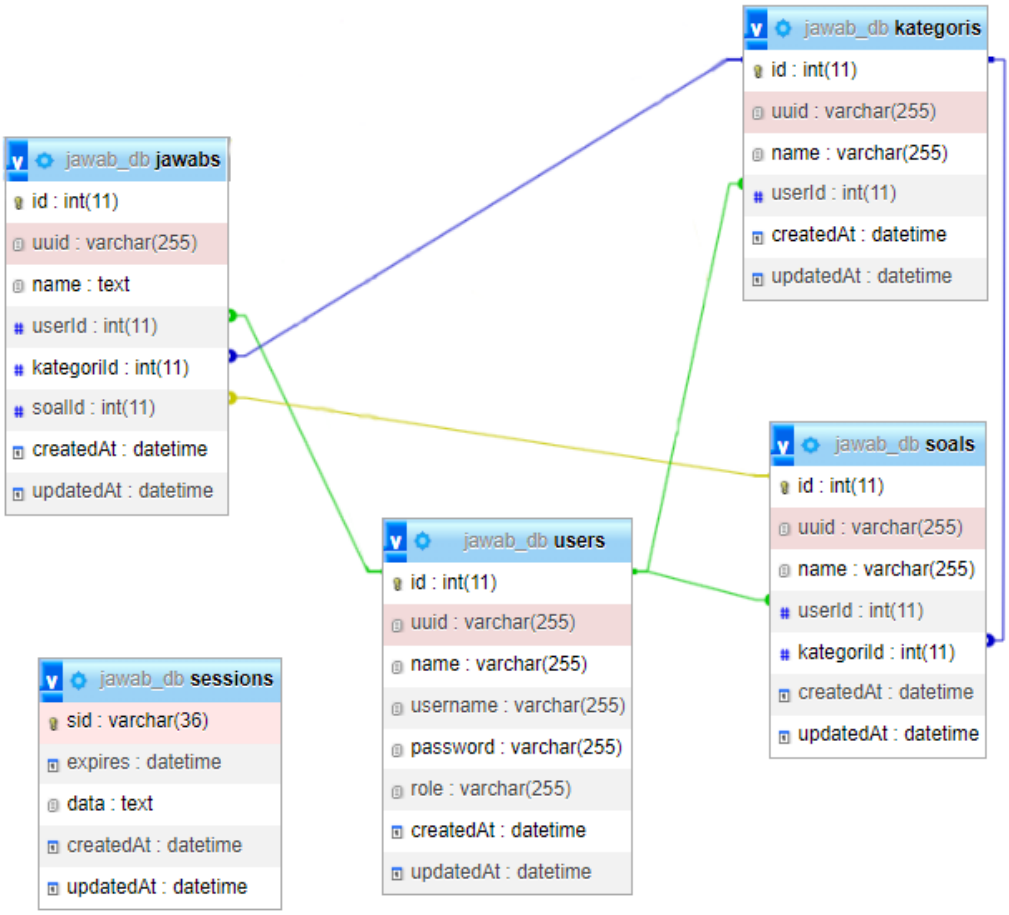
sessions	
Kolom	Tipe Data
sid	*varchar*
expires	*datetime*
data	*text*
createdAt	*datetime*
uddatedAt	*datetime*

```
export const Me = async (req, res) =>{
    if(!req.session.userId){
        return res.status(401).json({msg: "Mohon login ke akun Anda!"});
    }
    const user = await User.findOne({
        attributes:['uuid','name','username','role'],
        where: {
            uuid: req.session.userId
        }
    });
    if(!user) return res.status(404).json({msg: "User tidak ditemukan"});
    res.status(200).json(user);
}
```

Pada potongan kode diatas dapat dilihat proses penggunaan tabel sessions pada fungsi *Me* yang berguna untuk mengambil data *user* yang sedang aktif atau *login* yang ditandai dengan *User.findOne.*

6. Relasi Tabel

Pada relasi tabel ditunjukkan implementasi dari relasi antar tabel yang berguna untuk mengolah data dari satu tabel atau lebih. Relasi yang diimplementasikan adalah relasi berjenis *one to many* dimana diantaranya adalah relasi tabel users dengan tabel kategoris, soals, dan jawabs, relasi tabel kategoris dengan tabel soals dan tabel jawabs, dan relasi tabel soals dengan tabel jawabs. Implementasi dari relasi tabel dapat dilihat pada gambar 5.1.



Gambar 5. 1. Implementasi Relasi Tabel

```
#kategori
Users.hasMany(Kategoris);
Kategoris.belongsTo(Users, {foreignKey: 'userId'});

#soal
Users.hasMany(Soals);
Soals.belongsTo(Users, {foreignKey: 'userId'});
Kategoris.hasMany(Soals);
Soals.belongsTo(Kategoris, {foreignKey: 'kategoriId'});
```

```
#jawab
Users.hasMany(Jawabs);
Jawabs.belongsTo(Users, {foreignKey: 'userId'});
Kategoris.hasMany(Jawabs);
Jawabs.belongsTo(Kategoris, {foreignKey: 'kategoriId'});
Soals.hasMany(Jawabs);
Jawabs.belongsTo(Soals, {foreignKey: 'soalId'});
```

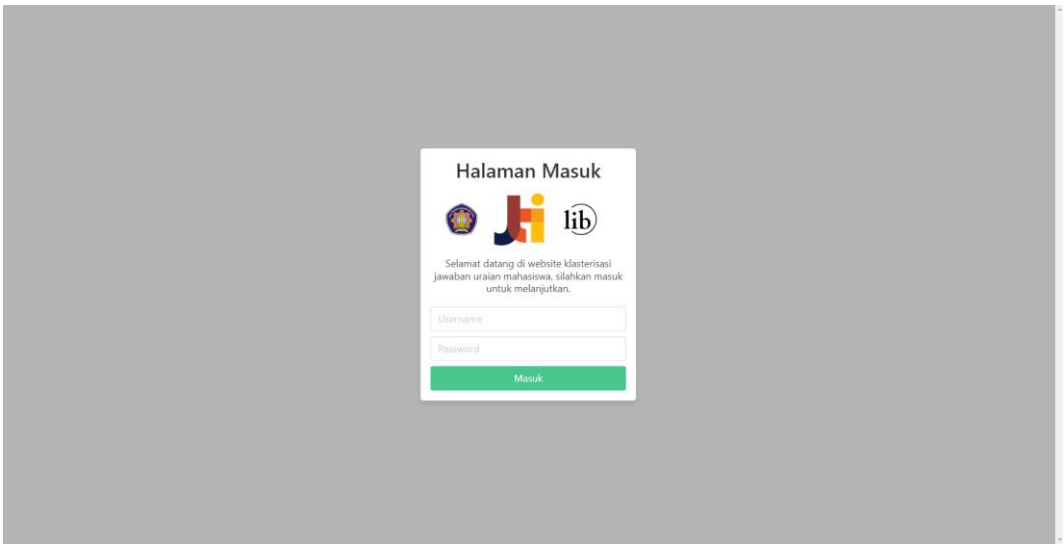
Pada potongan kode diatas dapat dilihat proses pembuatan relasi *one to many* antar tabel pada *database* yang ditandai dengan kode *hasMany* dan *belongsTo*.

5.3.2. Implementasi Antarmuka

Pada implementasi antarmuka akan dipaparkan hasil pembuatan antarmuka sistem berupa desain atau halaman sistem. Proses ini menggunakan bahasa pemrograman javascript, html, css dan menggunakan framework reactjs selain itu untuk template awal menggunakan redux dan react-router-dom, Setidaknya ada 18 halaman yang dibagi oleh 3 user, yaitu admin, dosen, dan mahasiswa.

1. Login (Admin, Dosen, Mahasiswa)

   Halaman ini berfungsi untuk masuknya pengguna kedalam sistem, dimana ada 3 jenis pengguna yaitu admin, dosen, dan mahasiswa. Untuk gambaran implementasi halaman login dapat dilihat pada gambar 5.2.



