BAB III
ANALISIS DAN PERANCANGAN SISTEM

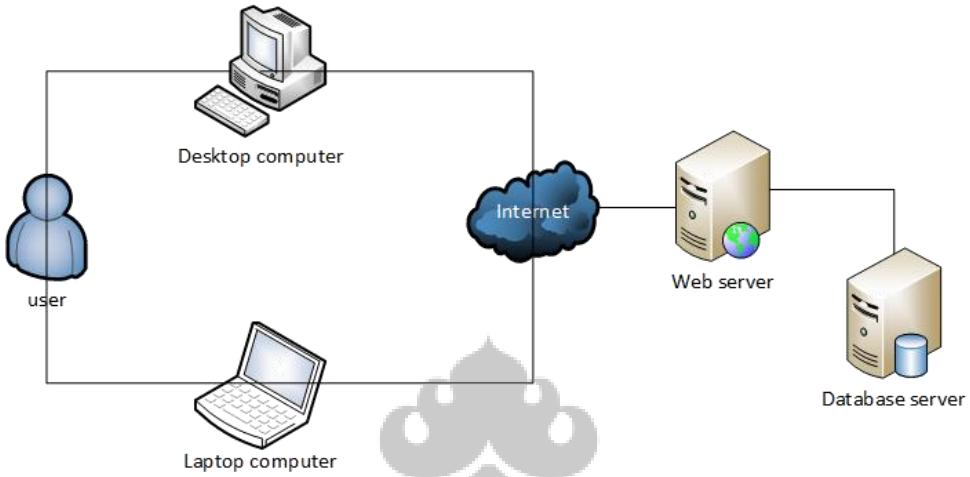
3. 1. Analisis Sistem

Metode analisis sistem yang digunakan dalam perancangan pembuatan aplikasi sistem pakar berbasis Android yaitu berorientasi objek, yang meliputi tahapan perencanaan, analisis perancangan serta implementasi sistem.

3.1.1. Analisis Sistem yang Berjalan

Sebelumnya proses untuk mendiagnosa hama dan penyakit yang terdapat pada tanaman padi masih dilakukan dengan cara konvensional, dalam arti analisa yang dilakukan petani dan diagnosa yang dihasilkan berdasarkan pengalaman, dan insting atau perkiraan. Dari hal tersebut maka validitas hasil diagnosa dapat dikatakan masih rendah. Sementara bagi petani yang telah mulai menerapkan perkembangan teknologi dalam hal ini memanfaatkan sistem pakar yang sudah ada, masih terkendala dengan kebutuhan perangkat keras pendukungnya.

Kebanyakan ditemui sistem pakar yang tersedia masih berbasis *desktop* sehingga agak menyulitkan petani yang kesehariannya bekerja di areal sawah. Sedangkan untuk menjalankan sistem pakar tersebut diperlukan spesifikasi *hardware* berwujud *desktop PC* ataupun *notebook* yang agak membebani baik dari sisi biaya, mobilitas, dan efektifitas. Atau beberapa sistem pakar yang berbasis *web* yang mengharuskan pengguna untuk selalu terhubung dengan *web server* melalui koneksi internet. Hal tersebut tentu sangat membebani bagi petani dari sisi operasional hingga kebutuhan perangkat tertentu seperti modem, atau terlebih lagi masih sulitnya jangkauan sinyal ponsel dan ketersediaan koneksi internet di tengah areal persawahan.

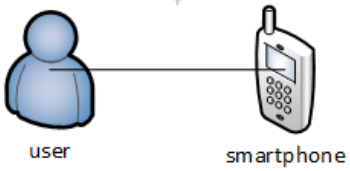


Gambar 3.1. Bagan Sistem yang Berjalan

3.1.2. Analisis Sistem yang Baru

Diperlukan pembuatan *knowledge base* dan *rule base* yang lengkap dan baik agar proses pemecahan masalah ketika mendiagnosa penyakit pada tanaman padi dapat berjalan dengan baik dan memberikan hasil diagnosa yang lebih akurat dibanding sebelumnya. Metode pemecahan masalah pada sistem pakar ini dengan menggunakan sebuah teori matematika *Dempster-Shafer* dengan aturan berdasarkan urutan dan pola tertentu. Pada metode ini menguji gejala sesuai yang dipilih oleh *user* kemudian membandingkan satu demi satu sampai didapat solusi dan nilai kepercayaan yang paling tinggi.

Sistem pakar berbasis Android ini tidak seperti pada sistem pakar kebanyakan yang masih berbasis *desktop* maupun *web*, sehingga dapat digunakan meski tanpa koneksi internet sekalipun. Karena berbasis Android yang memiliki dukungan terhadap basis data SQLite, maka aplikasi ini dapat dijalankan pada *smartphone* berbasis Android sehingga dapat mendukung mobilitas para petani.



Gambar 3.2. Bagan Sistem yang Baru

3.1.3. Analisis SWOT

1.) Kekuatan (*Strength*)

a. Penggunaan *smartphone* makin meningkat dengan didukung dengan harga *smartphone* yang relatif semakin terjangkau.

b. Lebih praktis digunakan jika dibandingkan dengan sistem pakar yang sudah ada sebelumnya yang kebanyakan masih harus dijalankan pada *desktop PC*.

c. Aplikasi dapat digunakan kapan saja dan di mana saja karena berbasis Android sehingga bersifat *mobile*.

d. Tidak memerlukan koneksi internet untuk melakukan diagnosa karena dukungan SQLite *database* oleh sistem operasi Android.

