# BAB III

# METODOLOGI PENELITIAN

## 3.1 Metodologi

Metode penelitian yang akan digunakan dalam penelitian ini antara lain adalah sebagai berikut:

1. Studi Literatur

Tahap awal dilakukan studi mengenai referensi-referensi yang berhubungan dengan dengan pokok bahasan penelitian, seperti *Image Processing* dan ekstrasi fitur *Sobel Edge Detection*, *server-client*, algoritma *Template Matching* dan berbagai konsep pendukung lainnya. Referensi-referensi tersebut dapat berupa jurnal, artikel, buku, dan lain-lain

2. Implementasi pada Aplikasi

Implementasi *template matching* pada aplikasi ini dibagi menjadi tiga tahapan. Pertama, melakukan perancangan awal terhadap aplikasi *server* dan *client* dalam bentuk diagram *Unified Modeling Language* (UML). Kedua, melakukan perancangan terhadap fitur *Image Processing*, dan *Feature Extraction* pada aplikasi *server*. Ketiga, melakukan perancangan terhadap aplikasi *client* yang berbasis bahasa pemrograman Java pada perangkat Android. Keempat, melakukan implementasi metode *Template Matching* menggunakan rumus *Sums of Absolute Difference* pada proses pengidentifikasi jenis retak pada jalan beraspal di dalam *server*.

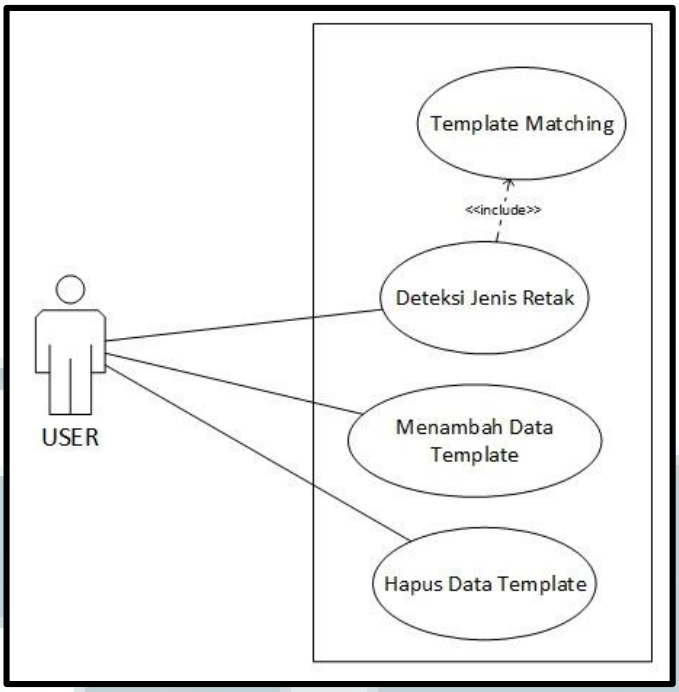
3. Uji Coba dan Evaluasi

Melakukan uji coba terhadap aplikasi disertai dengan evaluasi hasil yang didapatkan. Uji coba ini bertujuan untuk mencari *bug* yang masih ada pada aplikasi tersebut dan memperbaiki aplikasi tersebut. Melakukan pengisian *database template* dengan validasi kebenaran oleh pakar. Kemudian melakukan eksperimen menentukan *variable* terbaik dari ukuran megapiksel dari foto dan ukuran gambar untuk gambar *testing* dan *template* setelah di-*resize*. Lalu melakukan eksperimen menghitung akurasi ketepatan mendeteksi jenis retak dari gambar *testing* disertai validasi kebenaran oleh pakar.

## 3.2 Perancangan Sistem

### 3.2.1 Diagram *Use Case*

*Use Case Diagram* adalah gambaran *graphical* dari beberapa atau semua aktor, komponen, dan interaksi di antara komponen-komponen tersebut yang memperkenalkan suatu sistem yang akan dibangun. *Use case* diagram menjelaskan manfaat suatu sistem jika dilihat menurut pandangan orang yang berada di luar sistem. Diagram ini menunjukkan fungsionalitas suatu sistem atau kelas dan bagaimana sistem tersebut berinteraksi dengan dunia luar. Berikut ini diagram *use case* sistem secara keseluruhan.



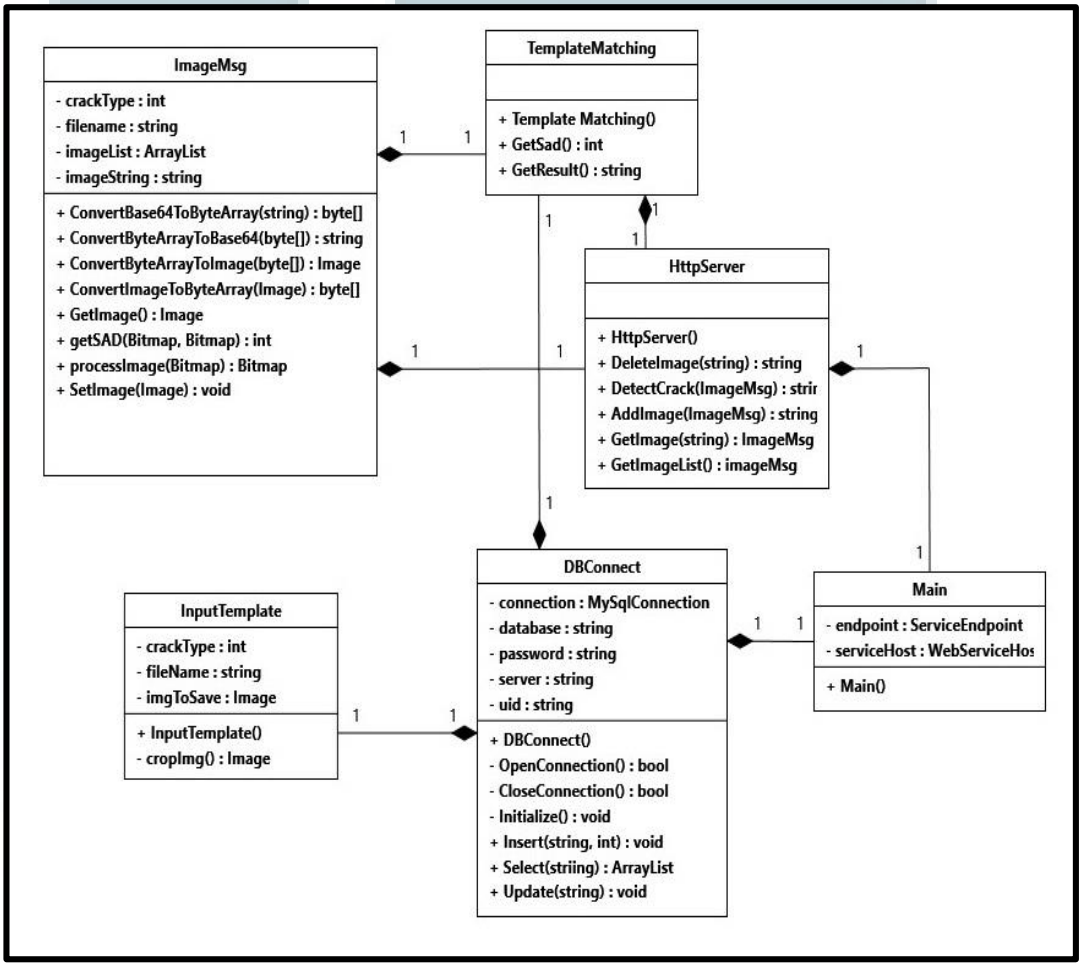

Gambar 3.1 Diagram *Use Case* sistem

Pada diagram *use case* sistem, terdapat *user* sebagai aktor. Aktor *user* dapat melakukan fungsi dektesi jenis retak, menambah data *template* dan *browse* serta hapus data *template*. Pada saat proses deteksi retak jalan diinisiasi oleh *user*, aplikasi pada perangkat Android mengirim request ke *server* untuk melakukan proses *template matching*. Hasil deteksi akan dikirim kembali ke perangkat Android untuk dikonfirmasi oleh *user*. Proses penambahan data *template* harus disertai validasi oleh pakar.

