BAB V

IMPLEMENTASI PERANGKAT LUNAK

5.1 Implementasi Secara Umum

Visualisasi aplikasi untuk Sistem Informasi pencarian jalur terpendek berbasis SMS ini diimplementasikan dengan menggunakan bahasa pemrograman Java 2 SDK versi 1.4.2. dan sebagai *database*nya menggunakan MySQL. Tahap implementasi sistem merupakan tahap meletakkan sistem supaya siap untuk dioperasikan, termasuk kegiatan penulisan kode program atau skrip pemrograman yang digunakan.

5.2 Alasan Pemilihan Perangkat Lunak

Bahasa pemrograman yang digunakan adalah Java 2 SDK versi 1.4.2 dengan pertimbangan sebagai berikut :

1. Bersifat sederhana dan relatif mudah
2. Berorientasi pada objek (Object Oriented)
3. Bersifat terdistribusi
4. Bersifat Multiplatform
5. Bersifat MultiThread

Sedangkan pertimbangan menggunakan MySQL sebagai *database server* adalah sebagai berikut :

1. Dapat diperoleh secara gratis

2. Bersifat kapabilitas
3. Dapat berjalan di banyak platform
4. MySQL memiliki jaminan keamanan yang sangat baik.

5.3 Batasan Implementasi

Batasan implementasi meliputi batasan minimal untuk perangkat keras (*hardware*) dan perangkat lunak (*software*) yang diperlukan agar sistem yang dibuat dapat berjalan dengan baik. Batasan tersebut ditinjau dari sisi *server* sebagai pengelola sistem.

5.3.1 Batasan Implementasi Ditinjau Dari Sisi *Server*

Batasan implementasi ditinjau dari sisi server adalah sebagai berikut :

1. Segi Hardware, dalam hal ini penulis menggunakan PC (Personal Computer) dengan spesifikasi : Prosesor Intel Pentium 1,66 GHz dan RAM 256 MB atau yang lebih baik.
2. Segi Software, dalam hal ini penulis menggunakan software sebagai berikut :
    a. Sistem Operasi menggunakan Windows XP
    b. Pemrograman script menggunakan Java 2 SDK 1.4.2 (J2SE)
    c. Editor Text menggunakan Notepad++
    d. Database Server menggunakan MYSQL
    e. Data Source manggunakan ODBC 3.51

5.4. Implementasi

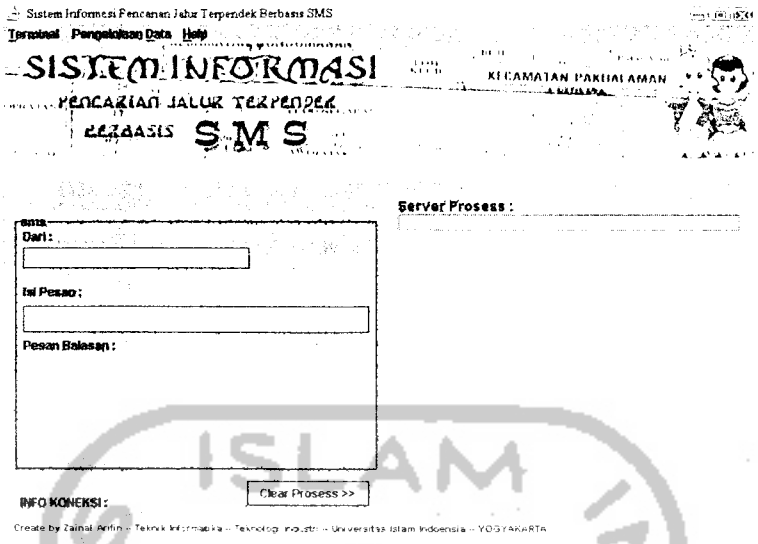
Sistem Informasi Pencarian Jalur Terpendek adalah perangkat lunak yang dibangun dengan fungsionalitas sebagai alat bantu (*tools*) untuk pencarian jalur terpendek dari suatu lokasi ke lokasi lainnya. Dengan menggunakan *tools* ini diharapkan konsumen bisa mendapat informasi yang diinginkan mengenai jalur mana saja yang harus dilewati jika akan menuju ke suatu lokasi.