2.) Kelemahan (*Weakness*)

a. Gejala yang ada masih merupakan gejala yang umum ditemui.

b. Untuk penanganan lebih lanjut disarankan untuk tetap melakukan pemeriksaan mendalam.

c. *User* tidak dapat merubah aturan inferensi yang ada karena sudah ditentukan sejak awal berdasarkan *transfer knowledge* dari pakar.

d. Tingkat validitas bergantung pada pemberian nilai kepercayaan pada masing-masing gejala oleh para pakar.

e. Karena berbasis Android aplikasi ini kurang sesuai digunakan oleh lanjut usia yang kurang familiar dengan penggunaan *smartphone*.

3.) Kesempatan (*Opportunity*)

a. Aplikasi sistem pakar ini memudahkan *user* untuk melakukan diagnosa secara mandiri.

b. Menambah pengetahuan *user* dalam mengetahui gejala dan penyakit pada tanaman padi.

4.) Ancaman (*Threat*)

a. Dikhawatirkan ke depannya ketika *smartphone* berbasis Android mulai ditinggalkan atau tergantikan sistem operasi lainnya.

3.1.4. Analisis Kebutuhan Sistem

Analisis kebutuhan sistem merupakan tahap penting dalam pembangunan suatu sistem yang terdiri dari kebutuhan fungsional yang merupakan pernyataan mengenai apa saja yang seharusnya dikerjakan oleh sebuah sistem dan secara umum menggambarkan layanan atau fitur yang terdapat di dalam sebuah sistemdan kebutuhan non-fungsional yaitu pernyataan yang berkaitan dengan properti dan pembentuk sistem tersebut.

3.1.4.1. Kebutuhan Fungsional

Kebutuhan fungsional pada sistem pakar ini yaitu:

1) Sistem menyediakan informasi mengenai jenis penyakit pada tanaman padi.
2) Sistem menyediakan informasi mengenai gejala yang terdapat pada tanaman padi.
3) Sistem menyediakan fasilitas untuk diagnosa penyakit pada tanaman padi.
4) Sistem menampilkan hasil diagosa penyakit.

3.1.4.2. Kebutuhan Non-Fungsional

Sistem pakar ini dibangun menggunakan bahasa pemrograman Java dengan Android SDK dan SQLite sebagai basis data.

3. 2. Perancangan Sistem

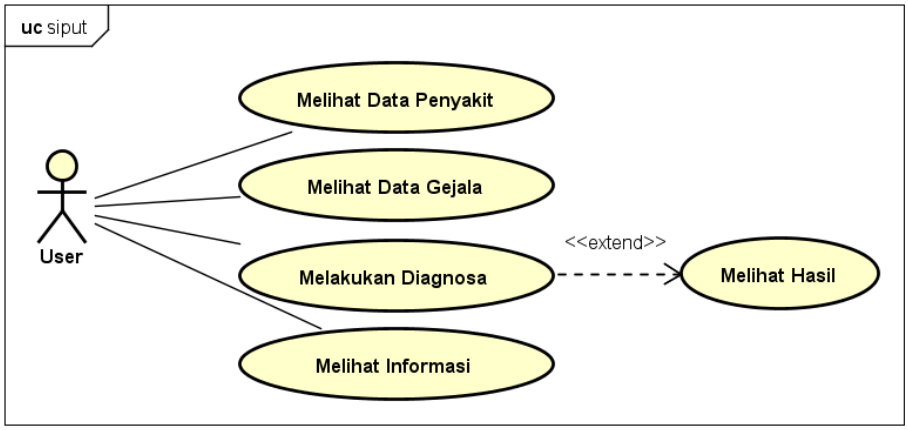
3.2.1. *Use Case Diagram*

Use case digunakan untuk memodelkan dan menyatakan unit fungsi yang disediakan sistem ke *user*. *Use case* dilingkupi dengan batasan sistem yang diberikan label nama sistem.

Tabel 3.1. *Use Case Requirement*

No	*Requirement*	*Actor*	*Use Case*
1	*User* melihat data penyakit	*User*	Melihat penyakit
2	*User* melihat data gejala	*User*	Melihat gejala
3	*User* melakukan diagnosa	*User*	Melakukan diagnosa
4	*User* melihat hasil diagnosa	*User*	Melihat hasil
5	*User* melihat informasi aplikasi	*User*	Melihat informasi

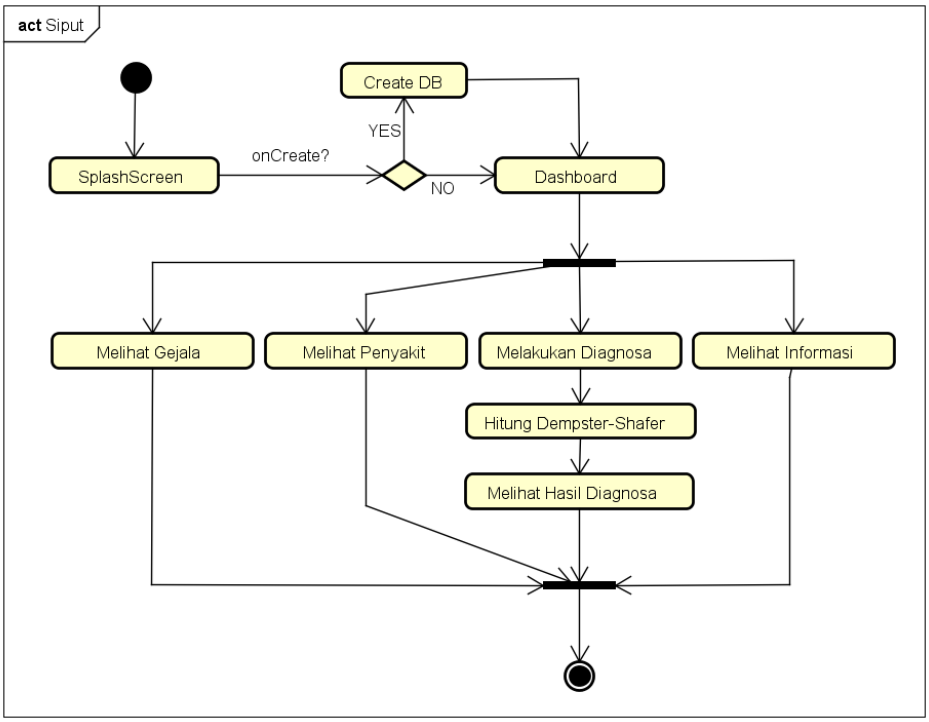
Use case diagram digunakan untuk mendapatkan persyaratan atau kebutuhan sistem (*requirements*) dan menggambarkan hubungan antara sistem dengan *user*.