Gambar 5. 2. Halaman Antarmuka Login

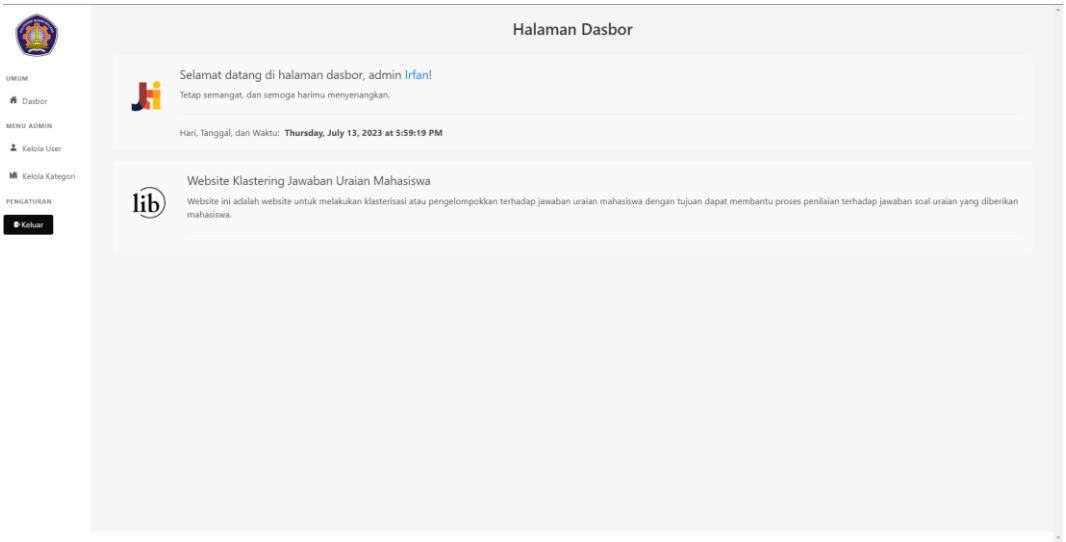
```
export const Login = async (req, res) =>{
  const user = await User.findOne({
```

```
            where: {
                username: req.body.username
            }
        });
        if(!user) return res.status(404).json({msg: "User tidak
    ditemukan"});
        const match = await argon2.verify(user.password,
    req.body.password);
        if(!match) return res.status(400).json({msg: "Password
    Salah"});
        req.session.userId = user.uuid;
        const uuid = user.uuid;
        const name = user.name;
        const username = user.username;
        const role = user.role;
        res.status(200).json({uuid, name, username, role});
}
```

Pada potongan kode diatas daapt dilihat user bisa dibedakan role nya karena ada kode *const role = user.role.*

2. Dasbor (Admin)

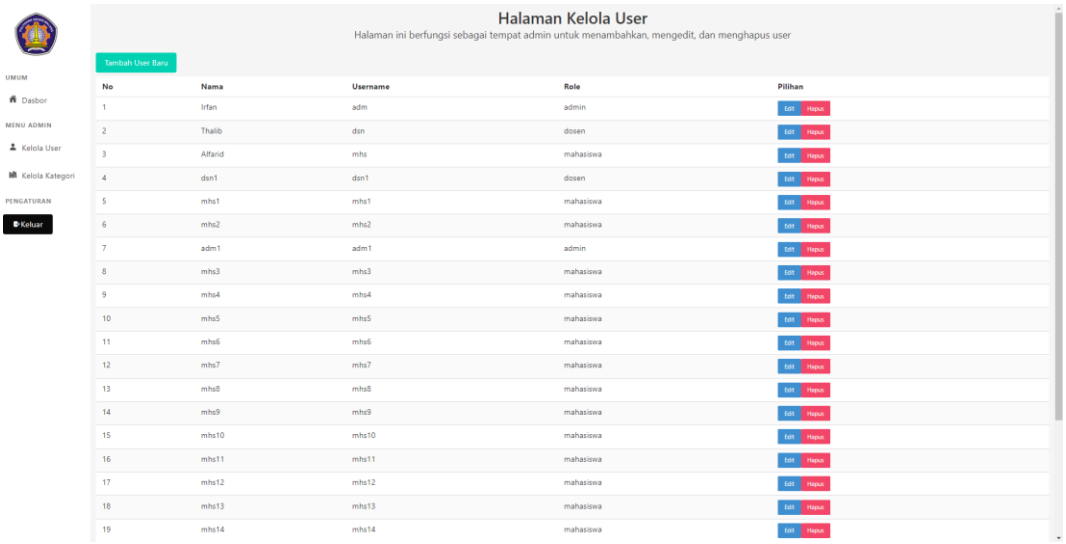
Halaman ini berfungsi untuk *landing page* sistem atau *home screen* dimana setelah admin masuk akan diarahkan ke halaman ini. Untuk gambaran implementasi halaman dasbor admin dapat dilihat pada gambar 5.3.



Gambar 5. 3. Halaman Antarmuka Dasbor Admin

3. Daftar User (Admin)

Halaman ini berfungsi sebagai tempat admin melihat daftar user dan mengelola user. Untuk gambaran implementasi halaman daftar user dapat dilihat pada gambar 5.4.



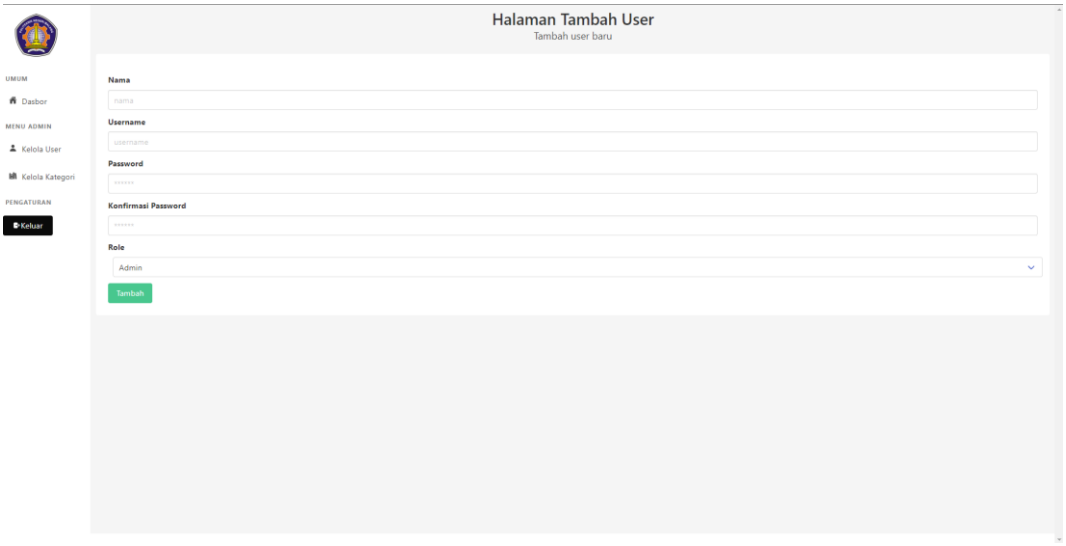
Gambar 5. 4. Halaman Antarmuka Daftar User

```
        export const getUsers = async(req, res) =>{
            try {
                const response = await User.findAll({
                    attributes:['uuid','name','username','role']
                });
                res.status(200).json(response);
            } catch (error) {
                res.status(500).json({msg: error.message});
            }
}
```

Pada potongan kode diatas dapat dilihat bahwa admin melihat dafar user yang ditunjukkan oleh kode *getUsers*.

4. Form Tambah User Baru (Admin)

Halaman ini berfungsi sebagai tempat admin menambahkan user baru. Untuk gambaran implementasi halaman form tambah user baru dapat dilihat pada gambar 5.5.



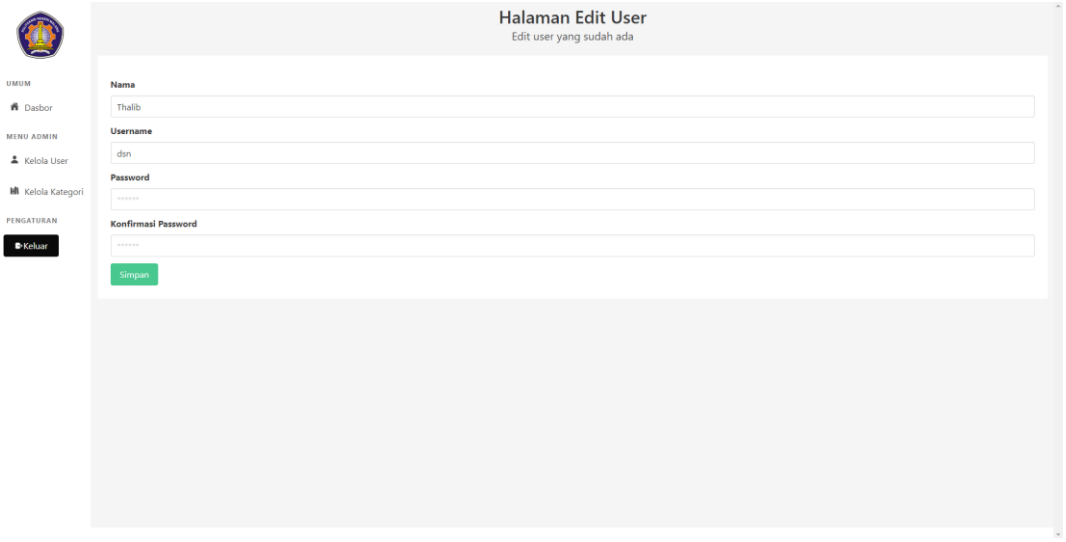
Gambar 5. 5. Halaman Antarmuka Form Tambah User Baru

```
export const createUser = async(req, res) =>{
    const {name, username, password, confPassword, role} =
req.body;
    if(password !== confPassword) return
res.status(400).json({msg: "Password dan Confirm Password
tidak sama"});
    const hashPassword = await argon2.hash(password);
    try {
        await User.create({
            name: name,
            username: username,
            password: hashPassword,
            role: role
        });
        res.status(201).json({msg: "Register Berhasil"});
    } catch (error) {
        res.status(400).json({msg: error.message});
    }
}
```

Potongan kode diatas menunjukkan bahwa admin dapat menambahkan data user baru yang ditandai dengan kode *createUser*.

5. Form Edit User (Admin)

Halaman ini berfungsi sebagai tempat admin mengedit user yang sudah dimasukkan sebelumnya. Untuk gambaran implementasi halaman form edit user dapat dilihat pada gambar 5.6.



