BAB IV
IMPLEMENTASI

4.1. Implementasi

Proses pengimplementasian sistem dilakukan berdasarkan perancangan sistem yang telah dijelaskan pada bab sebelumnya. Pengimplementasian sistem akan menjelaskan tentang spesifikasi lingkungan pengembangan sistem, batasan implementasi, implementasi basis data, implementasi *web service*, dan implementasi sistem.

4.1.1. Spesifikasi Lingkungan Pengembangan sistem

Sistem Buku Saku Perawat akan dikembangkan pada lingkungan implementasi sebagai berikut:

1. Spesifikasi Perangkat Keras

Spsesifikasi perangkat keras yang akan digunakan untuk pengimplementasian sistem akan dijelaskan pada Tabel 4.1. dan Tabel 4.2.

Tabel 4.1. Spesifikasi perangkat keras pengembangan sistem

Nama Komponen	Spesifikasi
System Model	ASUS N43SL
Processor	Intel(R) Core(TM) i5-2430M CPU @ 2.40GHz (4 CPUs), ~2.4GHz
Memory	4096 MB
Harddisk	750 GB
Display	NVIDIA GeForce GT540M CUDA 2 GB

Tabel 4.2. Spesifikasi perangkat keras instalasi dan pengujian sistem

Nama Komponen	Spesifikasi
System Model	SONY XPERIA Sola MT27i
Processor	1 GHz STE U8500 Dual Core
Memory Internal	8 GB
Memory RAM	512 MB
Display	TFT *capacitive touchscreen*, 16 *million colours* 854 x 480 *pixels*, 3.7 *inches*

2. Spesifikasi Perangkat Lunak

Spsesifikasi perangkat lunak yang akan digunakan untuk pengimplementasian sistem akan dijelaskan pada Tabel 4.3. dan Tabel 4.4.

Tabel 4.3. Spesifikasi perangkat lunak pengembangan sistem

Nama Komponen	Spesifikasi
Operating System	Windows 7 Home Premium 64-bit Service Pack 1
Programming Language	- PHP - Java
Text Editor / IDE	- Sublime Text 2 - Eclipse Indigo
Server	Apace HTTP Server (Localhost)
Database	MySQL

Tabel 4.4. Spesifikasi perangkat lunak instalasi dan pengujian sistem

Nama Komponen	Spesifikasi
Operating System	Android 4.0.4 (*Ice Cream Sandwich*)

4.1.2. Batasan Implementasi

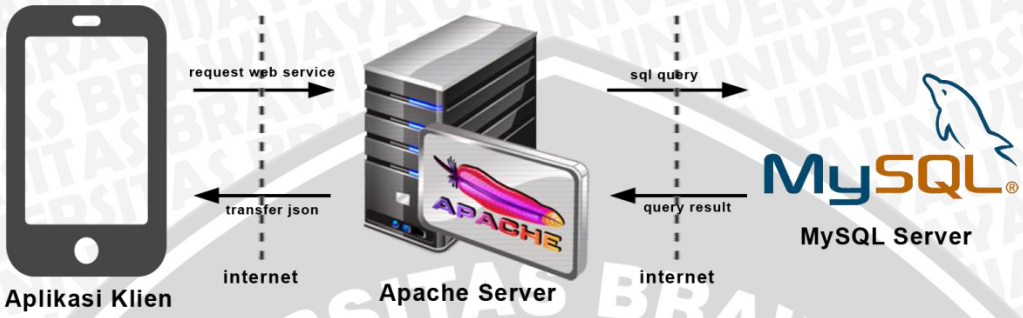
Berikut ini merupakan batasan implementasi sistem buku saku perawat:

1. Data yang digunakan dalam sistem diambil dari referensi buku keperawatan, yaitu Buku Perawatan Kritis [JJN-09] dan Buku Anatomi & Fisiologi untuk Paramedis [ECP-10], dan halaman web www.mims.com.
2. Untuk mengakses catatan dan profil, pengguna harus memiliki akun dan *login* terlebih dahulu. Sedangkan untuk mengakses konten keperawatan, pengguna tidak harus memiliki akun atau *login* terlebih dahulu.
3. Untuk mengakses sistem diperlukan koneksi internet, sehingga kecepatan akses untuk menampilkan data bergantung pada kecepatan koneksi internet masing-masing perangkat yang digunakan.

4.1.3. Implementasi Arsitektural

Implementasi arsitektural sistem dilakukan dengan mengimplementasikan rancangan arsitektural yang telah dijelaskan pada bab sebelumnya. Berikut ini

merupakan implementasi arsitektural sistem yang akan dijelaskan pada Gambar 4.1.



Gambar 4.1. Implementasi arsitektural

Gambar 4.1. menjelaskan kinerja sistem secara keseluruhan, dimana sistem dimulai dengan aplikasi klien yang melakukan *request web service* pada Apache *server* berdasarkan fitur yang dipilih oleh pengguna. *Web service* akan melakukan pencarian data pada MySQL *server* menggunakan *sql query*. Kemudian MySQL *server* akan merespon *query* dengan mengirimkan hasil dari *query* tersebut ke *web service*. *Web service* akan mengubah data tersebut ke format JSON dan akan mengirimkannya pada aplikasi klien.