### 3.2.2 *Class Diagram*

*Class Diagram* atau Diagram Kelas adalah diagram UML yang menggambarkan kelas dalam sebuah sistem dan hubungannya antara satu dengan yang lain, serta dimasukkan atribut dan *method* atau operasi. *Class Diagram* sistem terbagi atas dua diagram, yaitu diagram kelas untuk *server* yang menggunakan bahasa pemrograman C#, dan diagram kelas untuk *client* di perangkat Android yang menggunakan bahasa pemrograman Java.

A. *Class Diagram* bagian *Server*

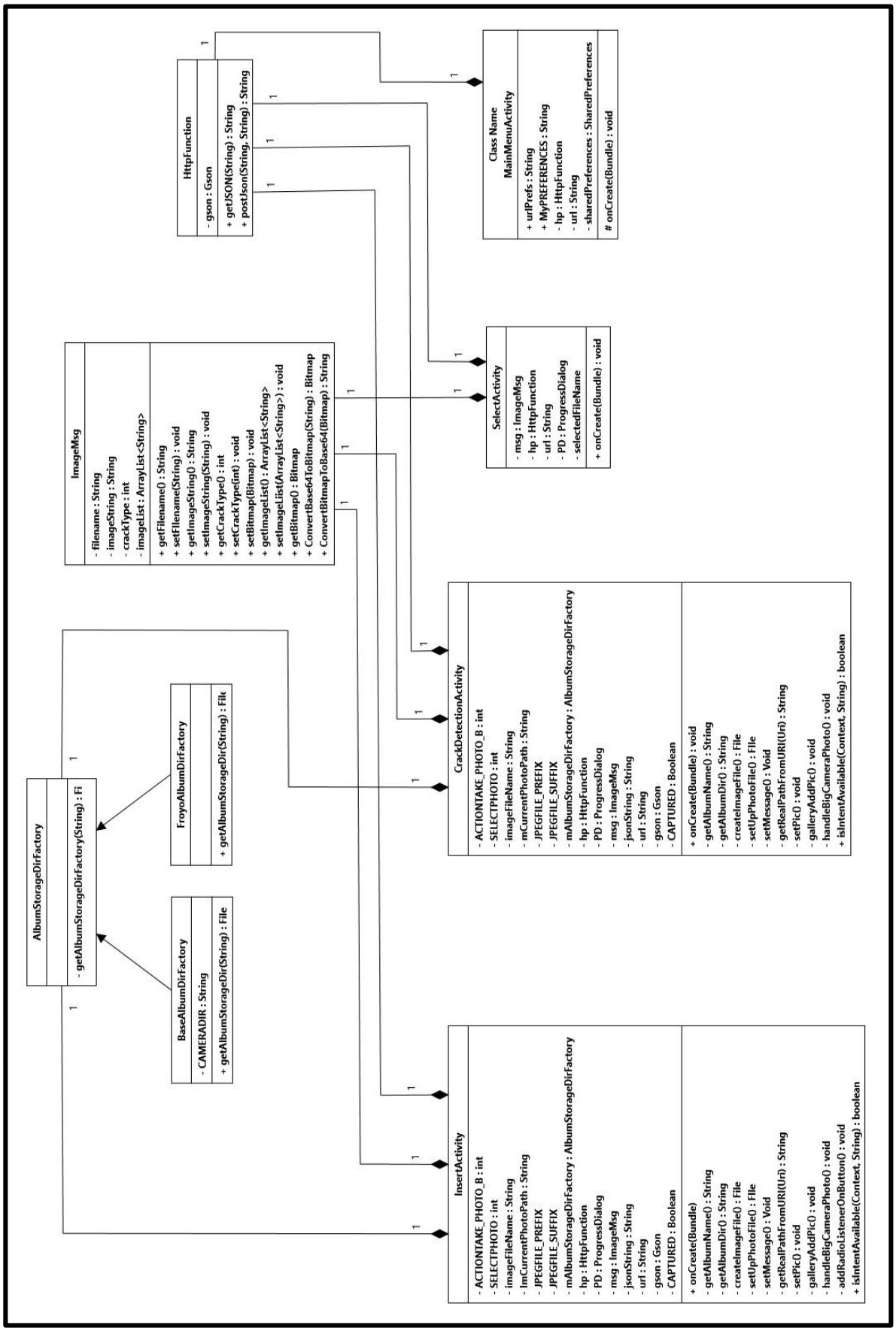



Gambar 3.2 *Server Class Diagram*

Pada gambar *Class Diagram* di atas terdiri dari enam kelas, yaitu kelas Main, kelas InputTemplate, kelas TemplateMatching, kelas DBConnect, kelas HttpServer, dan kelas ImageMsg. Kelas Main dan InputTemplate merupakan kelas *user interface form* yang berinteraksi langsung dengan pengguna. Kelas Main berfungsi sebagai pengendali utama dan pengawas *server* menjalankan tugasnya. Sedangkan kelas InputTemplate sebagai *form* data *template* yang akan di-*insert* ke dalam *database*. Kemudian kelas TemplateMatching berfungsi sebagai kelas utama metode *template matching* dengan rumus SAD.

Kelas DBConnect berfungsi menyambungkan aplikasi dengan *database* MySQL yang berisi segala fungsi *editor database* seperti menambah data (*insert*), memperbarui data (*update*), menyeleksi data (*select*), dan sebagainya. Kelas HttpServer berfungsi sebagai perantara aplikasi *server* dalam berkomunikasi dengan aplikasi *client* menggunakan *Web Service* dan berbasis bahasa *parser* JSON. Dalam berkomunikasi, kelas HttpServer membaca objek JSON yang di-*request* oleh *client* untuk eksekusi. Kelas terakhir yaitu kelas ImageMsg berfungsi sebagai format objek *message* yang digunakan oleh *server-client* untuk berkomunikasi yang kemudian dikonversi menjadi objek JSON.