Gambar 5. 6. Halaman Antarmuka Form Edit User

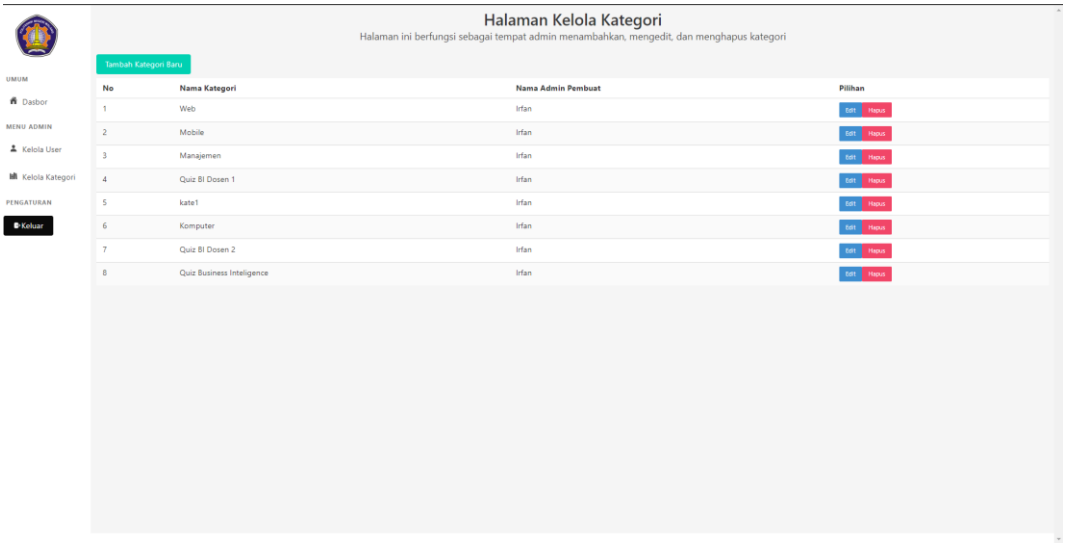
```
export const updateUser = async(req, res) =>{
    const user = await User.findOne({
        where: {
            uuid: req.params.id
        }
    });
    if(!user) return res.status(404).json({msg: "User tidak
ditemukan"});
    const {name, username, password, confPassword, role} =
req.body;
    let hashPassword;
    if(password === "" || password === null){
        hashPassword = user.password
    }else{
        hashPassword = await argon2.hash(password);
    }
    if(password !== confPassword) return
res.status(400).json({msg: "Password dan Confirm Password
tidak sama"});
    try {
        await User.update({
            name: name,
            username: username,
            password: hashPassword,
            role: role
        },{
            where:{
                id: user.id
            }
        });
        res.status(200).json({msg: "User di Update"});
```

```
        } catch (error) {
            res.status(400).json({msg: error.message});
        }
}
```

Potongan kode diatas menunjukkan bahwa admin dapat mengedit data user yang ditandai dengan kode *updateUser*.

6. Daftar Kategori (Admin)

Halaman ini berfungsi sebagai tempat admin melihat daftar kategori dan mengelola kategori. Untuk gambaran implementasi halaman daftar kategori dapat dilihat pada gambar 5.7.



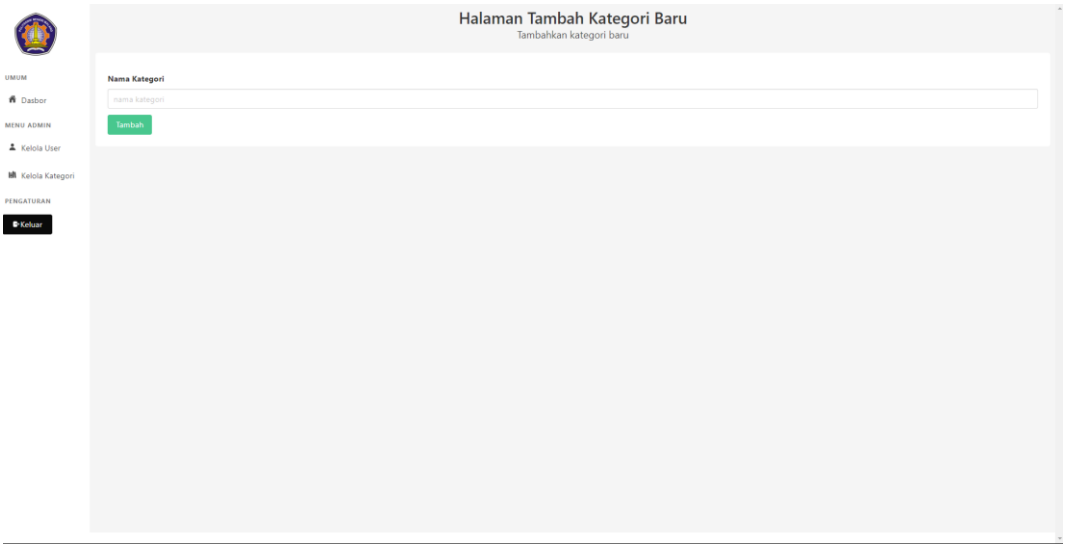
Gambar 5. 7. Halaman Antarmuka Daftar Kategori

```
export const getKategoris = async(req, res) =>{
    try {
        const response = await Kategori.findAll({
            attributes:['uuid','name'],
            include:[{
                model: User,
                attributes:['name']
            }]
        });
        res.status(200).json(response);
    } catch (error) {
        res.status(500).json({msg: error.message});
    }
}
```

Pada potongan kode diatas dapat dilihat bahwa admin melihat dafar kategori yang ditunjukkan oleh kode *getKategoris.*

7. Form Tambah Kategori Baru (Admin)

Halaman ini berfungsi sebagai tempat admin menambahkan kategori baru. Untuk gambaran implementasi halaman tambah kategori baru dapat dilihat pada gambar 5.8.



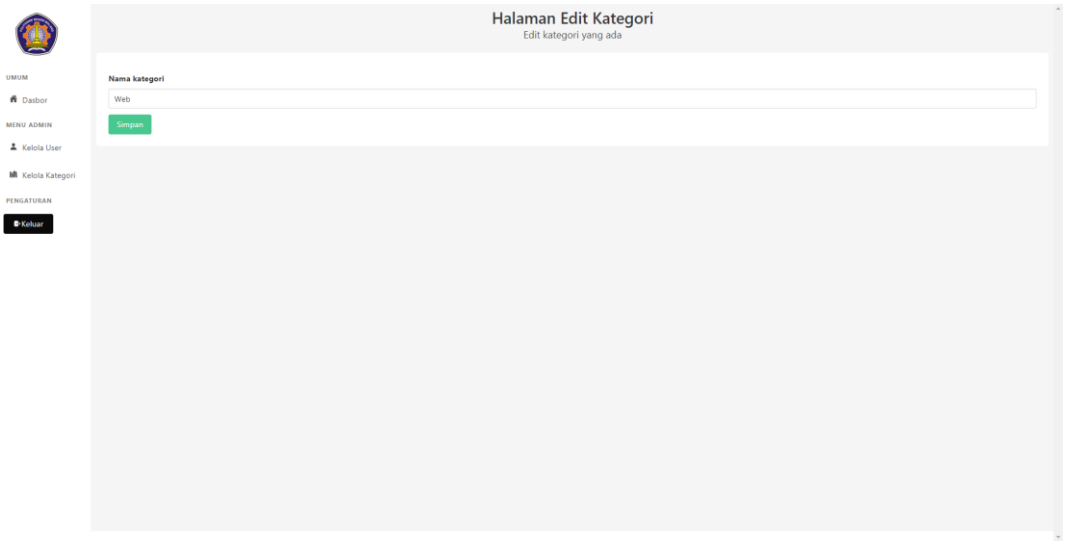
Gambar 5. 8. Halaman Antarmuka Tambah Kategori Baru

```
export const createKategori = async(req, res) =>{
    const {name} = req.body;
    try {
        await Kategori.create({
            name: name,
            userId: req.userId
        });
        res.status(201).json({msg: "Kategori Berhasil dibuat"});
    } catch (error) {
        res.status(500).json({msg: error.message});
    }
}
```

Potongan kode diatas menunjukkan bahwa admin dapat manambahjkan data kategori baru yang ditandai dengan kode *createKategori.*

8. Form Edit Kategori (Admin)

Halaman ini berfungsi sebagai tempat admin mengedit kategori yang sudah dimasukkan sebelumnya. Untuk gambaran implementasi halaman form edit kategori dapat dilihat pada gambar 5.9.