Gambar 3.3. Use Case Diagram

3.2.2. *Activity Diagram*

Diagram ini menggambarkan detail aliran fungsionalitas sistem pakar dan merupakan sebuah cara untuk memodelkan aliran kerja (*work flow*) maupun menggambarkan aliran kejadian (*flow of event*) dari sebuah sistem.

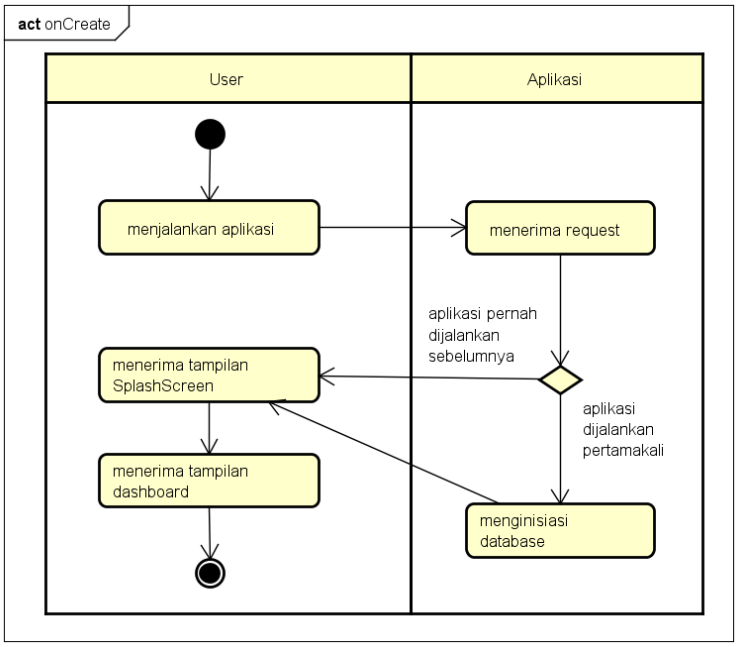


Gambar 3.4. Activity Diagram Aplikasi

Activity diagram digunakan untuk menggambarkan hubungan aliran kerja *business* terlepas dari *classes*, aliran *activities* dalam *use case* atau detail desain dari *method*.

3.2.2.1. *Activity Diagram onCreate*

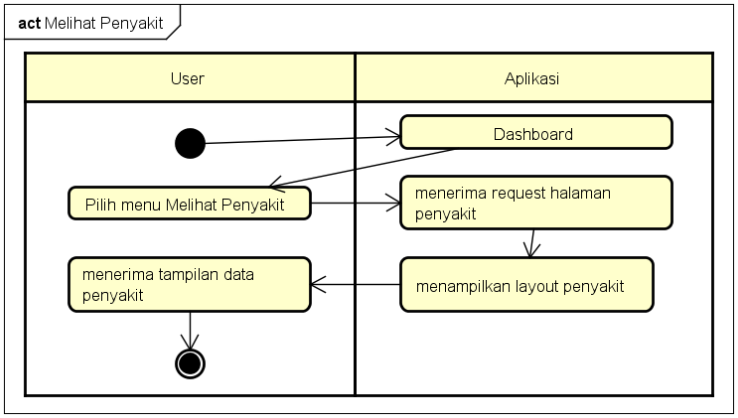
Diawali oleh *user* dengan menjalankan aplikasi yang kemudian direspon apabila aplikasi tersebut baru pertama kali dijalankan maka akan dilakukan inisiasi *database*, yaitu membuat sebuah SQLite *database* di perangkat *smartphone*. Setelah berhasil menginisiasi *database* maka akan ditampilkan *layout splashscreen* yang dilanjutkan dengan menampilkan *layout dashboard*.



Gambar 3.5. Activity Diagram onCreate

3.2.2.2. *Activity Diagram* Melihat Penyakit

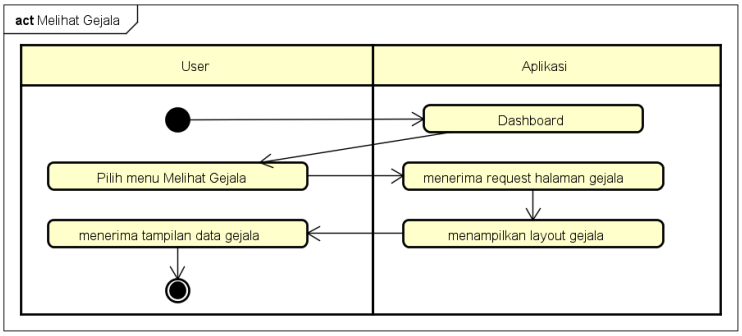
User melihat tampilan *dashboard* yang berisi menu dari aplikasi, kemudian memilih menu Melihat Penyakit yang oleh aplikasi akan direspon dengan cara menampilkan halaman *layout* penyakit yang berisikan data-data penyakit.