5.4.1. Implementasi Antarmuka

5.4.1.1. Tampilan Halaman Utama Sistem

Halaman utama sistem adalah halaman yang pertama kali muncul pada saat sistem dijalankan. halaman ini berisi menu-menu nafigasi yang berguna untuk mengakses halaman yang lain. Adapun menu nafigasi terdiri dari menu terminal, pengelolaan data, dan menu help. Menu terminal terdiri dari submenu mulai yang berguna untuk memulai sistem, submenu berhenti yang berguna untuk menghentikan sistem, dan submenu exit yang berguna untuk keluar dari sistem. Menu pengelolaan data terdiri dari submenu pengelolaan data rumah sakit, data toko, data atm, data jalan, dan data graph. Menu help terdiri dari submenu version yang menunjukkan produk informasi dari sistem.

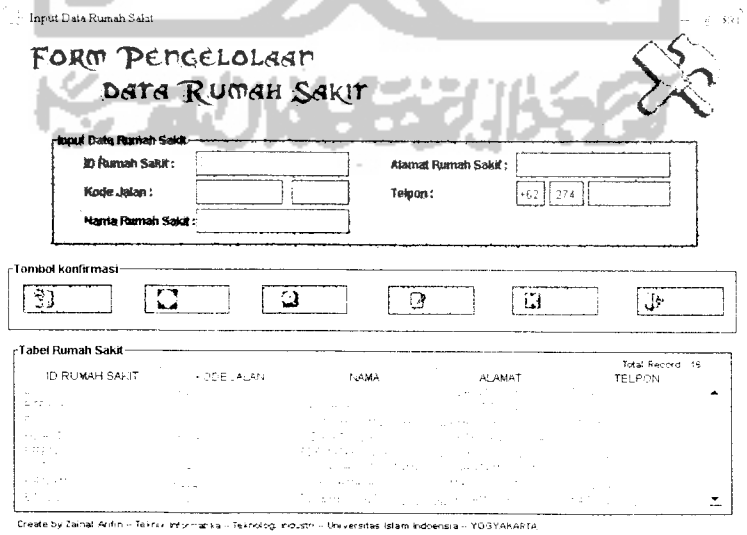
Halaman ini menampilkan nomor pengirim dan isi pesan SMS yang masuk serta balasan SMS yang akan dikirimkan oleh sistem. Halaman utama ini juga menampilkan proses yang terjadi pada server serta status koneksi yang sedang terjadi. Selengkapnya ditunjukkan pada Gambar 5.1 di bawah ini.



Gambar 5.1 Tampilan Halaman Utama Sistem

5.4.1.2. Tampilan Halaman Pengelolaan Data Rumah Sakit

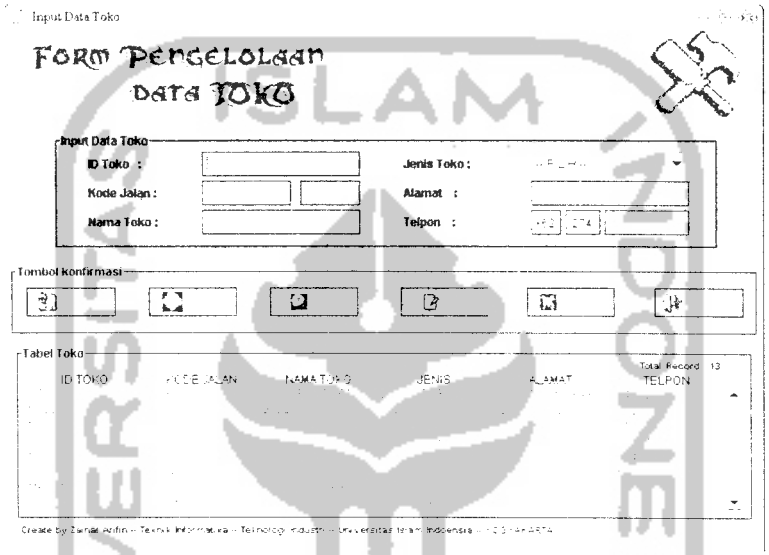
Halaman ini digunakan untuk melakukan pengelolaan terhadap data rumah sakit. Pengelolaan tersebut meliputi penambahan data rumah sakit, pencarian, pengubahan dan penghapusan data rumah sakit. Selengkapnya ditunjukkan pada Gambar 5.2 di bawah ini.