B. *Class Diagram* bagian *Client*




Gambar 3.3 *Client Class Diagram*

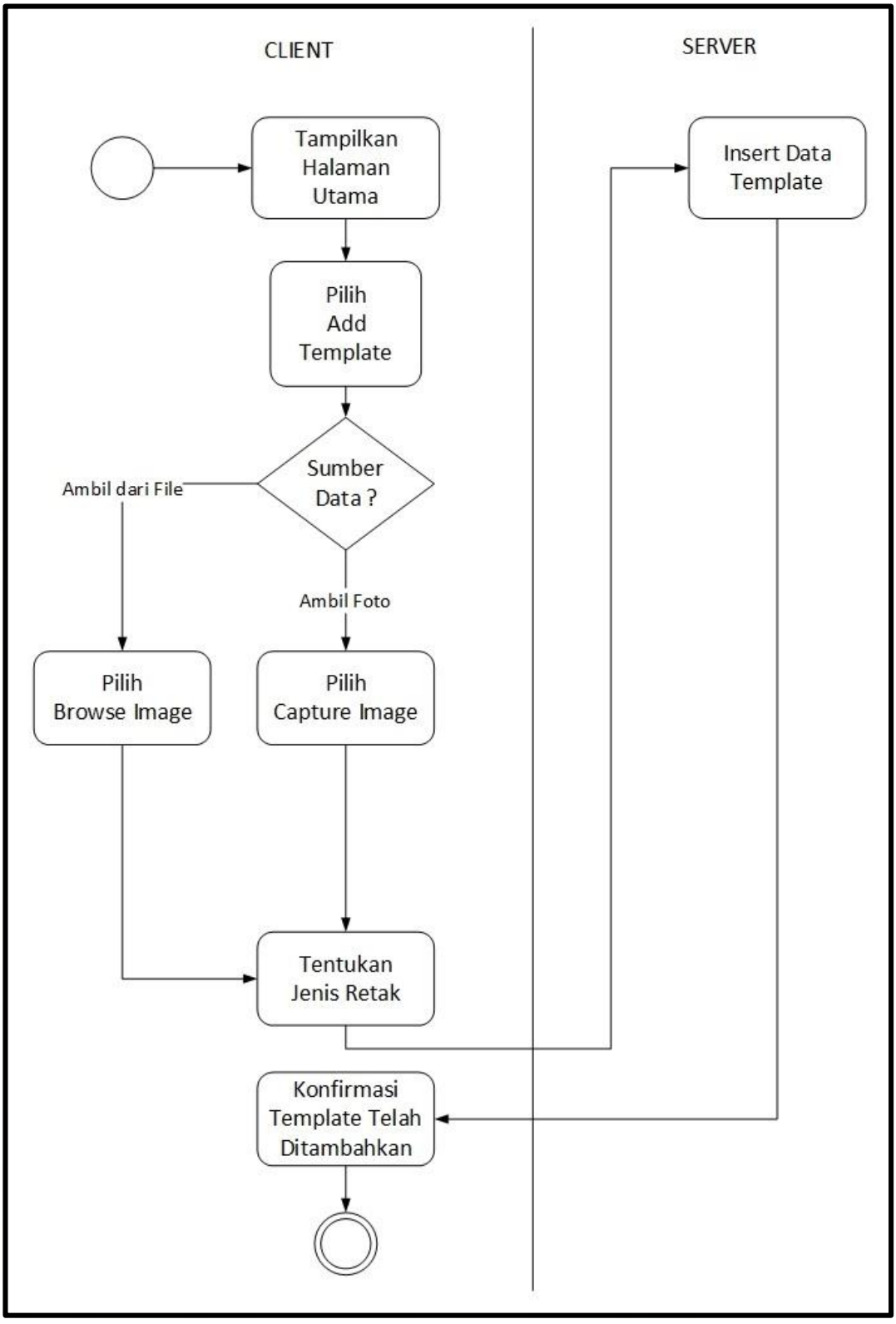
Pada gambar *Class Diagram* bagian *client* yang menggunakan bahasa pemrograman Java terdiri dari sembilan kelas, yaitu kelas MainMenuActivity, kelas CrackDetectionActivity, kelas InsertActivity, kelas SelectActivity, kelas AlbumDirStorageFactory, kelas BaseAlbumDirFactory, kelas FroyoAlbumDirFactory, kelas HttpFunction dan kelas ImageMsg. Kelas *activity* atau kelas yang berfungsi sebagai *user interface*, yaitu kelas MainMenuActivity, kelas CrackDetectionActivity, kelas InsertActivity dan kelas SelectActivity. Kelas MainMenuActivity merupakan menu utama aplikasi yang diakses pertama kali oleh pengguna setelah membuka aplikasi. Kelas CrackDetectionActivity berfungsi sebagai *user interface* interaksi pengguna untuk mendeteksi jenis retak yang diinginkan pengguna. Apabila pengguna mau menambahkan gambar ke *database template*, pengguna harus melalui *user interface* kelas InsertActivity. Sedangkan, pengguna dapat menghapus gambar *template* yang diinginkan melalui *user interface* kelas SelectActivity.

Kelas AlbumDirStorageFactory, kelas BaseAlbumDirFactory, dan kelas FroyoAlbumDirFactory merupakan kelas penentu *storage* dalam perangkat *client* yang digunakan untuk menyimpan foto sementara sebelum diproses lebih lanjut. Kelas HttpFunction dan kelas ImageMsg merupakan kelas yang berkomunikasi dengan *server* dimana kelas HttpFunction berfungsi untuk mengirimkan *request* dan *message* ke *server*, dan ImageMsg berfungsi sebagi format objek dari *message* yang dikirim dan diterima.

### 3.2.3 *Activity Diagram*

Diagram Aktivitas atau *Activity Diagram* adalah representasi grafis dari seluruh tahapan alur kerja. Diagram ini mengandung aktivitas, pilihan tindakan, perulangan dan hasil dari aktivitas tersebut. Pada pemodelan UML, diagram ini dapat digunakan untuk menjelaskan proses bisnis dan alur kerja operasional secara langkah demi langkah dari komponen suatu sistem. *Activity Diagram* aplikasi terdiri dari empat diagram, yaitu diagram proses menambah data *template*, diagram proses deteksi jenis retak, diagram proses hapus data *template* yang diinginkan, dan diagram proses *template matching*.

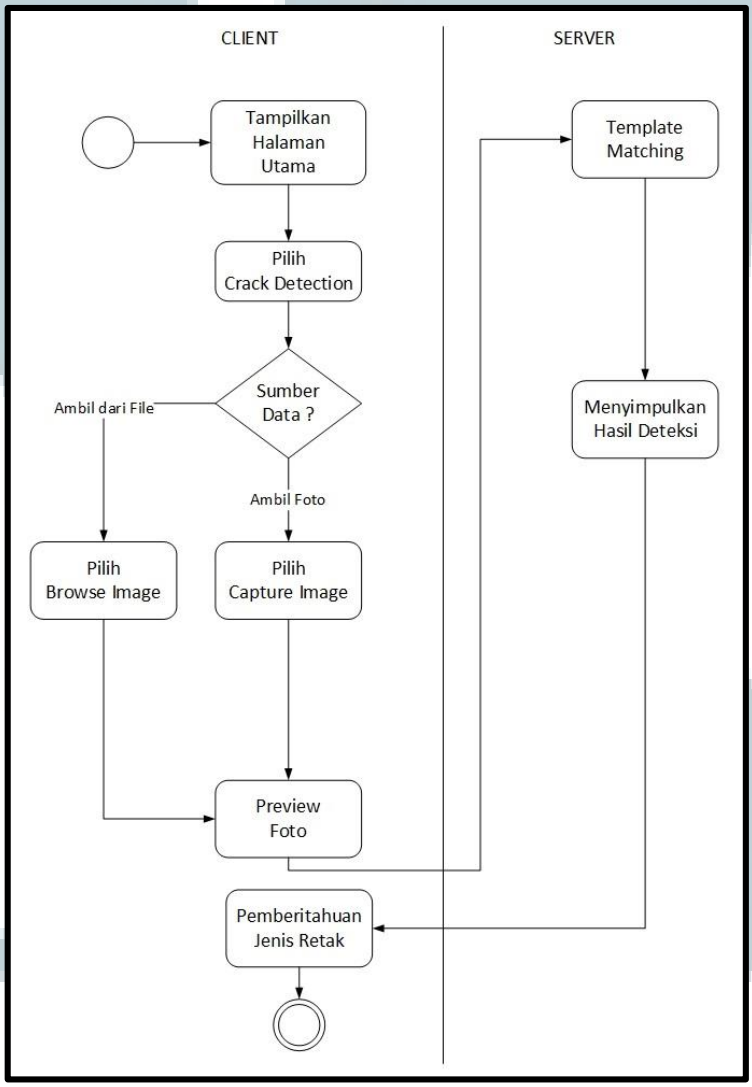
A. *Activity Diagram* Proses Menambah Data *Template*