Gambar 5. 9. Halaman Antarmuka Edit Kategori

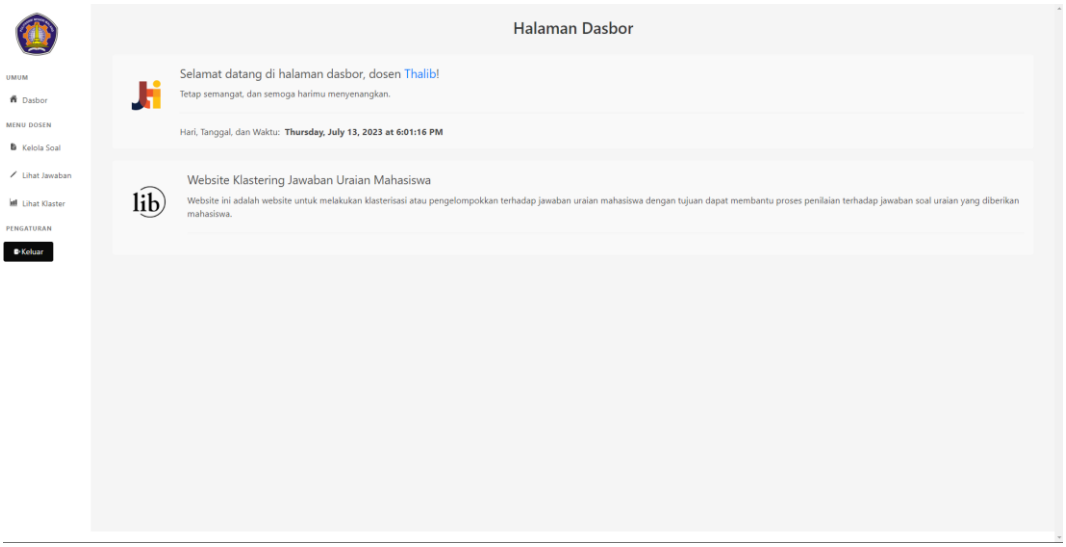
```
export const updateKategori = async(req, res) =>{
    try {
        const kategori = await Kategori.findOne({
            where:{
                uuid: req.params.id
            }
        });
        if(!kategori) return res.status(404).json({msg: "Data tidak
ditemukan"});
        const {name} = req.body;

            await Kategori.update({
                name
            },{
                where:{
                    id: kategori.id
                }
            });
        res.status(200).json({msg: "Kategori berhasil di update"});
    } catch (error) {
        res.status(500).json({msg: error.message});
    }
}
```

Potongan kode diatas menunjukkan bahwa admin dapat mengedit data kategori yang ditandai dengan kode *updateKategori*.

9. Dasbor (Dosen)

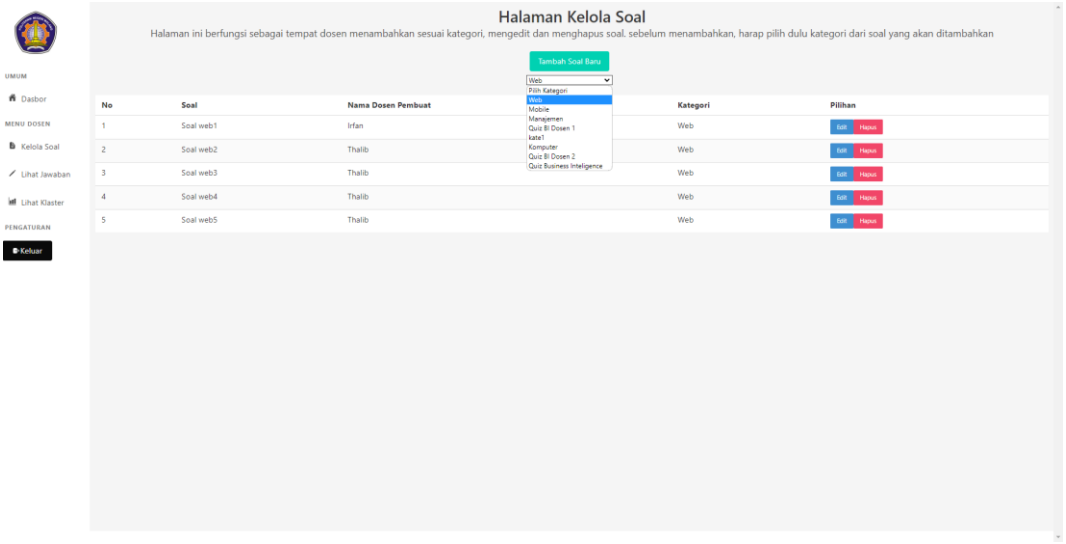
Halaman ini berfungsi untuk *landing page* sistem atau *home screen* dimana setelah dosen masuk akan diarahkan ke halaman ini. Untuk gambaran implementasi halaman dasbor dosen dapat dilihat pada gambar 5.10.



Gambar 5. 10. Halaman Antarmuka Dasbor Dosen

10. Daftar Soal (Dosen)

Halaman ini berfungsi sebagai tempat dosen melihat daftar soal dan mengelola soal. Untuk gambaran implementasi halaman daftar soal dapat dilihat pada gambar 5.11.



Gambar 5. 11. Halaman Antarmuka Daftar Soal

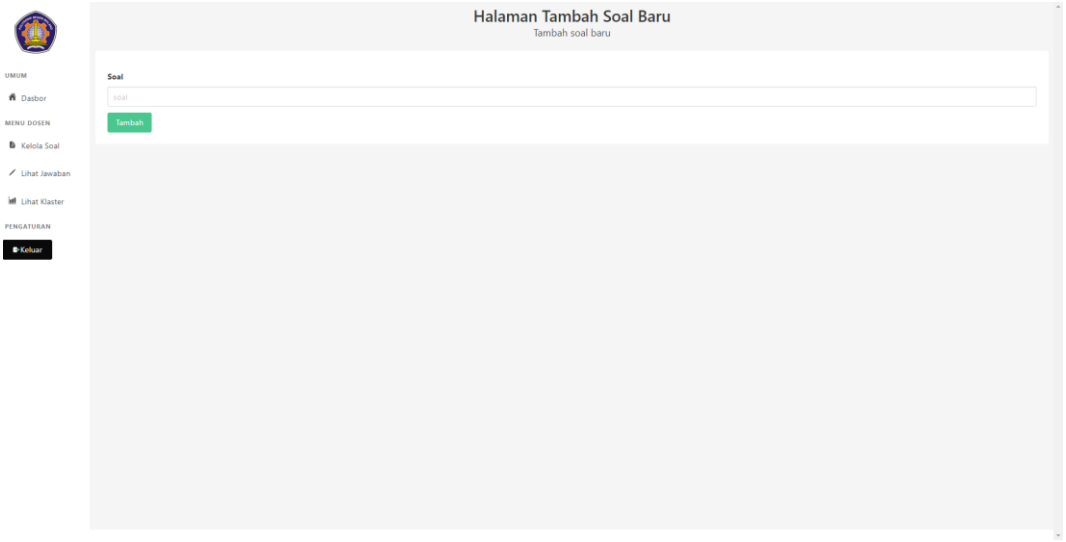
```
export const getSoals = async(req, res) =>{
    try {
        const response = await Soal.findAll({
            attributes:['uuid','name'],
            include:[{
                model: User,
                attributes: ['name']
            },{
                model: Kategori,
                attributes:['name']
            }

        ]
        });
        res.status(200).json(response);
    } catch (error) {
        res.status(500).json({msg: error.message});
    }
}
```

Pada potongan kode diatas dapat dilihat bahwa admin melihat dafar user yang ditunjukkan oleh kode *getSoals.*

11. Form Tambah Soal Baru (Dosen)

Halaman ini berfungsi sebagai tempat dosen menambahkan soal baru. Untuk gambaran implementasi halaman form tambah soal baru dapat dilihat pada gambar 5.12.



Gambar 5. 12. Halaman Antarmuka Tambah Soal Baru

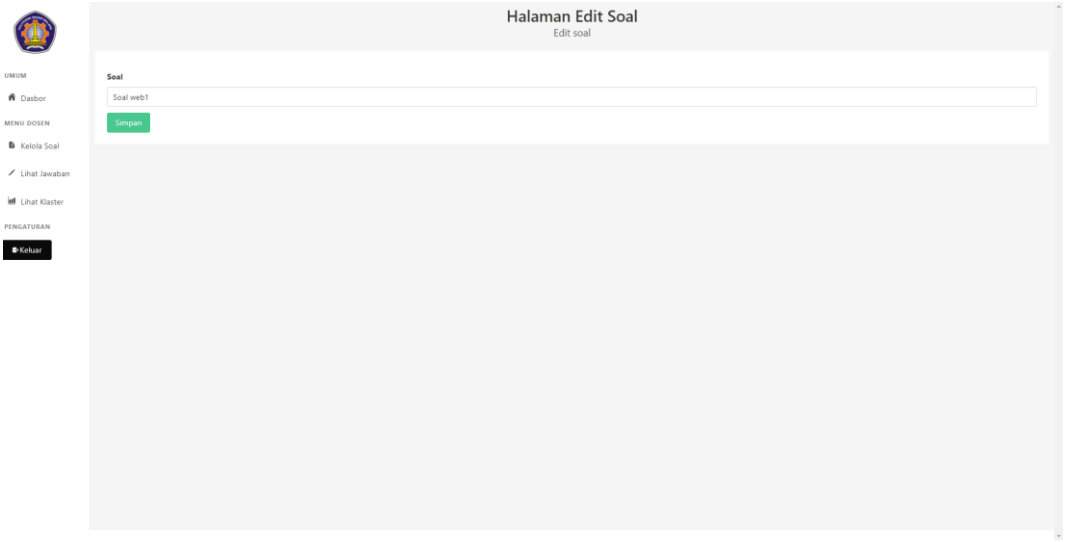
```
export const createSoal = async(req, res) =>{
    try {
        const kategori = await Kategori.findOne({
            where:{
                name: req.params.kategoriName
            }
        });
        if(!kategori) return res.status(404).json({msg: "Data tidak
ditemukan"});
        const {name} = req.body;
        await Soal.create({
            name: name,
            kategoriId: kategori.id,
            userId: req.userId

        });
        res.status(201).json({msg: "Soal Berhasil dibuat"});
    } catch (error) {
        res.status(500).json({msg: error.message});
    }
}
```

Potongan kode diatas menunjukkan bahwa dosen dapat manambahjkan data soal baru yang ditandai dengan kode *createSoal.*

12. Form Edit Soal (Dosen)

Halaman ini berfungsi sebagai tempat dosen mengedit soal yang sudah dimasukkan sebelumnya. Untuk gambaran implementasi halaman form edit soal dilihat pada gambar 5.13.