Gambar 3.6. Activity Diagram Melihat Penyakit

3.2.2.3. *Activity Diagram* Melihat Gejala

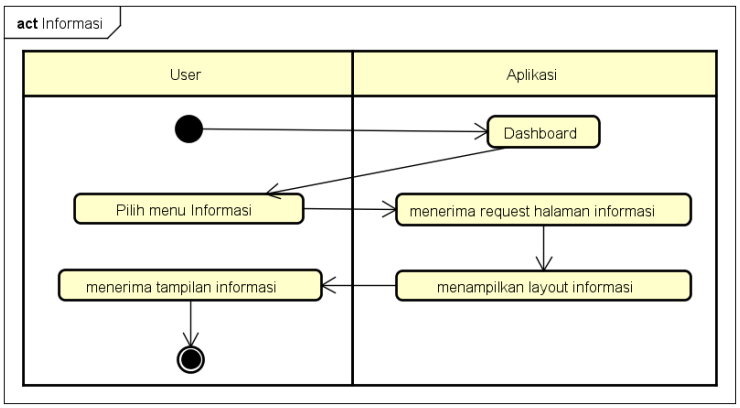
User melihat tampilan *dashboard* yang berisi menu dari aplikasi, kemudian memilih menu Melihat Gejala yang kemudian oleh aplikasi akan direspon dengan cara menampilkan halaman *layout* gejala yang berisikan data-data gejala.



Gambar 3.7. Activity Diagram Melihat Gejala

3.2.2.4. *Activity Diagram* Melihat Informasi

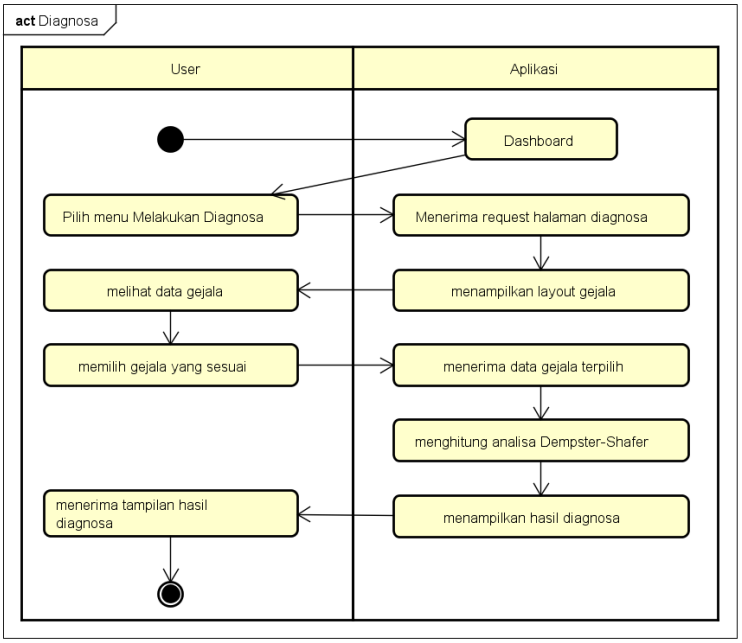
User melihat tampilan *dashboard* yang berisi menu dari aplikasi, kemudian memilih menu Melihat Informasi yang kemudian oleh aplikasi akan direspon dengan cara menampilkan halaman *layout* gejala yang berisikan informasi mengenai aplikasi sistem pakar.



Gambar 3.8. Activity Diagram Melihat Informasi

3.2.2.5. *Activity Diagram* Melakukan Diagnosa

User melihat tampilan *dashboard* yang berisi menu dari aplikasi, kemudian memilih menu Melakukan Diagnosa yang kemudian oleh aplikasi akan direspon dengan cara menampilkan halaman layout diagnosa yang berisikan data-data gejala. Setelah itu user memilih gejala apa saja kemudian melanjutkan proses diagnosa dengan metode *Dempster-Shafer* hingga ditemukan penyakit dengan nilai keyakinan tertinggi. Hasil tersebut ditampilkan melalui layout hasil diagnosa.