Gambar 3.4 *Activity Diagram* proses menambah data *template*

Proses ini dimulai dengan memilih menu *Add Template* pada halaman utama yang dilanjutkan dengan mengambil foto atau *browse* gambar dari *gallery*. Kemudian pengguna menentukan jenis retak berdasarkan ciri-ciri jenis retak. Foto atau gambar yang sudah ditentukan akan diunggah ke *server*. Lalu *server* akan mengirim konfirmasi ke *client* bahwa gambar telah sukses ditambahkan ke *database*.

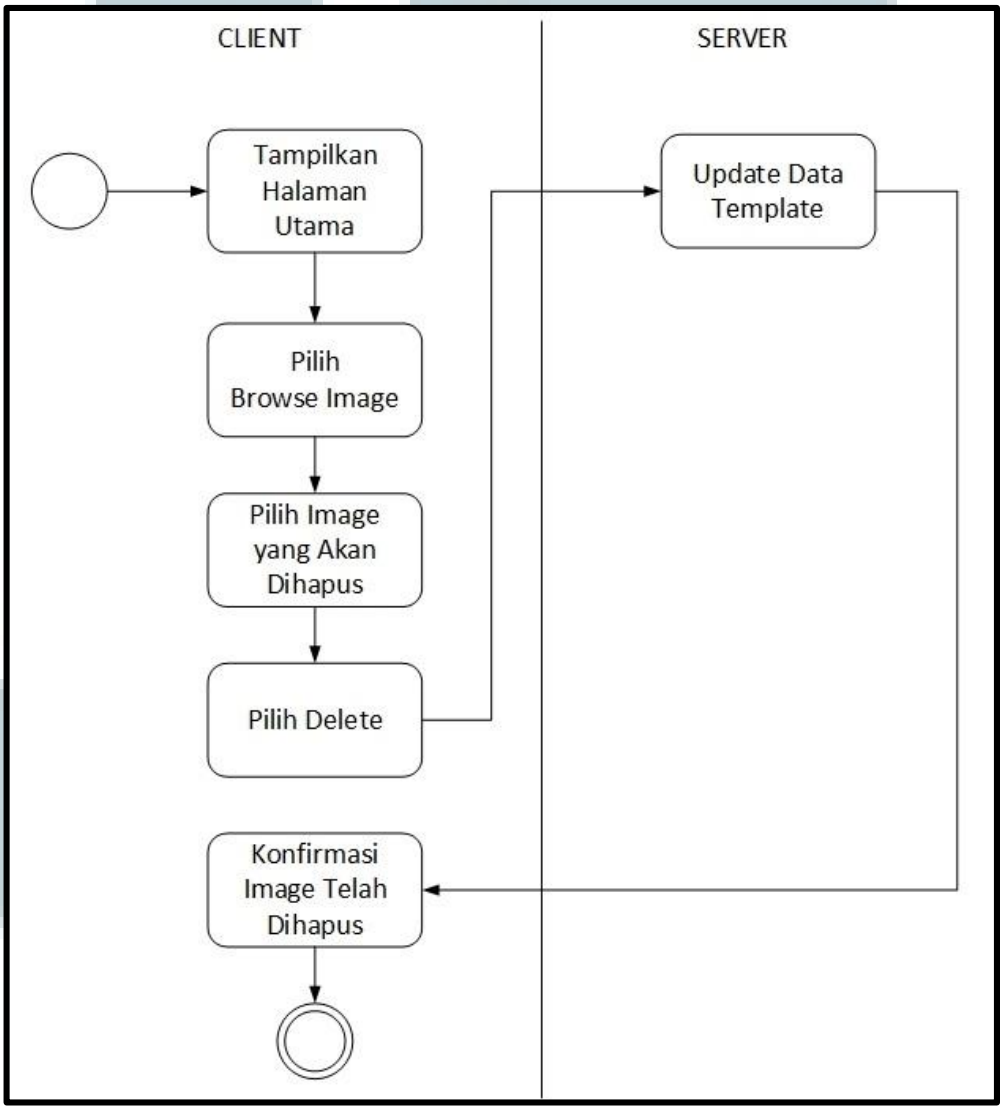
B. *Activity Diagram* Proses Deteksi Jenis Retak Jalan




Gambar 3.5 *Activity Diagram* proses deteksi retak jalan

Proses ini dimulai dengan memilih menu *Crack Detection* pada halaman utama yang dilanjutkan dengan mengambil foto atau *browse* gambar dari *gallery*. Foto atau gambar yang telah dipilih akan masuk jendela *preview*. Kemudian, pengguna mengunggah gambar atau foto tersebut ke *server* untuk dilakukan proses *template matching*. Proses berikutnya *server* akan mengirim pesan konfirmasi beserta hasil deteksi jenis retak untuk ditampilkan di *client*.

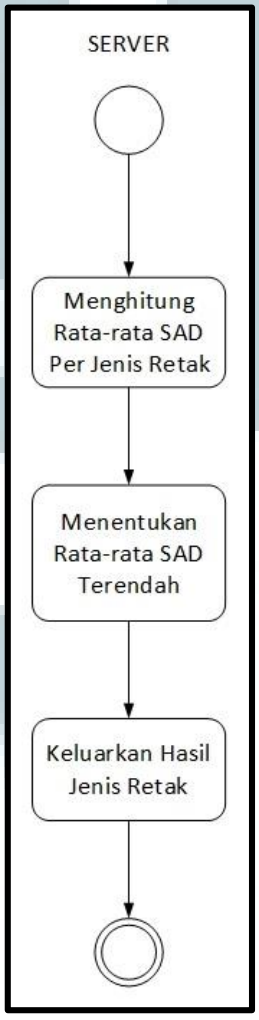
C. *Activity Diagram* Proses Hapus Data Template




Gambar 3.6 *Activtiy Diagram* proses hapus data *template*

Proses ini dimulai memilih menu *Browse Templates* pada halaman utama yang dilanjutkan dengan pengguna memilih *filename* atau nama gambar *template* yang dikehendaki. Setelah memilih gambar, pengguna dapat menghapus gambar tersebut dengan menekan tombol *DELETE*. Kemudian, *client* mengirim *request* kepada *server* untuk menghapus gambar dengan *filename* atau nama gambar yang telah dipilih. Lalu *server* mengirim konfirmasi ke *client* bahwa proses penghapusan data *template* telah berhasil.