Gambar 5. 13. Halaman Antarmuka Form Edit Soal

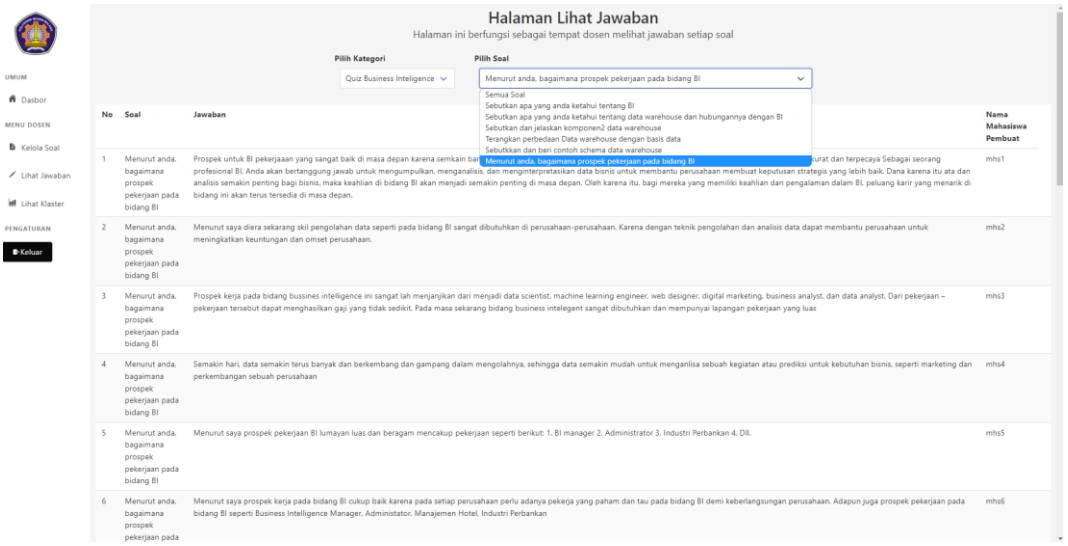
```
export const updateSoal = async(req, res) =>{
    try {
        const soal = await Soal.findOne({
            where:{
                uuid: req.params.id
            }
        });
        if(!soal) return res.status(404).json({msg: "Data tidak
ditemukan"});
        const {name} = req.body;

            await Soal.update({
                name
            },{
                where:{
                    id: soal.id
                }
            });
        res.status(200).json({msg: "Soal berhasil di update"});
    } catch (error) {
        res.status(500).json({msg: error.message});
    }
}
```

Potongan kode diatas menunjukkan bahwa dosen dapat mengedit data soal yang ditandai dengan kode *updateSoal.*

13. Daftar Jawaban (Dosen)

Halaman ini berfungsi sebagai tempat dosen melihat daftar jawaban yang dimasukkan mahasiswa. Untuk gambaran implementasi halaman daftar jawaban dapat dilihat pada gambar 5.14.



Gambar 5. 14. Halaman Antarmuka Daftar Jawaban

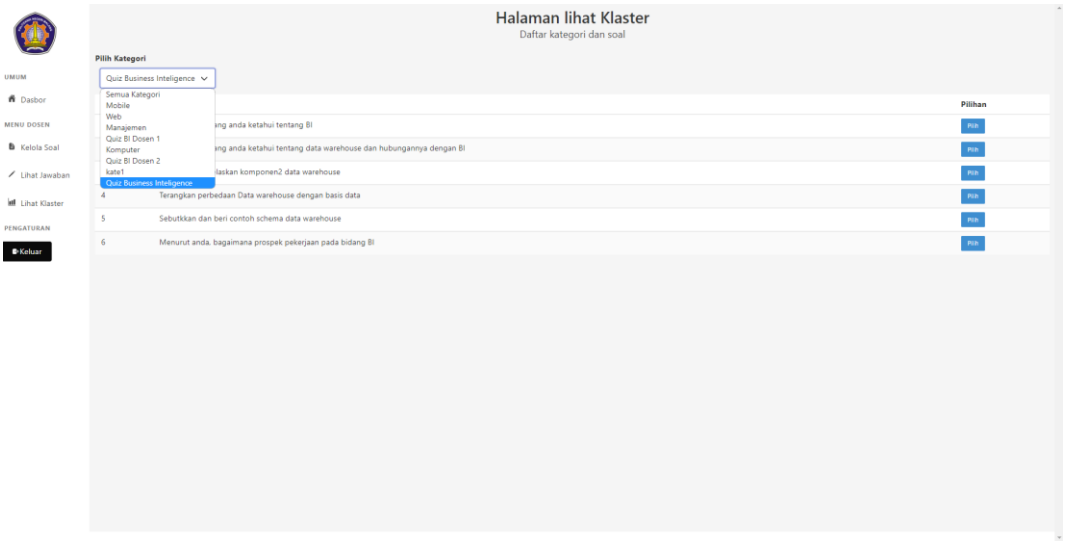
```
export const getJawabBySoal = async(req, res) =>{
    try {
        const soal = await Soal.findOne({
            where:{
                uuid: req.params.id
            }
        });
        if(!soal) return res.status(404).json({msg: "Data soal tidak
ditemukan"});
        const response = await Jawab.findAll({
                attributes:['uuid','name'],
                include:[{
                    model: Soal,
                    attributes:['name']
                },
                {
                    model: User,
                    attributes: ['name']
                },
            ],
            where:{
                soalId: soal.id
            },
            });
        res.status(200).json(response);
    } catch (error) {
        res.status(500).json({msg: error.message});
```

```
        }
}
```

Pada potongan kode diatas dapat dilihat bahwa dosen memilih soal untuk dilihat jawabannya ditandai dengan kode *getJawabBySoal.*

14. Pilih Soal (Dosen)

Halaman ini berfungsi sebagai tempat dosen memilih soal yang jawabannya akan di klaster. Untuk gambaran implementasi halaman pilih soal dilihat pada gambar 5.15.



Gambar 5. 15. Halaman Antarmuka Pilih Soal

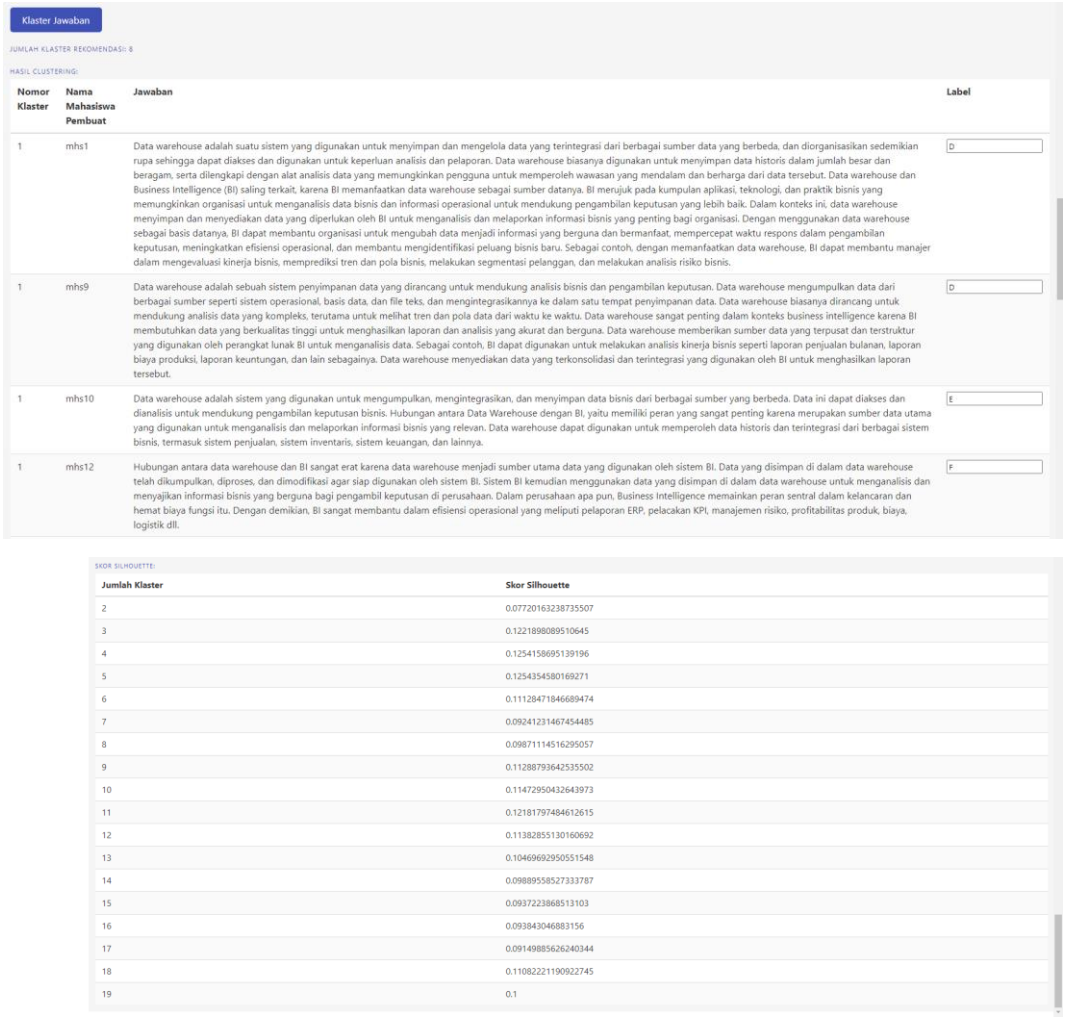
```
export const getSoalByKategori = async(req, res) =>{
    try {
        const kategori = await Kategori.findOne({
            where:{
                uuid: req.params.id
            }
        });
        if(!kategori) return res.status(404).json({msg: "Data
kategori tidak ditemukan"});
        const response = await Soal.findAll({
                attributes:['uuid','name'],
                include:[{
                    model: Kategori,
                    attributes:['name']
                },
                {
                    model: User,
                    attributes: ['name']
                },
```

```
                    ],
                    where:{
                        kategoriId: kategori.id
                    },
                    });
                res.status(200).json(response);
            } catch (error) {
                res.status(500).json({msg: error.message});
            }
}
```