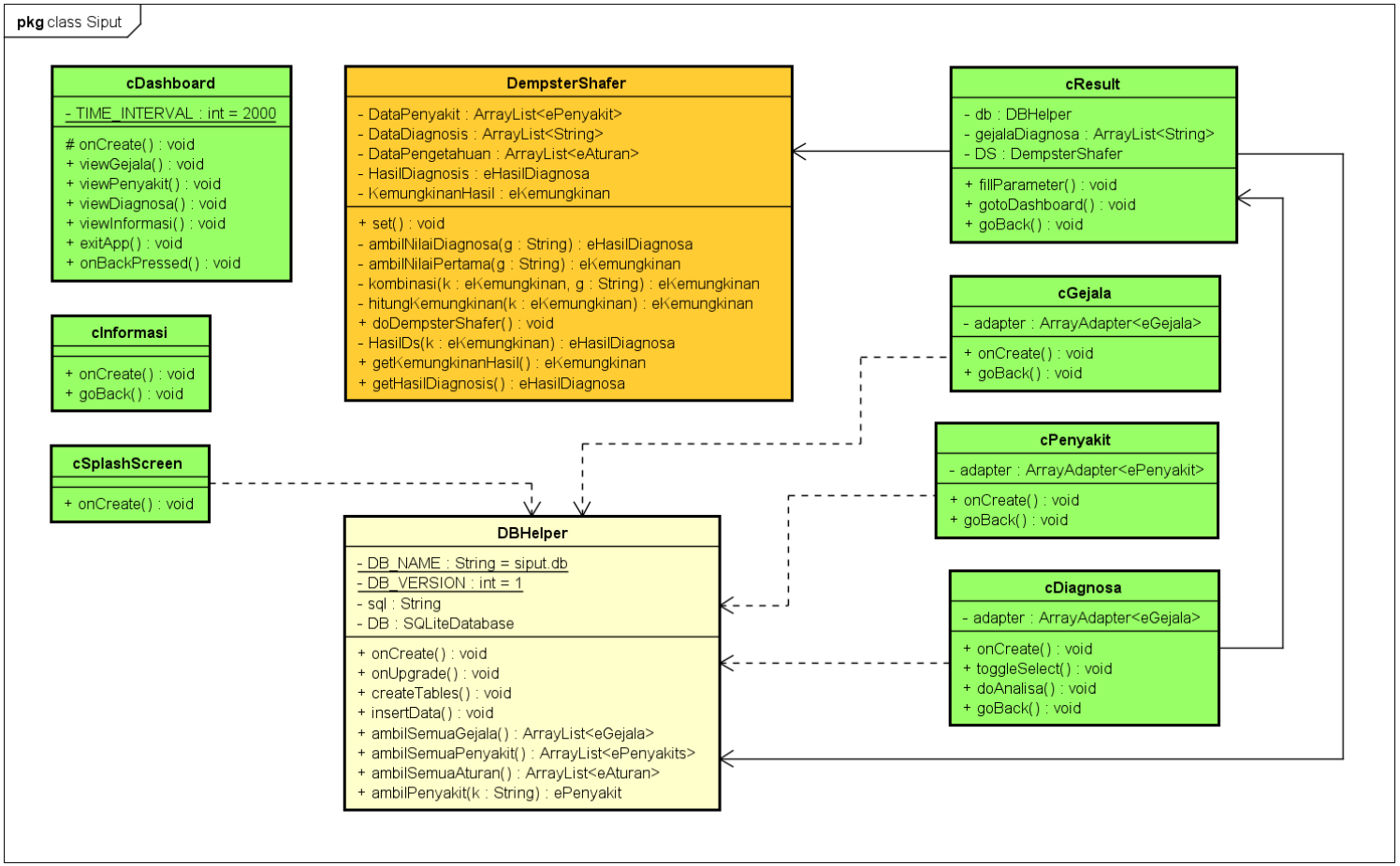
Gambar 3.9. Activity Diagram Melakukan Diagnosa

3.2.3. *Class Diagram*

Class diagram merupakan diagram yang menunjukkan kebutuhan *class-class* dalam suatu sistem, dimana *class* tersebut mengandung suatu *attributes* dan *operations / methods* yang diperlukan. *Class* juga merupakan pembentuk utama dari sistem dengan model *Object Oriented* karena *class* menunjukkan kumpulan obyek yang memiliki atribut dan operasi yang sama. Diagram ini menggambarkan hubungan antara model *class* di dalam sistem.

Terdapat satu *class* utama dalam sistem ini yang bernama *class* DempsterShafer yang menangani perhitungan dengan metode DST (*Dempster-Shafer Theory*). Pada *class* DempsterShafer memiliki ketergantungan terhadap beberapa *class* lain di antaranya adalah *class* eAturan, eGejala, eKemungkinan, ePenyakit dan eHasilDiagnosa yang merupakan POJO (*Plain Old Java Object*) atau enkapsulasi dari objek-objek.

Selain *class* DempsterShafer ada lagi *class* yang memuat koneksi terhadap *database* SQLite serta menangani pengambilan data. *Class* lainnya merupakan *controller class* yang menangani *behaviour* atau perilaku dari masing-masing *use case*.