Gambar 5.2 Tampilan Halaman Pengelolaan Data Rumah Sakit

5.4.1.3. Tampilan Halaman Pengelolaan Data Toko

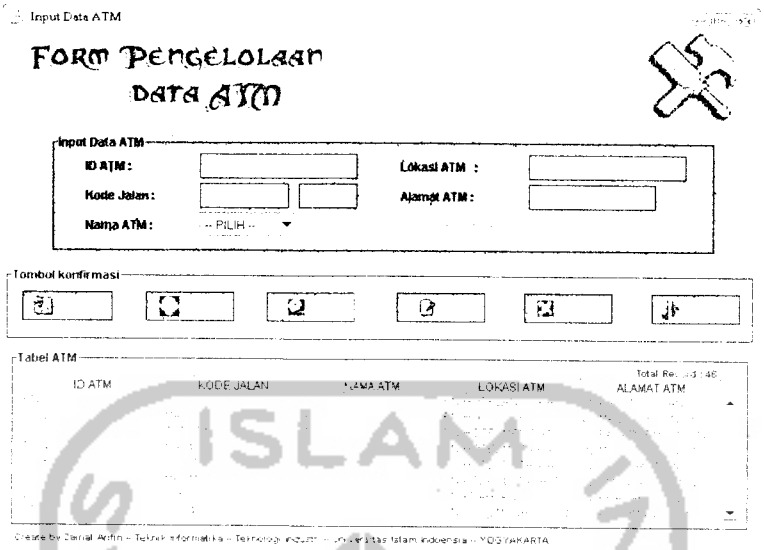
Halaman ini digunakan untuk melakukan pengelolaan terhadap data toko. Pengelolaan tersebut meliputi penambahan data toko, pencarian, pengubahan dan penghapusan data toko. Selengkapnya ditunjukkan pada Gambar 5.3 di bawah ini.



Gambar 5.3 Tampilan Halaman Pengelolaan Data Toko

5.4.1.4. Tampilan Halaman Pengelolaan Data Atm

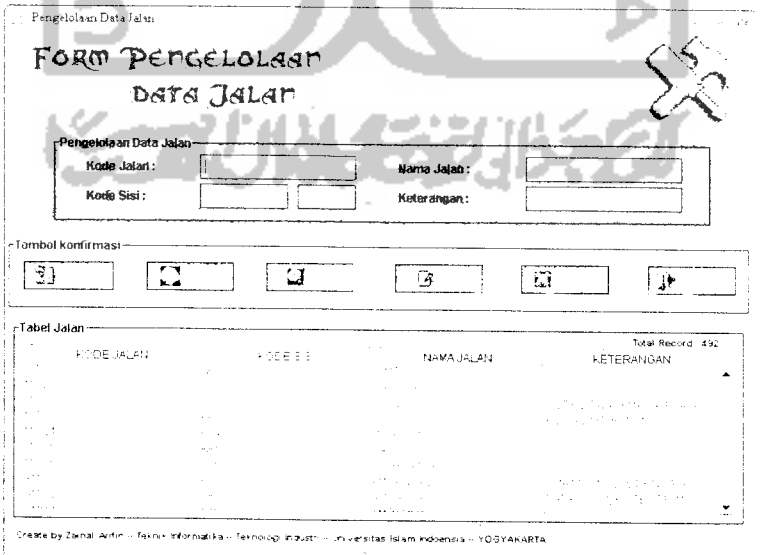
Halaman ini digunakan untuk melakukan pengelolaan terhadap data ATM. Pengelolaan tersebut meliputi penambahan data ATM, pencarian, pengubahan dan penghapusan data ATM. Selengkapnya ditunjukkan pada Gambar 5.4 di bawah ini.