*D. Activity Diagram* Proses *Template Matching*




Gambar 3.7 *Activity Diagram* proses *Template Matching*

Proses ini sepenuhnya dilakukan di bagian *server* yang dimulai dengan menghitung nilai SAD (*Sums of Absoulte Difference*) untuk setiap gambar *template* per jenis retak terhadap gambar *testing*. Kemudian dilanjutkan dengan menghitung rata-rata SAD per jenis retak terhadap *testing*. Apabila nilai rata-rata SAD jenis retak tertentu berada di atas nilai toleransi kecocokan dan lebih besar dari rata-rata SAD jenis retak lainnya, gambar *testing* tersebut akan ditambahkan sebagai *template* jenis retak tersebut ke dalam *database template*.

### 3.2.4 *Sequence Diagram*

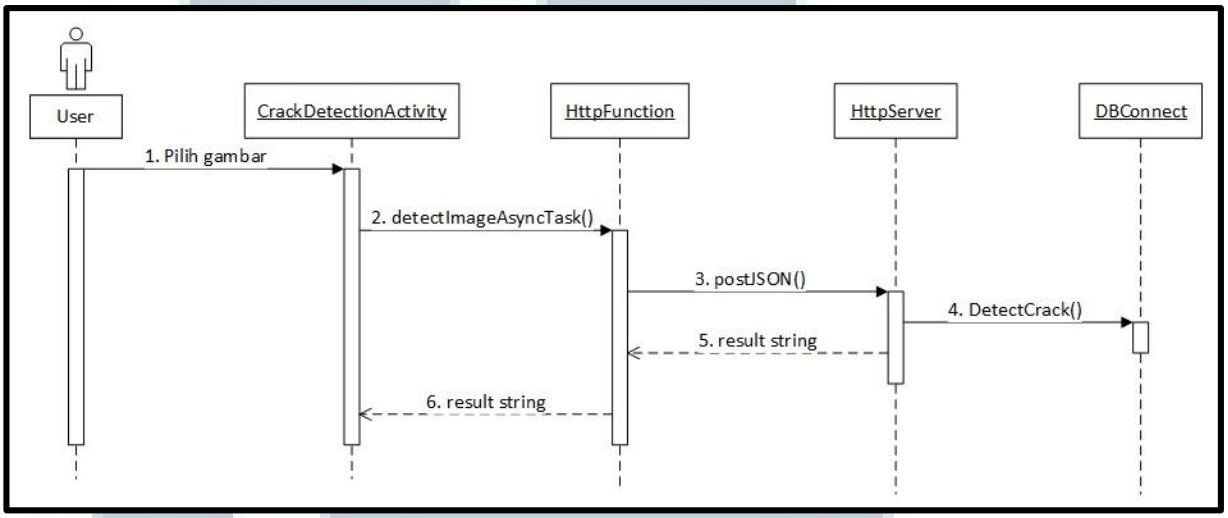
*Sequence Diagram* adalah suatu diagram yang menggambarkan interaksi antar obyek dan mengindikasikan komunikasi diantara obyek-obyek tersebut. Diagram ini juga menunjukkan serangkaian pesan yang berkomunikasi antara obyek-obyek yang melakukan suatu tugas atau aksi tertentu. Obyek-obyek tersebut kemudian diurutkan dari kiri ke kanan, aktor yang menginisiasi interaksi biasanya ditaruh di paling kiri dari diagram.

Pada diagram ini, dimensi vertikal merepresentasikan waktu. Bagian paling atas dari diagram menjadi titik awal dan waktu berjalan ke bawah sampai dengan bagian dasar dari diagram. Garis vertikal disebut *lifeline*, dilekatkan pada setiap obyek atau aktor. Kemudian, *lifeline* tersebut digambarkan menjadi kotak ketika obyek melakukan suatu operasi, kotak tersebut disebut *activation box*. Obyek dikatakan mempunyai *live activation* pada saat tersebut.

Pesan yang dipertukarkan antar obyek digambarkan sebagai sebuah anak panah antara activation box pengirim dan penerima. Kemudian diatasnya diberikan label

pesan. *Sequence diagram* sistem terdiri dari empat diagram, yaitu diagram proses deteksi retak jalan, diagram proses menambah data *template*, diagram proses menghapus data *template*, dan diagram proses *template matching*.

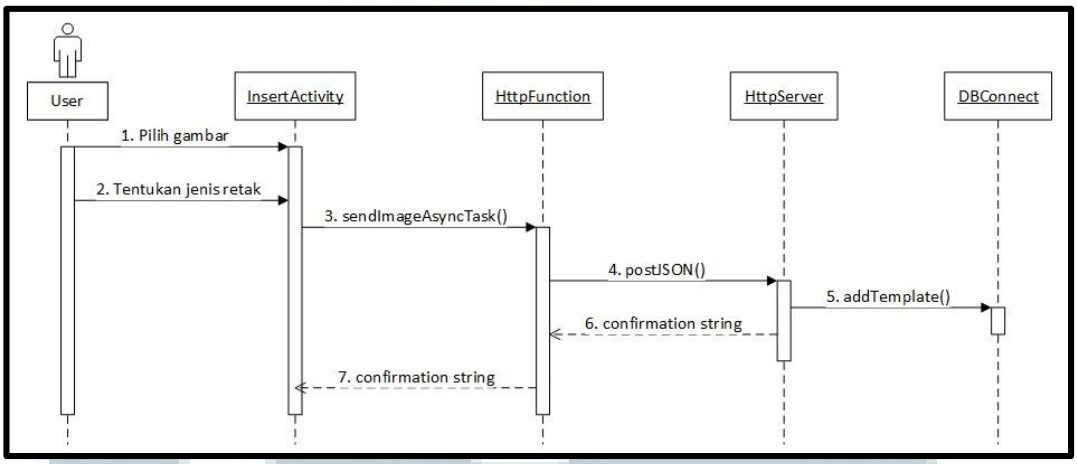
A. *Sequence Diagram* Proses Deteksi Jenis Retak Jalan




Gambar 3.8 *Sequence Diagram* proses deteksi retak jalan

Pada gambar *Sequence Diagram* di atas, proses deteksi jenis retak dimulai dari pengguna memilih gambar yang diperoleh dari foto kamera atau *browse* gambar dari *gallery*. Kemudian gambar yang dipilih tersebut diproses menjadi sebuah objek JSON di kelas CrackDetectionActivity yang kemudian diubah menjadi *string* JSON untuk dikirimkan ke *server*. Setelah itu *server* mengakses *database* melalui DBConnect untuk melakukan algoritma *template matching*, Hasil perhitungan berupa *string* jenis retak dari gambar yang dipilih pengguna dikirimkan kembali sebagai respon melalui langkah proses yang sama, yaitu *string* hasil deteksi diubah menjadi *string* JSON yang kemudian dikirim ke HttpFunction di *client*. Lalu ditampilkan di *user interface* kelas CrackDetectionActivity.