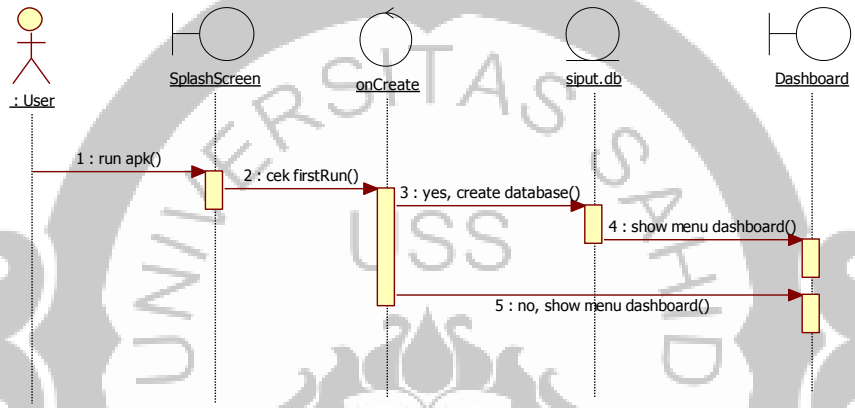


Gambar 3.10. Class Diagram Sistem Pakar

3.2.4. *Sequence Diagram*

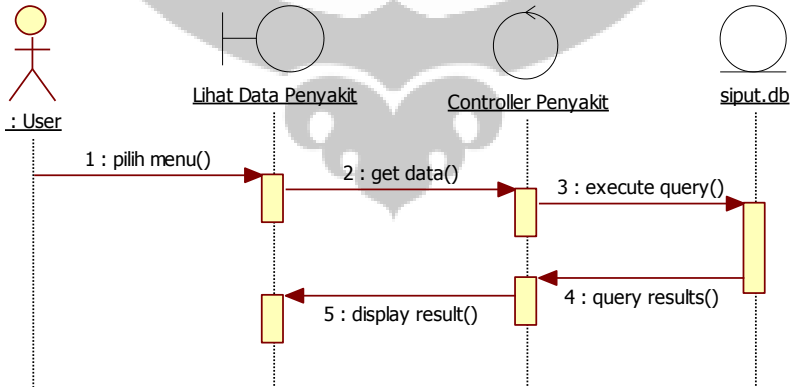
Diagram ini menjelaskan interaksi obyek yang disusun dalam urutan waktu. Dalam *sequence diagram* terdapat *stereotype* antara *boundary* untuk mendefinisikan obyek GUI (*Graphical User Interface*), *class control* untuk mengontrol logika program dan *entity* yang mewakili penyimpanan data. Obyek-obyek tersebut kemudian diurutkan dari kiri ke kanan, aktor yang menginisiasi interaksi biasanya diletakkan di paling kiri dari diagram.

Sequence Diagram onCreate



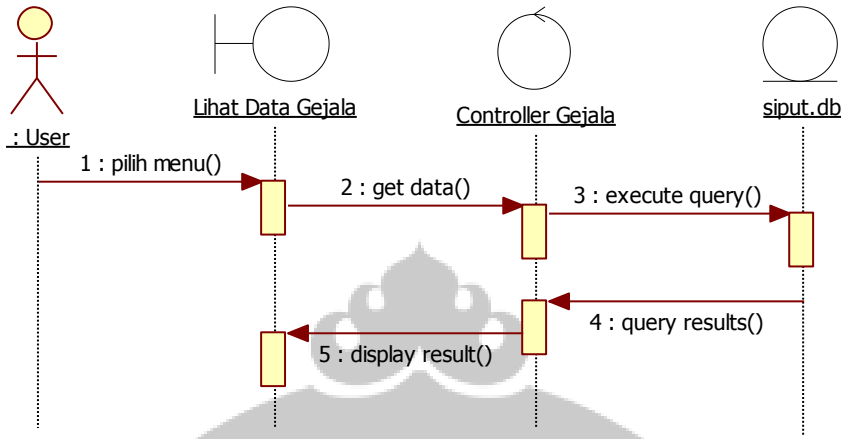
Gambar 3. 11. Sequence Diagram onCreate

Sequence Diagram menu Lihat Data Penyakit



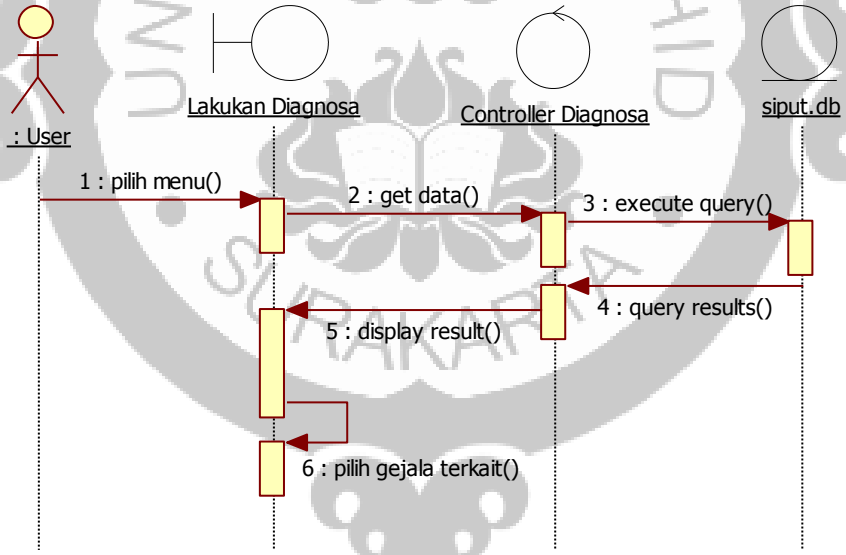
Gambar 3.12. Sequence Diagram Menu Lihat Data Penyakit

Sequence Diagram menu Lihat Data Gejala



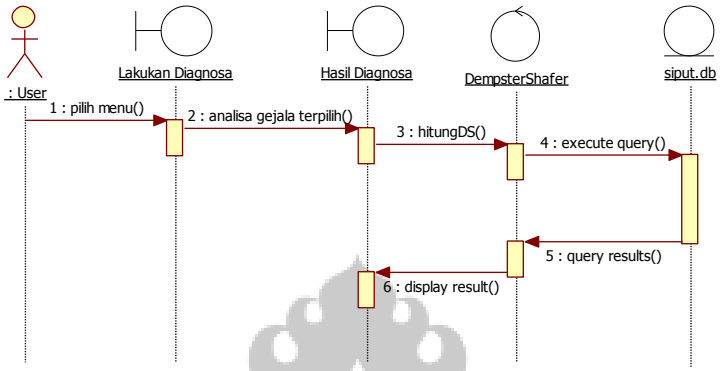
Gambar 3.13. Sequence Diagram Menu Lihat Data Gejala

Sequence Diagram menu Melakukan Diagnosa



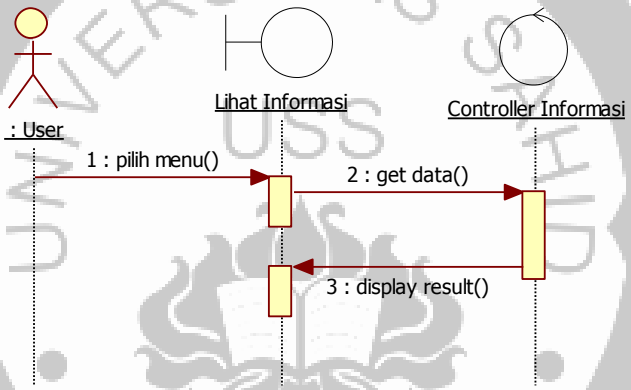
Gambar 3.14. Sequence Diagram Melakukan Diagnosa