Pada potongan kode diatas dapat dilihat bahwa dosen memilih soal berdasarkan kategori ditandai dengan kode *getSoalByKategori.*

15. Hasil Klaster (Dosen)

Halaman ini berfungsi sebagai tempat dosen melihat hasil klaster jawaban rekomendasi sistem dari soal yang dipilih. Untuk gambaran implementasi halaman lihat klaster dapat dilihat pada gambar 5.16.



Gambar 5. 16. Halaman Antarmuka Lihat Klaster

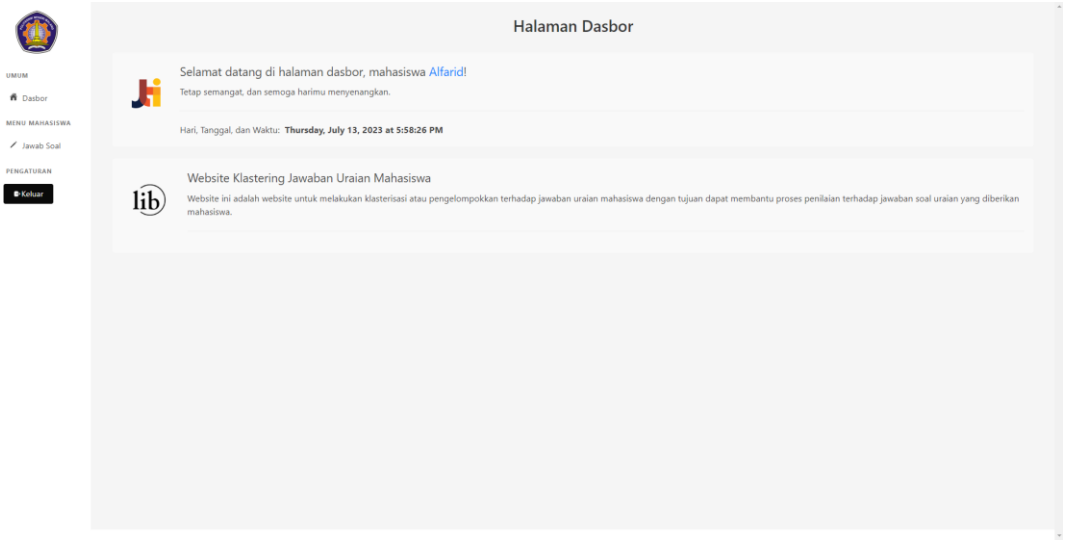
```
{BestCluster && BestCluster.length > 0 && (
 <div>
  <h2 className="heading">Hasil Clustering:</h2>
  <table className="table is-striped is-fullwidth">
   <thead>
    <tr>
     <th>Nomor Klaster</th>
     <th>Nama Mahasiswa Pembuat</th>
     <th>Jawaban</th>
    </tr>
   </thead>
   <tbody>
   {BestCluster
     .map((clusterLabel, index) => ({
      clusterLabel,
      jawaban: jawabans[index].name,
      mahasiswa: jawabans[index].user &&
jawabans[index].user.name,
```

```
                  }))
                  .sort((a, b) => a.clusterLabel - b.clusterLabel)
                  .map((item, index) => (
                    <tr key={index}>
                      <td>{item.clusterLabel}</td>
                      <td>{item.mahasiswa}</td>
                      <td>{item.jawaban}</td>
                    </tr>
                  ))}
                </tbody>
              </table>
            </div>
)}
          {silhouetteScores && silhouetteScores.length > 0 && (
            <div>
              <h2 className="heading">Skor Silhouette:</h2>
              <table className="table is-striped is-fullwidth">
                <thead>
                  <tr>
                    <th>Jumlah Klaster</th>
                    <th>Skor Silhouette</th>
                  </tr>
                </thead>
                <tbody>
                  {silhouetteScores.map((score, index) => (
                    <tr key={index}>
                      <td>{index + 2}</td>
                      <td>{score}</td>
                    </tr>
                  ))}
                </tbody>
              </table>
            </div>
)}
```

Pada potongan kode diatas dapat dilihat bahwa proses *clustering* menghasilkan data klaster terbaik yang ditandai dengan *BestCluster*, dan data nilai *silhouette* yang ditandai pada kode *silhouetteScores*.

16. Dasbor (Mahasiswa)

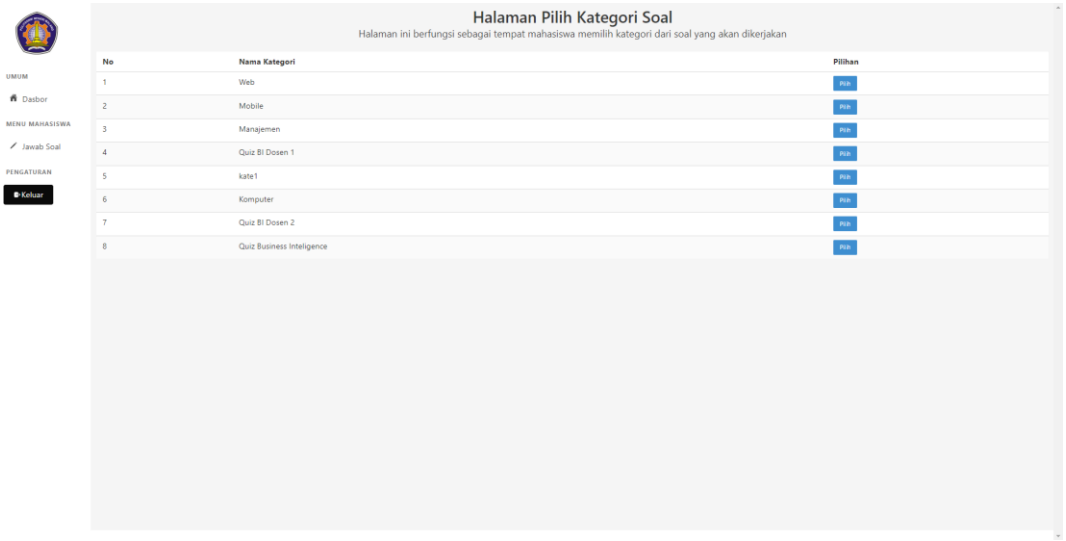
Halaman ini berfungsi untuk *landing page* sistem atau *home screen* dimana setelah mahasiswa masuk akan diarahkan ke halaman ini. Untuk gambaran implementasi halaman dasbor mahasiswa dapat dilihat pada gambar 5.17.



Gambar 5. 17. Halaman Antarmuka Dasbor Mahasiswa

17. Pilih Kategori Soal (Mahasiswa)

Halaman ini berfungsi sebagai tempat mahasiswa memilih kategori soal yang akan dijawab. Untuk gambaran implementasi halaman pilih kategori soal dapat dilihat pada gambar 5.18.