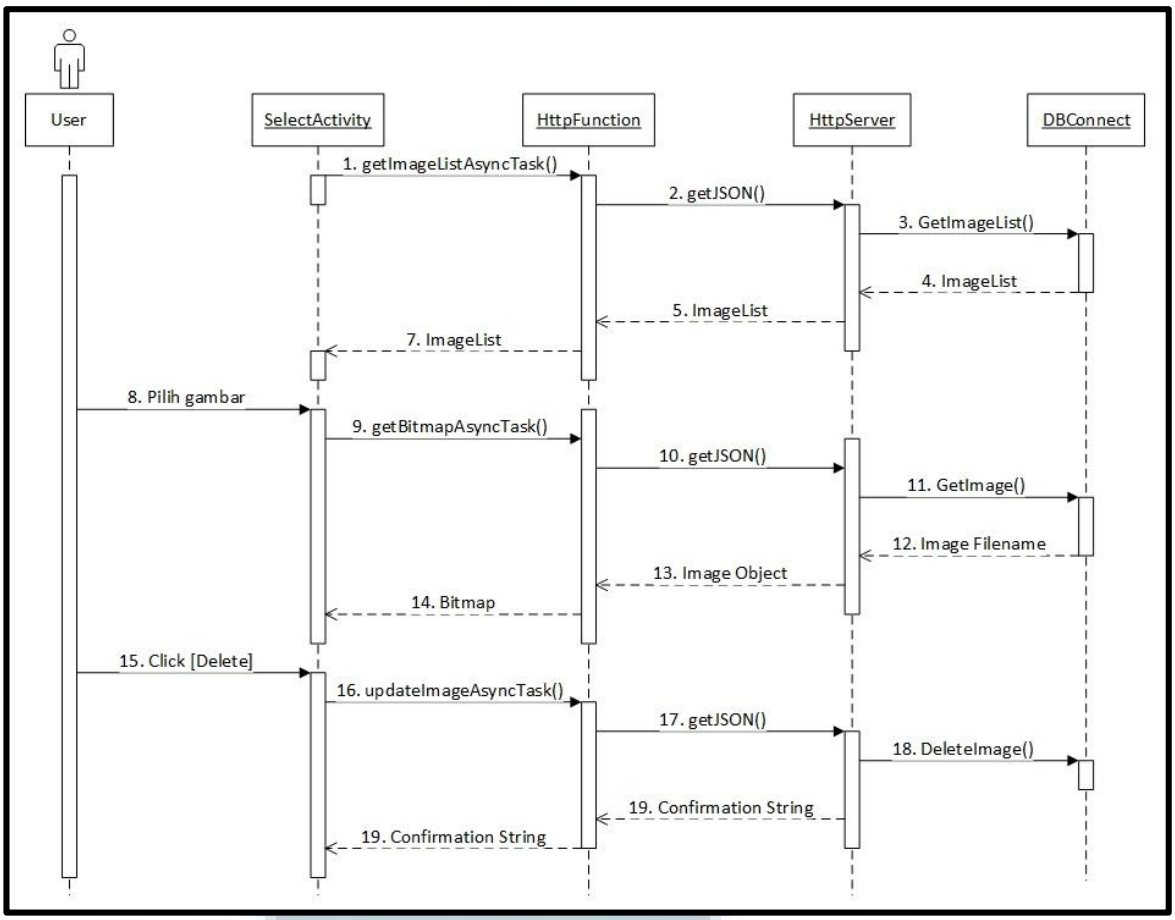
B. *Sequence Diagram* Proses Menambah Data *Template*




Gambar 3.9 *Sequence Diagram* proses menambah data *template*

Pada gambar *Sequence Diagram* di atas, proses penambahan data *template* dimulai dari pengguna memilih gambar yang diperoleh dari foto kamera atau *browse* gambar dari *gallery*. Kemudian pengguna menentukan jenis retak dari gambar yang dipilih tersebut untuk diproses menjadi sebuah objek JSON di kelas InsertActivity yang kemudian diubah menjadi *string* JSON untuk dikirimkan ke *server*. Setelah itu *server* mengakses *database* melalui DBConnect untuk *insert* data *template* dari pengguna ke dalam *database template*. Hasil konfirmasi data telah di-*insert* berupa *string* dikirimkan kembali sebagai respon melalui langkah proses yang sama, yaitu *string* hasil diubah menjadi *string* JSON yang kemudian dikirim ke HttpFunction di *client*. Lalu ditampilkan di *user interface* kelas InsertActivity.

C. *Sequence Diagram* Proses Hapus Data *Template*



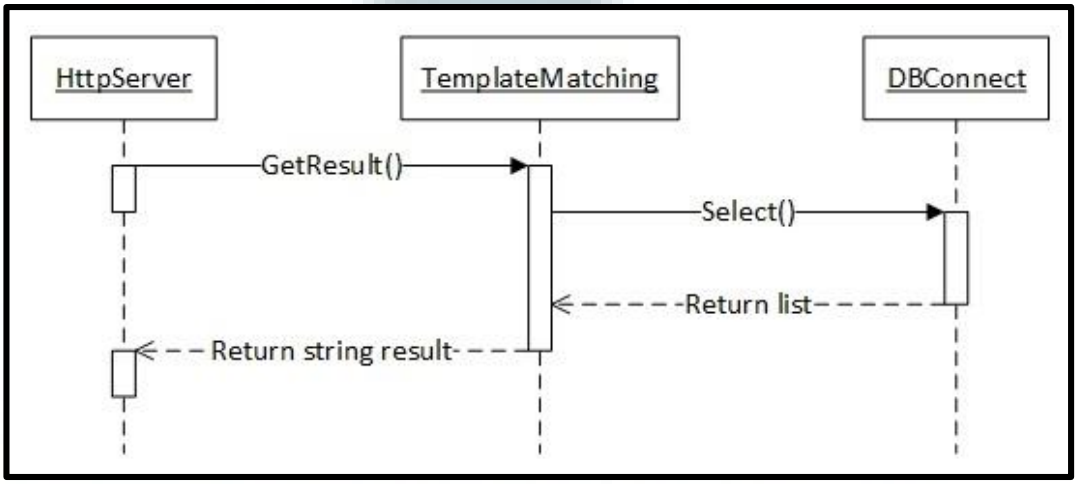

Gambar 3.10 *Sequence Diagram* proses hapus data *template*

Pada gambar *Sequence Diagram* di atas, proses penambahan data *template* dimulai dari pengguna memilih menu *Browse Template* yang kemudian dialihkan ke kelas *user interface* SelectActivity. Ketika *user interface* di-*load* pertama kali, aplikasi langsung melakukan *request* daftar gambar *template* ke *server*. Kemudian *server* mengakses *database* melalui DBConnect untuk *select* semua data *template* untuk dikirim balik sebagai respon *request client*. Respon berupa *array* dari daftar *template* tersebut dimasukkan ke dalam *spinner*. Setelah daftar *template* sudah bisa

dipilih melalui *spinner*, pengguna dapat memilih *filename* dari gambar *template* untuk ditampilkan ke dalam bingkai gambar. Proses pemgambilan gambar sama seperti pengambilan list, yaitu *client* melakukan *request* gambar berdasarkan *filename* yang dipilih pengguna. Kemudian *server* memberikan respon berupa gambar yang memiliki *filename* tersebut untuk ditampilkan ke dalam bingkai gambar.