Sequence Diagram untuk hasil diagnosa



Gambar 3.15. Sequence Diagram Hasil Diagnosa

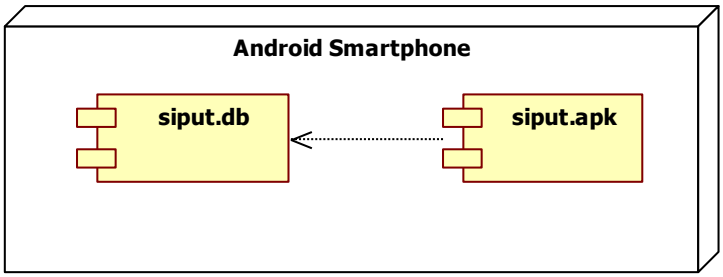
Sequence Diagram menu Informasi



Gambar 3.16. Sequence Diagram Menu Informasi

3.2.5. *Deployment Diagram*

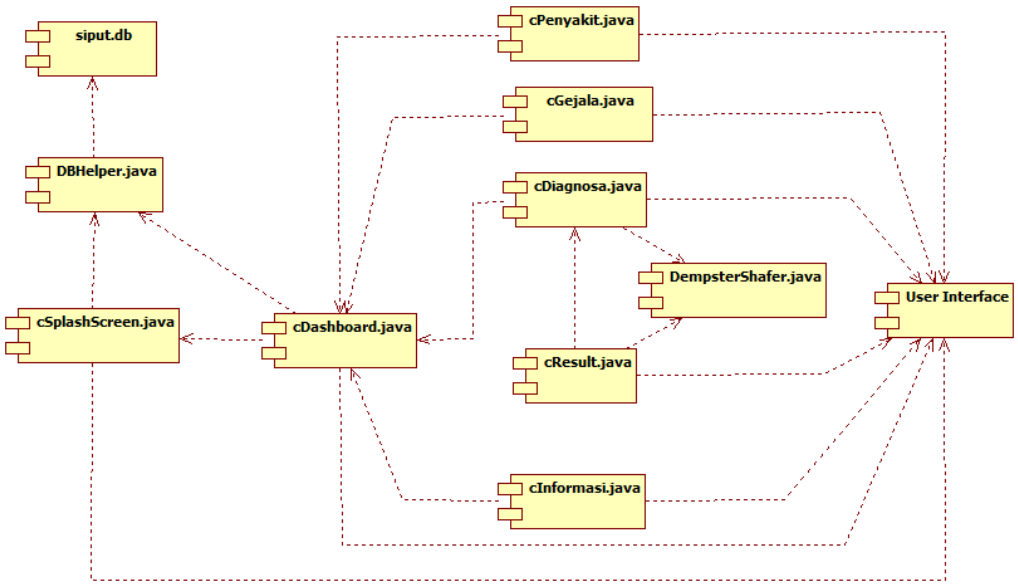
Jika diartikan ke dalam bahasa Indonesia, maka *Deployment Diagram* adalah diagram pendistribusian yang menggambarkan arsitektur fisik dari sebuah sistem. Sesuai dengan kebutuhan sistem yang diterapkan di mana sistem membutuhkan perangkat smartphone yang berbasis Android dengan dukungan SQLite sehingga tidak diperlukan perangkat lain untuk menjalankan sistem.



Gambar 3.17.Deployment Diagram

3.2.6. *Component Diagram*

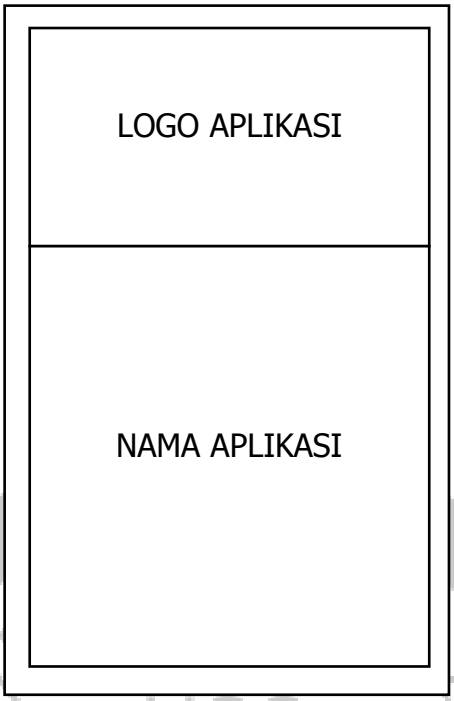
Component diagram menggambarkan struktur antara komponen perangkat lunak termasuk ketergantungan satu dengan lainnya. Diagram komponen fokus pada komponen sistem yang dibutuhkan dan ada di dalam sistem.