Gambar 5.4 Tampilan Halaman Pengelolaan Data ATM

5.4.1.5. Tampilan Halaman Pengelolaan Data Jalan

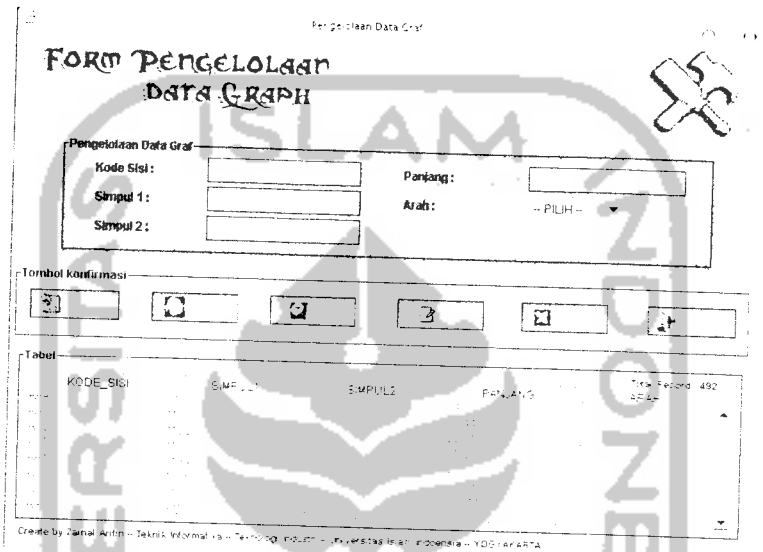
Halaman ini digunakan untuk melakukan pengelolaan terhadap data jalan. Pengelolaan tersebut meliputi penambahan data jalan, pencarian, pengubahan dan penghapusan data jalan. Selengkapnya ditunjukkan pada Gambar 5.5 di bawah ini.



Gambar 5.5 Tampilan Halaman Pengelolaan Data Jalan

5.4.1.6. Tampiian Halaman Pengelolaan Data Graph

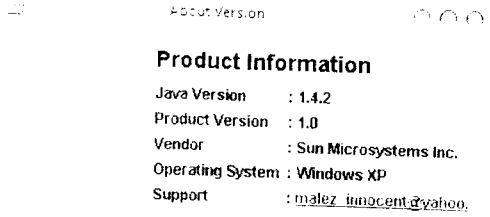
Halaman ini digunakan untuk melakukan pengelolaan terhadap data graph. Pengelolaan tersebut meliputi penambahan data graph, pencarian, pengubahan dan penghapusan data graph. Selengkapnya ditunjukkan pada Gambar 5.6 di bawah ini.



Gambar 5.6 Tampilan Halaman Pengelolaan Data Graph

5.4.1.7. Tampilan Halaman About Version

Halaman ini menampilkan informasi produk dari sistem. Adapun informasi yang ditampilkan adalah java version, produk version, vendor, dan sistem operasi yang digunakan oleh sistem. Selengkapnya ditunjukkan pada Gambar 5.7 di bawah ini :



Gambar 5.7 Tampilan Halaman About Version

5.5 Implementasi Prosedural

Dalam pembuatan aplikasi sistem informasi pencarian jalur terpendek berbasis SMS ini, terdapat pakage-package yang menyediakan class-class yang digunakan untuk menjalankan sistem. Clas-class tersebut adalah :