Langkah berikutnya pengguna dapat menghapus gambar *template* yang dipilih dengan menekan tombol *DELETE* di bawah bingkai gambar. Proses penghapusan kurang lebih sama dengan proses sebelumnya, yaitu *client* melakukan *request* penghapusan data *template* ke *server*. Kemudian *server* mengakses *database* melalui DBConnect untuk melakukan *update* pada data *template* yang dipilih. Data *template* yang di-*update* tidak akan muncul lagi pada saat *client* melakukan *request* daftar data *template* untuk memenuhi *spinner*.

D. *Sequence Diagram* Proses Template Matching




Gambar 3.11 *Sequence Diagram* proses *Template Matching*

Pada gambar Sequence Diagram di atas, proses template matching tidak melibatkan aktor. Pertama-pertama, kelas HttpServer mengirim objek *message* JSON yang sudah dikonversi ke objek gambar ke kelas TemplateMatching. Kemudian dilakukan proses *template matching* dengan menghitung nilai SAD untuk setiap gambar *template*. Kelas *template matching* melakukan pemanggilan kelas DBConnect untuk mengakses *database* dan melakukan *query select*. DBConnect merespon dengan mengirim daftar *template* yang kemudian diproses oleh kelas TemplateMatching untuk mendapatkan nilai rata-rata SAD per jenis retak. Hasil perhitungan berupa *string* jawaban jenis retak akan dikembalikan ke kelas HttpServer.

### 3.3 Perancangan Database

*Database* yang digunakan oleh sistem adalah *MySql*. Berikut struktur tabel – tabel yang terdapat di dalamnya.

1. Nama Tabel : template

   Fungsi : menyimpan daftar *filename* gambar *template* beserta jenis retaknya

   *Primary Key* : -

   *Foreign Key* : -

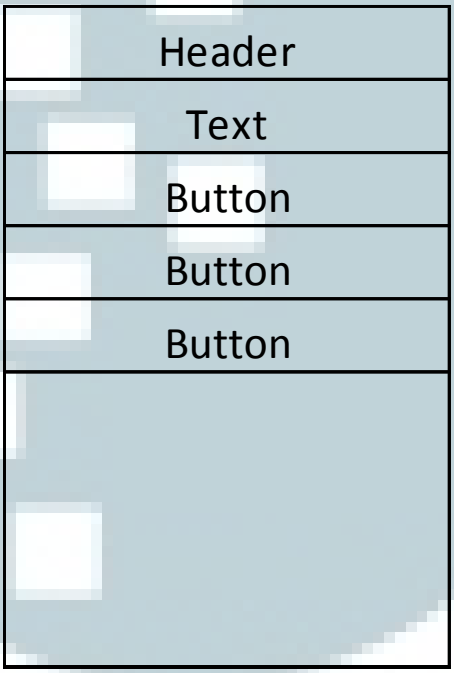
Tabel 3.1 Tabel template

| No | Nama Field | Tipe Data | Panjang | Keterangan |
|---|---|---|---|---|
| 1 | template_no | int | | |
| 2 | template_filename | varchar | 40 | Nama gambar |
| 3 | template_crack_type | int | | Jenis retak pada gambar |

### 3.4 Desain Antarmuka

Pada bagian ini, ditampilkan beberapa sketsa antarmuka sistem dari sisi pengguna.

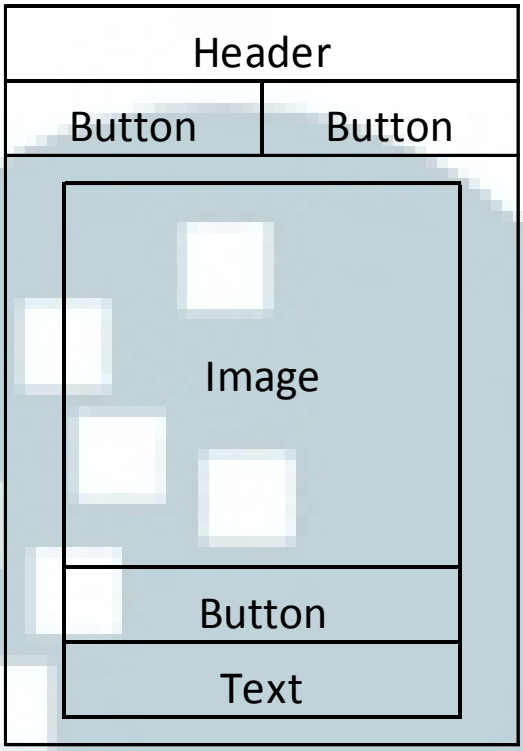
3.4.1 Halaman Utama




Gambar 3.12 Desain halaman menu utama

Halaman ini adalah halaman yang pertama kali ditampilkan ketika pengguna membuka aplikasi pada perangkat Android. Pada bagian atas terdapat bagian *header*. Kemudian pada bagian bawah *header* terdapat *textfield* yang dapat diisi dan di-*edit*. Pada bagian bawah *textfield* terdapat tiga buah tombol atau *button* navigasi untuk berpindah halaman.

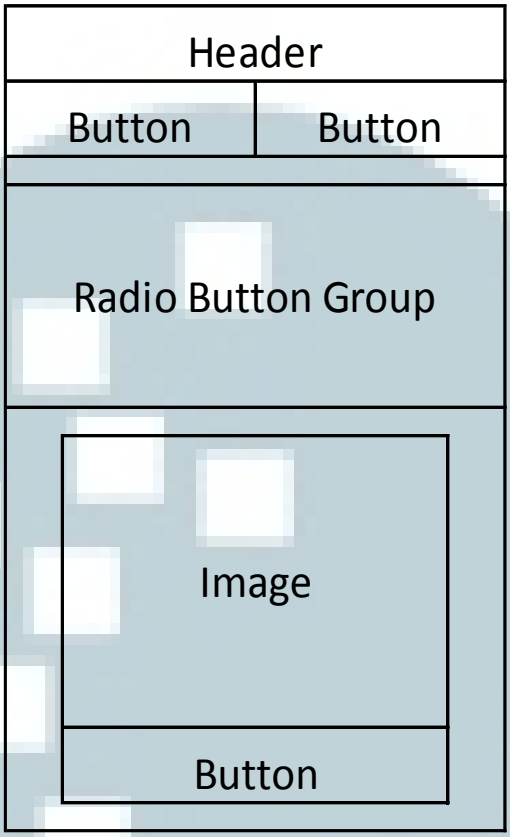
3.4.2 Halaman *Crack Detection*




Gambar 3.13 Desain halaman *Crack Detection*

Pada halaman *Crack Detection*, terdapat tiga buah tombol, sebuah bingkai gambar atau *Image view*, dan sebuah *textfield*. Bagian *textfield* berfungsi untuk menampilkan hasil deteksi jenis retak dari gambar *testing* yang diambil pengguna. Bagian *textfield* tersebut tidak dapat diisi maupun di-*edit*. Pengguna dapat mengambil gambar melalui kedua tombol pada bagian atas. Kemudian tombol bagian bawah berfungsi untuk memulai proses deteksi jenis retak yang kemudian memunculkan hasil deteksi ke dalam *textfield*.