Gambar 3.18. Component Diagram

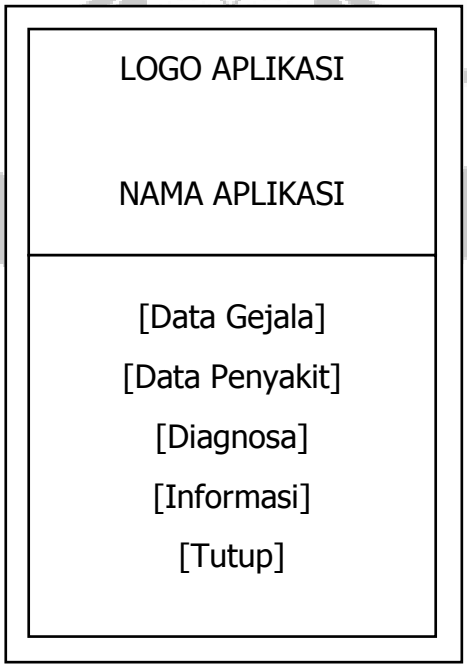
3.2.7. Perancangan Layout Aplikasi

3.2.7.1. Tampilan Awal (*SplashScreen*)



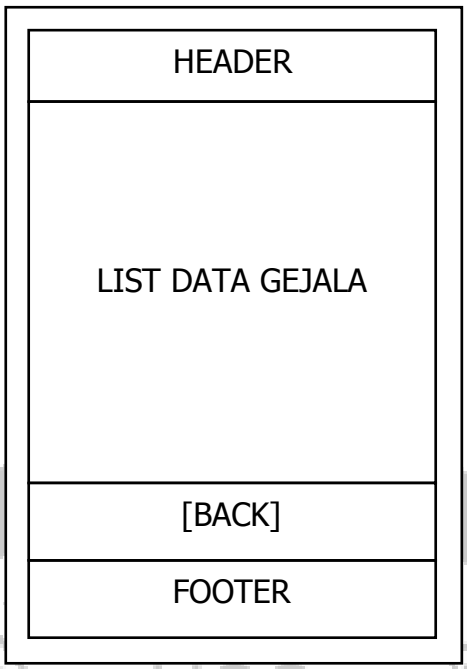
Gambar 3. 19. Layout Splash Screen

3.2.7.2. Tampilan Menu (*Dashboard*)



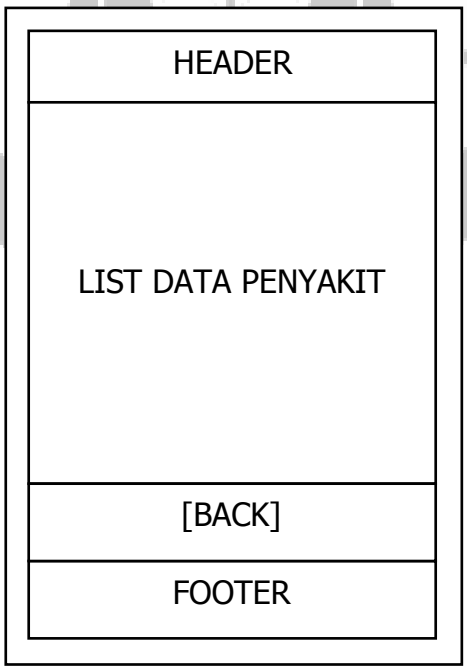
Gambar 3. 20. Layout Dashboard

3.2.7.3. Tampilan untuk Melihat Data Gejala



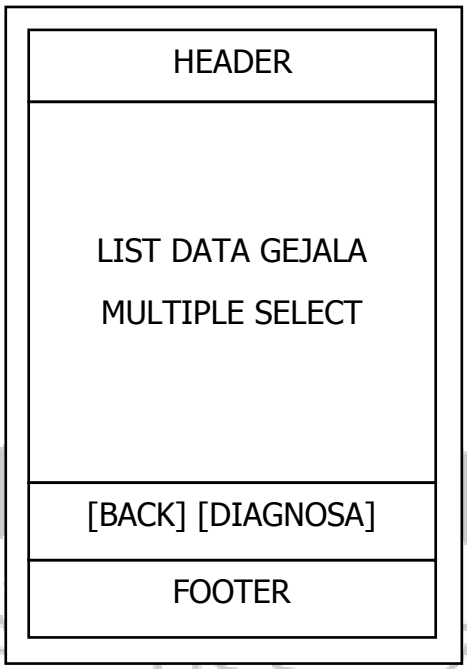
Gambar 3. 21. Layout Data Gejala

3.2.7.4. Tampilan untuk Melihat Data Penyakit



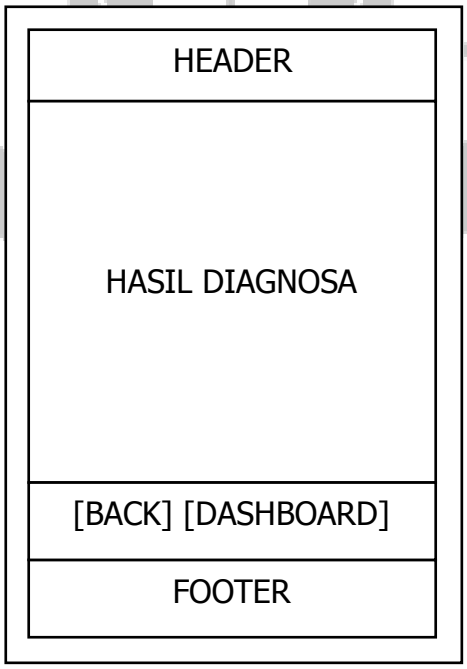
Gambar 3. 22. Layout Data Penyakit

3.2.7.5. Tampilan untuk Melakukan Diagnosa



Gambar 3. 23. Layout Diagnosa

3.2.7.6. Tampilan untuk Hasil Diagnosa



Gambar 3. 24. Layout Hasil Diagnosa

3.2.7.7. Tampilan untuk Informasi

Gambar 3. 25. Layout Informasi