1. Class Indeks
2. Class Datajalan
3. Class Datarumahsakit
4. Class Datatoko
5. Class Dataatm
6. Class Datagraph
7. Class Graph
8. Class BinaryHeap

5.5.1 Class Indeks

Class ini berisi methode-methode yang digunakan untuk menjalankan sistem. Methode-methode tersebut digunakan untuk menghubungkan sistem dengan database, menghubungkan sistem dengan terminal, mengolah SMS yang masuk, dan membalas SMS kepada user. Di dalam class indeks, terdapat methode yang digunakan untuk melakukan proses query terhadap database sesuai dengan isi pesan SMS dari user. Methode tersebut berupa proses query data rumah sakit, proses query data toko dan proses query data ATM.

5.5.2 Class Datajalan

Class ini berisi methode-methode yang digunakan untuk melakukan pengelolaan terhadap data jalan. Pengelolaan yang dapat dilakukan adalah penambahan data jalan, pencarian, pengubahan dan penghapusan data jalan. Methode-methode tersebut adalah methode prosestambahdatajalan, prosescaridatajalan, proseseditdatajalan, dan proseshapusdatajalan.

5.5.3 Class Datarumahsakit

Class ini berisi methode-methode yang digunakan untuk melakukan pengelolaan terhadap data rumah sakit. Pengelolaan yang dapat dilakukan adalah penambahan data rumah sakit, pencarian data rumah sakit, pengubahan dan penghapusan data rumah sakit. Methode-methode tersebut adalah methode prosestambahdataRS, caridataRS, editdataRS, dan hapusdataRS.

5.5.4 Class Datatoko

Class ini berisi methode-methode yang digunakan untuk melakukan pengelolaan terhadap data toko. Pengelolaan yang dapat dilakukan adalah penambahan data toko, pencarian data toko, pengubahan data toko dan penghapusan data toko. Methode-methode tersebut adalah methode prosestambahdatatoko, caridatatoko, editdatatoko, dan hapusdatatoko.

5.5.5 Class DataATM

Class ini berisi methode-methode yang digunakan untuk melakukan pengelolaan terhadap data ATM. Pengelolaan yang dapat dilakukan adalah penambahan data ATM, pencarian data ATM, pengubahan dan penghapusan data ATM. Methode-methode tersebut adalah methode prosestambahdataatm, caridataatm, editdataatm, dan hapusdataatm.

5.5.6 Class Datagraph

Class ini berisi methode-methode yang digunakan untuk melakukan pengelolaan terhadap data graph. Pengelolaan yang dapat dilakukan adalah penambahan data graph, pencarian, pengubahan dan penghapusan data graph. Methode-methode tersebut adalah methode prosestambahdatagraf, caridatagraf, editdatagraf, dan hapusdatagraf.

5.5.7 Class Graph

Class ini berisi implementasi dari algoritma djikstra. Masukan yang digunakan untuk melakukan pencarian adalah vertex awal dan vertex akhir. Vertex awal berupa nama jalan atau kode jalan, dan vertex akhir berupa nama fasilitas umum yang akan dituju. Dalam kasus ini vertex akhir bisa berupa rumah sakit, toko atau ATM. Class graph ini menghasilkan jarak terpendek antara simpul asal dan simpul tujuan. Sedangkan untuk menentukan urutan jalur terpendek digunakan class BinaryHeap.

5.5.8 Class BinaryHeap

Class ini digunakan untuk melakukan proses pengurutan jarak terpendek. Urutan jarak terpendek bisa dicari dengan cara *backward* yang dimulai dari titik tujuan bergerak ke titik asalnya. Hasil akhir dari class BinaryHeap ini adalah nama-nama jalan yang harus dilalui.

5.5.9 Algoritma Djikstra

Algoritma ini digunakan untuk melakukan pencarian jalur terpendek. Implementasi dari algoritma djikstra terdapat pada methode djikstra.