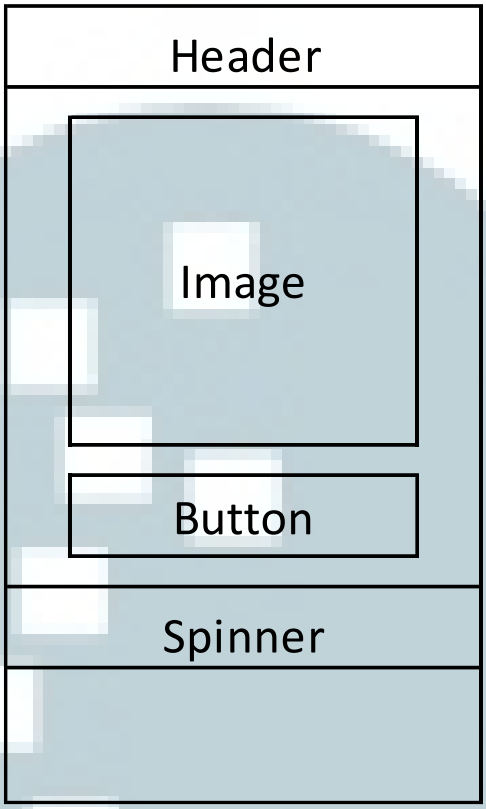
3.4.3 Halaman *Add Template*




Gambar 3.14 Desain halaman *Add Template*

Pada halaman *Add Template* terdapat tiga buah tombol, satu buah *container radio button group* yang berisi tiga buah *radio button*, dan sebuah bingkai gambar atau *image view*. Halaman ini menampilkan informasi gambar yang diinginkan pengguna untuk ditambahkan ke dalam *database template*. Pengguna dapat mengambil gambar melalui kedua tombol pada bagian atas dan men-*submit* gambar tersebut melalui tombol pada bagian bawah.

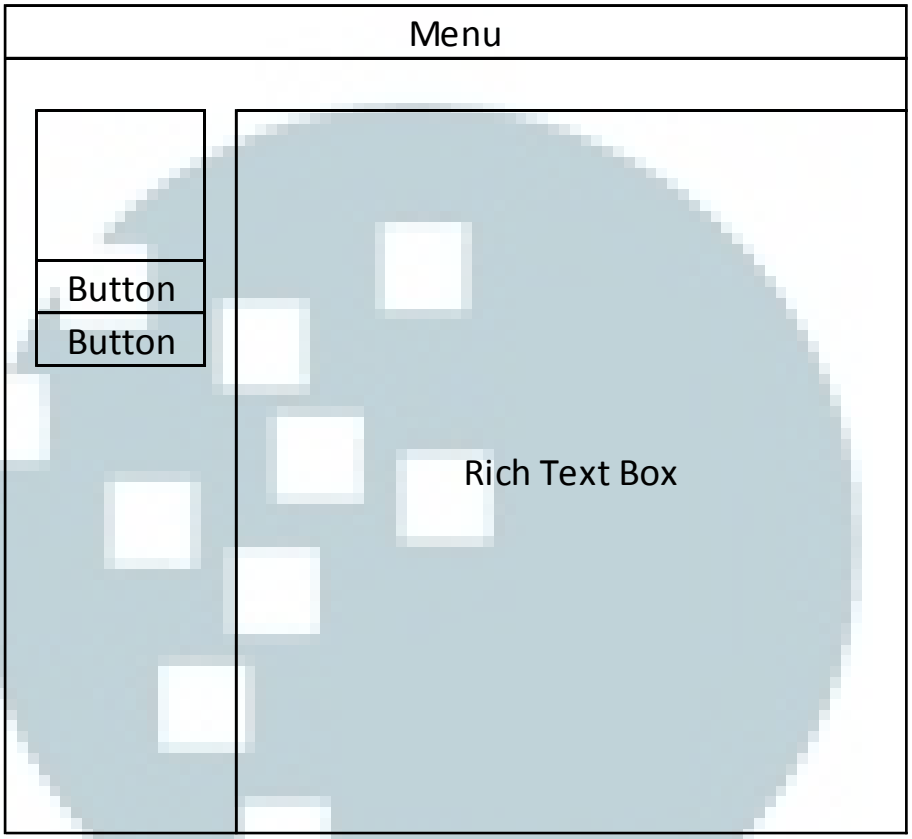
3.4.4 Halaman Browse Image




Gambar 3.15 Desain halaman *Browse Image*

Halaman *Browse Image* terdapat sebuah bingkai gambar atau *image view*, sebuah tombol, dan sebuah menu *spinner*. Halaman ini menampilkan informasi gambar *template* yang dipilih dari bagian menu *Spinner* yang akan ditampilkan ke dalam bingkai gambar dan memberikan opsi menghapus gambar yang dipilih melalui tombol di bawah bingkai gambar.

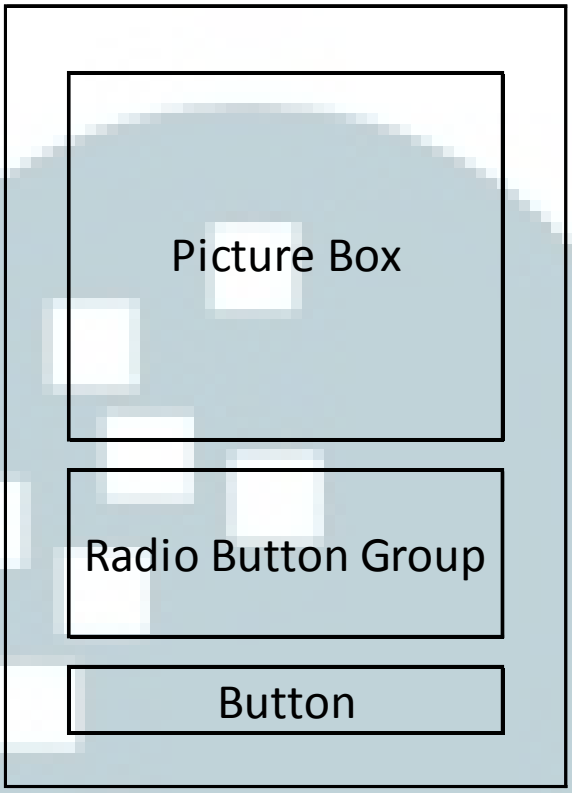
3.4.5 Halaman *Server*




Gambar 3.16 Desain halaman *Server*

Halaman ini berada pada aplikasi *server* yang menggunakan bahasa C#. Halaman *server* terdiri dari sebuah *rich text box* yang memiliki fungsi *scroll* dan menampilkan informasi *server log* pada *text box* tersebut. Selain itu terdapat bingkai gambar atau *picture box* di sebelah atas tombol yang berfungsi sebagai status *Web Service*. Apabila picture box tersebut berwarna merah, status *Web Service* sedang mati, apabila berwarna hijau, status *Web Service* sudah dinyalakan.

3.4.6 Halaman *InputTemplate*




Gambar 3.17 Desain halaman *Input Template*

Halaman *Input Template* dapat diakses melalui halaman utama *server*. Halaman ini terdiri dari satu buah bingkai gambar atau *picture box*, satu buah *container radio button group* yang berisi tiga buah *radio button*, dan sebuah tombol. Halaman ini menampilkan informasi gambar yang dimuat pada bingkai gambar yang kemudian ditambahkan ke *database template*