Gambar 5. 18. Halaman Antarmuka Pilih Kategori Soal

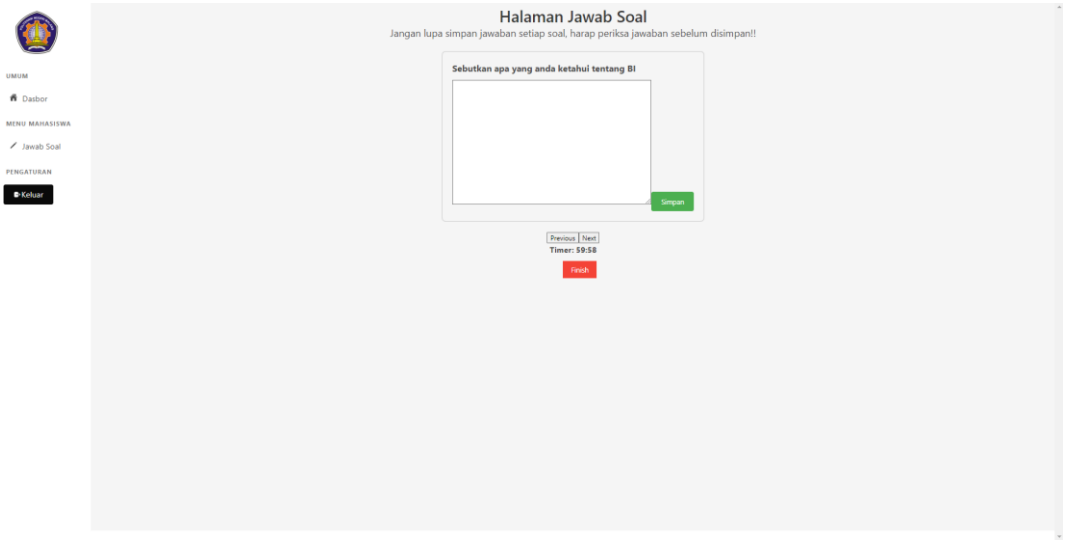
```
export const getSoalByKategori = async(req, res) =>{
  try {
    const kategori = await Kategori.findOne({
      where:{
        uuid: req.params.id
      }
    });
```

```
            if(!kategori) return res.status(404).json({msg: "Data
        kategori tidak ditemukan"});
            const response = await Soal.findAll({
                attributes:['uuid','name'],
                include:[{
                    model: Kategori,
                    attributes:['name']
                },
                {
                    model: User,
                    attributes: ['name']
                },
                ],
                where:{
                    kategoriId: kategori.id
                },
                });
            res.status(200).json(response);
        } catch (error) {
            res.status(500).json({msg: error.message});
        }
}
```

Pada potongan kode diatas dapat dilihat bahwa mahasiswa memilih soal berdasarkan kategori ditandai dengan kode *getSoalByKategori*.

18. Jawab Soal (Mahasiswa)

Halaman ini berfungsi sebagai tempat mahasiswa memasukkan jawaban dari soal yang dipilih.



Gambar 5. 19. Halaman Antarmuka Jawab Soal

```
export const createJawab = async (req, res) => {
   try {
     const soal = await Soal.findOne({
       where: {
         uuid: req.params.id,
       },
     });
     if (!soal) return res.status(404).json({ msg: "Data soal tidak
ditemukan" });
     await Jawab.create({
       name: req.body.name,
       soalId: soal.id,
       kategoriId: soal.kategoriId,
       userId: req.userId,
     });
     res.status(201).json({ msg: "Jawab Berhasil dibuat" });
   } catch (error) {
     res.status(500).json({ msg: error.message });
   }
};
```

Pada potongan kode diatas dapat dilihat bahwa mahasiswa memasukkan datanya jawaban yang ditandai dengan *Jawab.create* namun sebelumnya diambil dulu data soal dari jawaban yang dipilih ditandai dengan *Soal.findOne*.

5.4. Pengujian

Implementasi pengujian akan terdiri dari tiga jenis pengujian terhadap klaster menggunakan, pengujian terhadap sistem, pengujian efektifitas.

5.4.1. Pengujian Klaster

Pengujian Klaster dilakukan menggunakan nilai *silhouette coefficient* dimana nilai ini akan dibandingkan dan nilai terbesar akan menjadi indikator bahwa jumlah klaster dengan nilai *silhouette* terbesar adalah jumlah klaster yang direkomendasikan sistem. Karena semakin besar nilai *silhouette* nya maka semakin bagus persebaran datanya.

```
length_cluster = len(jawabans)
max_clusters = length_cluster - 1
def perform_kmeans_clustering(tfidf_matrix, max_clusters):

    min_clusters = 2
```

```
    silhouette_scores = []

    for num_clusters in range(min_clusters, max_clusters + 1):
    # KMeans clustering
        kmeans = KMeans(n_clusters=num_clusters, random_state=42)
        kmeans.fit(tfidf_matrix)
        cluster_labels = kmeans.labels_ + 1
        print("Number of Clusters:", num_clusters)
        print("Cluster Labels:", cluster_labels)

        # Silhouette score
        silhouette_avg = silhouette_score(tfidf_matrix, cluster_labels)
        silhouette_scores.append(silhouette_avg)
        print("sil:", silhouette_avg)

    # Find the index of the highest silhouette score
    max_silhouette_index = silhouette_scores.index(max(silhouette_scores))

    # Get the number of clusters corresponding to the highest silhouette score
    best_num_clusters = max_silhouette_index + min_clusters

    # Perform KMeans clustering with the best number of clusters
    best_kmeans = KMeans(n_clusters=best_num_clusters, random_state=42)
    best_kmeans.fit(tfidf_matrix)
    best_cluster_labels = best_kmeans.labels_ + 1
    print("Best Number of Clusters:", best_num_clusters)
    print("BEst Clusters label:", best_cluster_labels)

    return best_cluster_labels.tolist(), silhouette_scores, best_num_clusters
```

Pada potongan kode diatas dapat kita lihat bahwa perhitungan *silhouette* yang ditandai dengan # *Silhouette score* ada didalam perulangan klaster, hal ini menunjukkan bahwa setiap dilakukan proses *clustering* maka akan dilakukan juga proses perhitungan *silhouette*. Setelah semua nilai *silhouette* terhadap setiap jumlah klaster didapatkan, akan dihitung nilai terbesar ada pada jumlah klaster berapa yang akhirnya jumlah klaster itulah yang akan menjadi klaster rekomendasi sistem yang ditandai dengan *best_num_clusters*. Selain itu dengan nilai *silhouette* kita dapat mengetahui seberapa bagus persebaran data dan seberapa cocok data yang diolah pada algoritma.

5.4.2. Pengujian Sistem

Pengujian Sistem akan dilakukan menggunakan *use case scenario*. Pengujian ini akan dilakukan oleh rekan peneliti yang tidak ikut serta dalam

pembuatan sistem. Disini penguji diharapkan dapat menguji sistem sesuai dengan tabel yang sudah disediakan, menambahkan catatan bila perlu atau ada proses yang gagal, dan menyatakan sukses ata gagalnya proses tersebut dengan cara menambahkan warna kuning pada tempat yang sudah disediakan. Pengujian dilakukan pada tangga 17 Juli 2023. Untuk use case diagram dapat dilihat pada gambar 5.20.

Gambar 5. 20. Use Case Diagram

Pada use case dapat dilihat ada tiga role dan enam proses yang dapat dilakukan sistem. Terdapat role admin yang dapat login, mengelola user, dan mengelola kategori soal. Role dosen yang dapat login, mengelola soal, dan melihat hasil klaster, dan Role mahasiswa yang dapat login dan menjawab soal. Untuk skenario beserta tabel pengujian sistem dapat dilihat pada tabel 5.11 Sampai 5.17.

1. Login

Tabel 5. 11. Pengujian Proses Login

Aktor: Admin, Dosen, Mahasiswa		
Deskripsi: User masuk kedalam sistem		
Pra-Kondisi: User mengetahui keperluan login (username & password)		
Proses Utama	Respon	
1. Akses http://libclusterjum.my.id/	Menampilkan halaman login	
2. Memasukkan data username dan password yang sesuai lalu klik tombol masuk	Menampilkan dasbor	
Proses Alternatif 1	Respon	
2. Memasukkan data username dan password tidak sesuai lalu klik tombol masuk	Menampilkan peringatan user tidak ditemukan	
Proses Alternatif 2	Respon	
2. Memasukkan data username tidak sesuai, dan password sesuai lalu klik tombol masuk	Menampilkan peringatan user tidak ditemukan	
Proses Alternatif 3	Respon	
2. Memasukkan data username sesuai, dan password tidak sesuai lalu klik tombol masuk	Menampilkan peringatan password salah	
Catatan	Pada Proses ke	
Hasil		
Gagal Mayor	Gagal Minor	Sukses

2. Kelola User

Tabel 5. 12. Pengujian Proses Kelola User

Aktor: Admin
Deskripsi: Admin menambahkan, mengedit, dan menghapus data user

Pra-Kondisi: Login kedalam sistem sebagai admin, terdapat data user pada database	
Proses Utama	Respon
1. Akses dasbor admin lalu klik menu kelola user pada sidebar	Menampilkan halaman kelola user
2. Klik tombol tambah user baru	Menampilkan halaman tambah user
3. Memasukkan data dengan sesuai dan memilih role lalu klik tombol tambah	Menampilkan peringatan user berhasil dibuat dan data baru muncul pada tabel
4. Kembali ke menu kelola user, pilih salah satu user lalu klik tombol edit	Menampilkan halaman edit user
5. Memasukkan data dengan sesuai lalu klik tombol edit	Menampilkan peringatan user berhasil diedit dan data muncul pada tabel
6. Kembali ke menu kelola user, pilih salah satu user lalu klik tombol delete	Data yang dihapus hilang
Proses Alternatif 1	Respon
3. Memasukkan data dengan sesuai dan tidak memilih role lalu klik tombol tambah	Menampilkan peringatan proses gagal
Proses Alternatif 2	Respon
3. Memasukkan data password dan confirm password tidak sesuai dan memilih role lalu klik tombol tambah	Menampilkan peringatan data password dan confirm password tidak sama
Proses Alternatif 3	Respon
3. Memasukkan data password dan confirm password tidak sesuai dan tidak memilih role lalu klik tombol tambah	Menampilkan peringatan data password dan confirm password tidak sama