Berikut ini adalah implementasi dari algoritma Djikstra dengan menggunakan BinaryHeap untuk menentukan jalur jalan terpendek.

```
/**************************************************
      * METHODE   : dijkstra
       * Algoritma Dijkstra
**************************************************/

public void dijkstra(String startName, String destName)
{
      clearAll();
      // Mendapatkan vertex awal
      Vertex start = (Vertex) vertexMap.get(startName);
      if (start == null)
      {
      System.out.println("Vertex awal tidak ditemukan");
      }

      // Untuk vertex awal, set nilai dist dengan 0
      start.dist = 0;

      // Membuat/ Inisialisasi priorityQueue
      BinaryHeap binH = new BinaryHeap();
      LinkedList bin = new LinkedList();
      binH.insert(new Path(start, 0));

      while (!binH.isEmpty())
      {
            // mendapatkan nilai terendah
            Path vrec = (Path)binH.deleteMin();
            Vertex v = vrec.dest;
            v.known = true;
```

```
                // kemudian telusuri vertex terhubung lainnya
                for (Iterator itr = v.adj.iterator(); itr.hasNext(); )
                {
                        Edge e = (Edge) itr.next();
                        Vertex w = e.dest;
                        double cw = e.cost;
                        double jml = v.dist+e.cost;
                        double jm = e.dest.dist;

                        // Mengganti nilai dist sebelumnya dengan nilai
dist yang lebih kecil
                        if (!e.dest.known)
                        {
                                if ((v.dist+e.cost) < e.dest.dist)
                                {
                                        e.dest.dist = v.dist + e.cost;
                                        e.dest.path = v;
                                        w.dist = v.dist + e.cost;
                                        w.path = v;
                                        binH.insert(new Path(w, w.dist));
                                }
                        }
                }
        }
}
```

Berikut ini adalah implementasi dari BinaryHeap yang digunakan untuk menghasilkan urutan jarak terpendek yang harus dilalui berupa nama-nama jalan.

```
/**************************************************
      * METHODE    : BinaryHeap
**************************************************/

public class BinaryHeap
    {
        private static final int DEFAULT_CAPACITY = 100;
        private int currentSize;  // Number of elements in heap
        private Comparable [ ] array; // The heap array
        public BinaryHeap( )
        {
            this( DEFAULT_CAPACITY );
        }
        public BinaryHeap( int capacity )
        {
            currentSize = 0;
            array = new Comparable[ capacity + 1 ];
        }
        public void insert( Comparable x )
        {
            if( isFull( ) )
            System.out.println("Full");
                //throw new OverflowException( );
                // Percolate up
```

```
            int hole = ++currentSize;
            for( ; hole > 1 && x.compareTo( array[ hole / 2 ] ) <
0; hole /= 2 )
            array[ hole ] = array[ hole / 2 ];
            array[ hole ] = x;
        }
        public Comparable findMin( )
        {
            if( isEmpty( ) )
                return null;
            return array[ 1 ];
        }
        public Comparable deleteMin( )
        {
            if( isEmpty( ) )
                return null;
            Comparable minItem = findMin( );
            array[ 1 ] = array[ currentSize-- ];
            percolateDown( 1 );
            return minItem;
        }
        private void buildHeap( )
        {
            for( int i = currentSize / 2; i > 0; i-- )
                percolateDown( i );
        }
        public boolean isEmpty( )
        {
            return currentSize == 0;
        }
        public boolean isFull( )
        {
            return currentSize == array.length - 1;
        }
        public void makeEmpty( )
        {
            currentSize = 0;
        }
        private void percolateDown( int hole )
        {
            int child;
            Comparable tmp = array[ hole ];
            for( ; hole * 2 <= currentSize; hole = child ) {
            child = hole * 2;
            if( child != currentSize && array[ child + 1
].compareTo( array[ child ] ) < 0 )
            child++;
            if( array[ child ].compareTo( tmp ) < 0 )
            array[ hole ] = array[ child ];
                  else
                  break;
                  }
            array[ hole ] = tmp;
         }
    }
```