Proses Alternatif 4	Respon
3. Tidak mengisi data lalu klik tombol tambah	Menampilkan peringatan proses gagal
Proses Alternatif 5	Respon
5. Memasukkan data password dan confirm password tidak sesuai lalu klik tombol edit	Menampilkan peringatan data password dan confirm password tidak sama
Proses Alternatif 6	Respon
5. Tidak mengisi data lalu klik tombol simpan	Menampilkan peringatan proses gagal
Catatan	Pada Proses ke

Hasil		
Gagal Mayor	Gagal Minor	Sukses

3. Kelola Kategori

Tabel 5. 13. Pengujian Proses Kelola Kategori

Aktor: Admin	
Deskripsi: Admin menambahkan, mengedit, dan menghapus data kategori	
Pra-Kondisi: Login kedalam sistem sebagai admin, terdapat data user dan kategori pada database	
Proses Utama	Respon
1. Akses dasbor admin lalu klik menu kelola kategori pada sidebar	Menampilkan halaman kelola kategori
2. Klik tombol tambah kategori baru	Menampilkan halaman tambah kategori
3. Memasukkan data lalu klik tombol tambah	Menampilkan peringatan kategori berhasil dibuat dan data baru muncul pada tabel

4. Kembali ke menu kelola kategori, pilih salah satu kategori lalu klik tombol edit	Menampilkan halaman edit kategori
5. Memasukkan data lalu klik tombol simpan	Menampilkan peringatan kategori berhasil diedit dan data muncul pada tabel
6. Kembali ke menu kelola kategori, pilih salah satu kategori lalu klik tombol hapus	Data yang dihapus hilang
Proses Alternatif 1	Respon
3. Tidak mengisi data lalu klik tombol simpan	Menampilkan peringatan proses gagal
Proses Alternatif 2	Respon
5. Tidak mengisi data lalu klik tombol tambah	Menampilkan peringatan proses gagal
Catatan	Pada Proses ke

Hasil		
Gagal Mayor	Gagal Minor	Sukses

4. Kelola Soal

Tabel 5. 14. Pengujian Proses Kelola Soal

Aktor: Dosen	
Deskripsi: Dosen menambahkan, mengedit, dan menghapus data soal	
Pra-Kondisi: Login kedalam sistem sebagai dosen, terdapat data user, kategori, dan soal pada database	
Proses Utama	Respon
1. Akses dasbor dosen lalu klik menu kelola soal pada sidebar	Menampilkan halaman kelola soal
2. Pilih kategori lalu klik tombol tambah soal baru	Menampilkan halaman tambah kategori

3. Memasukkan data lalu klik tombol tambah	Menampilkan peringatan soal berhasil dibuat dan data baru muncul pada tabel
4. Kembali ke halaman kelola soal, pilih salah satu soal lalu klik tombol edit	Menampilkan halaman edit soal
5. Memasukkan data lalu klik tombol simpan	Menampilkan peringatan soal berhasil diedit dan data muncul pada tabel
6. Kembali ke halaman kelola soal, pilih salah satu soal lalu klik tombol hapus	Menampilkan peringatan data berhasil dihapus dan data yang dihapus hilang
Proses Alternatif 1	Respon
2. Tidak pilih kategori lalu klik tombol tambah soal baru	Menampilkan peringatan harap pilih kategori
Proses Alternatif 2	Respon
3. Tidak mengisi data lalu klik tombol tambah	Menampilkan peringatan proses gagal
Proses Alternatif 3	Respon
5. Tidak mengisi data lalu klik tombol simpan	Menampilkan peringatan proses gagal
Catatan	Pada Proses ke

Hasil		
Gagal Mayor	Gagal Minor	Sukses

5. Lihat Klaster

Tabel 5. 15. Pengujian Proses Lihat Klaster

Aktor: Dosen
Deskripsi: Dosen melihat data klaster

Pra-Kondisi: Login kedalam sistem sebagai dosen, terdapat data user, kategori, soal, dan jawaban pada database		
Proses Utama	Respon	
1. Akses dasbor dosen lalu klik menu lihat klaster pada sidebar	Menampilkan halaman lihat klaster	
2. Pilih kategori	Menampilkan data yang dipilih	
3. Pilih salah satu soal lalu klik tombol pilih	Menampilkan halaman klaster soal	
4. Klik tombol klaster jawaban	Menampilkan tabel klaster rekomendasi sistem	
Proses Alternatif 1	Respon	
3. Pilih salah satu soal lalu klik tombol pilih	Menampilkan peringatan tidak ada data	
Catatan	Pada Proses ke	
Hasil		
Gagal Mayor	Gagal Minor	Sukses

6. Jawab Soal

Tabel 5. 16. Pengujian Proses Jawab Soal

Aktor: Mahasiswa	
Deskripsi: Mahasiswa menambahkan data jawaban	
Pra-Kondisi: Login kedalam sistem sebagai mahasiswa, terdapat data user, kategori, soal, pada database	
Proses Utama	Respon
1. Akses dasbor mahasiswa lalu klik menu jawab soal pada sidebar	Menampilkan halaman pilih kategori soal
2. Pilih salah satu kategori lalu klik tombol pilih	Menampilkan halaman jawab soal

3. Mengisi data jawaban lalu klik tombol simpan	Menampilkan peringatan jawaban disimpan lalu data jawaban akan disimpan
4. Mengisi data jawaban lalu klik tombol next/previous	Menampilkan soal sebelum/sesudahnya dengan data jawaban ada pada textarea
5. Klik tombol next/previous di soal awal/akhir	Menampilkan peringatan and sudah di pertanyaan pertama/terakhir
6. Mengisi data jawaban lalu klik tombol simpan dan klik finish/waktu habis	Menyimpan data jawaban dan menampilkan halaman pilih kategori soal
Proses Alternatif 1	Respon
2. Pilih salah satu kategori tanpa soal lalu klik tombol pilih	Menampilkan halaman kosong
Proses Alternatif 2	Respon
3. Tidak mengisi data jawaban lalu klik tombol simpan	Menampilkan peringatan gagal menyimpan jawaban
Proses Alternatif 3	Respon
4. Tidak mengisi data jawaban lalu klik tombol next/previous	Menampilkan soal sebelum/sesudahnya dengan textarea kosong
Proses Alternatif 4	Respon
6. Mengisi data jawaban tidak klik tombol simpan dan klik finish/waktu habis	Tidak menyimpan data jawaban dan menampilkan halaman pilih kategori soal
Catatan	Pada Proses ke

Hasil		
Gagal Mayor	Gagal Minor	Sukses

7. Logout

Tabel 5. 17. Pengujian Proses Logout

Aktor: Admin, Dosen, Mahasiswa	
Deskripsi: User keluar dari sistem	
Pra-Kondisi: User sedang didalam sistem	
Proses Utama	Respon

1. Akses dasbor salah satu user lalu klik tombol keluar	Menampilkan halaman login	
Proses Alternatif 1	Respon	
1. Akses dasbor salah satu user lalu klik tombol keluar	Menampilkan halaman dasbor	
Catatan	Pada Proses ke	
Hasil		
Gagal Mayor	Gagal Minor	Sukses

5.4.3. Pengujian Efektifitas

Pengujian Efektifitas sistem akan dilakukan menggunakan perbandingan hasil pengelompokkan jawaban secara manual dan pengelompokkan jawaban menggunakan sistem yang dibuat, untuk pengelompokkan secara manual dilakukan oleh salah satu dosen Jurusan Teknologi Informasi, Politeknik Negeri Malang dimana pengujiannya sendiri dilakukan pada tanggal 17 Juli 2023, berikut adalah kemungkinan yang dapat timbul dari hasil pengujian efektifias.

1. Tidak efektif: Jika persentase < 25% atau perbandingan durasi antara proses manual dengan sistem adalah > 1/1
2. Kurang efektif: jika persentase > 25% dan < 50% atau perbandingan durasi antara proses manual dengan sistem adalah < 1/1 dan > 1/2
3. Cukup efektif: jika persentase > 50% dan < 75% atau perbandingan durasi antara proses manual dengan sistem adalah < 1/2
4. Sangat efektif: jika persentase > 75% dan perbandingan durasi antara proses manual dengan sistem adalah < 1/2

Untuk contoh tabel pengujian efektifitas sistem dapat dilihat pada tabel 5.18.

Tabel 5. 18. Pengujian Efektifitas

Klaster = 3 (dimisalkan)				
Soal	Jawaban	Kelompok/ Klaster (A-C)		Hasil (Sesuai/Tidak)
		Manual	Sistem	

Sebutkan definisi komputer.	Komputer adalah perangkat elektronik yang digunakan untuk mengolah data!	A/B/C	A/B/C	Sesuai/Tidak
	Komputer merupakan alat bantu yang memproses informasi dengan cepat.	A/B/C	A/B/C	Sesuai/Tidak
	Komputer dapat menjalankan program sesuai dengan instruksi yang diberikan.	A/B/C	A/B/C	Sesuai/Tidak
	Komputer memiliki kemampuan untuk menyimpan dan mengambil data.	A/B/C	A/B/C	Sesuai/Tidak
	Komputer terdiri dari beberapa komponen seperti CPU, RAM, dan hard disk!	A/B/C	A/B/C	Sesuai/Tidak
Durasi		menit/	menit	
Persentase				%