Berikut ini merupakan spesifikasi perangkat yang digunakan pada arsitektur sistem yang dijelaskan pada Tabel 4.5.

Tabel 4.5. Spesifikasi perangkat lunak arsitektur sistem

Nama Komponen	Spesifikasi
Operating System	Android 4.0.3 (*Ice Cream Sandwich*) keatas
Web Server	Apache 2.2.14
Database Server	MySQL 5.1.41

4.1.4. Implementasi Antarmuka

Implementasi antarmuka dilakukan dengan mengimplementasikan rancangan antarmuka sistem yang telah dibuat pada bab sebelumnya.

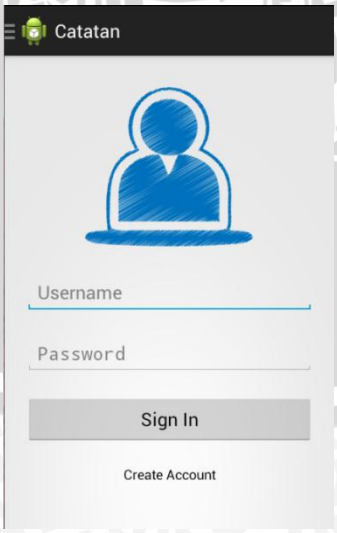
4.1.4.1. Antarmuka Beranda

Berikut ini merupakan implementasi antarmuka beranda yang akan dijelaskan pada Gambar 4.2.

Gambar 4.2. Antarmuka beranda

4.1.4.2. Antarmuka Login

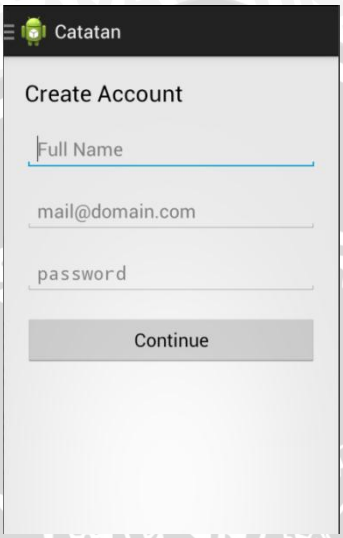
Berikut ini merupakan implementasi antarmuka *login* yang akan dijelaskan pada Gambar 4.3.



Gambar 4.3. Antarmuka *login*

4.1.4.3. Antarmuka Registrasi

Berikut ini merupakan implementasi antarmuka registrasi akun yang akan dijelaskan pada Gambar 4.4.



Gambar 4.4. Antarmuka registrasi

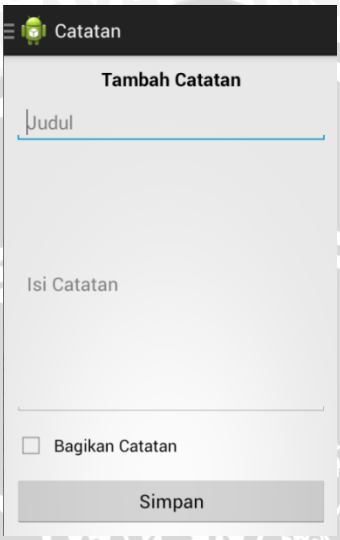
4.1.4.4. Antarmuka Lihat Catatan

Berikut ini merupakan implementasi antarmuka lihat catatan yang akan dijelaskan pada Gambar 4.5.

Gambar 4.5. Antarmuka lihat catatan

4.1.4.5. Antarmuka Tambah Catatan

Berikut ini merupakan implementasi antarmuka tambah catatan yang akan dijelaskan pada Gambar 4.6.



Gambar 4.6. Antarmuka tambah catatan

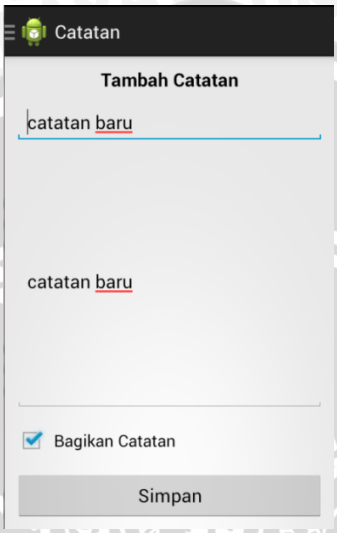
4.1.4.6. Antarmuka Lihat Detail Catatan

Berikut ini merupakan implementasi antarmuka lihat detail catatan yang akan dijelaskan pada Gambar 4.7.

Gambar 4.7. Antarmuka lihat detail catatan

4.1.4.7. Antarmuka Update Catatan

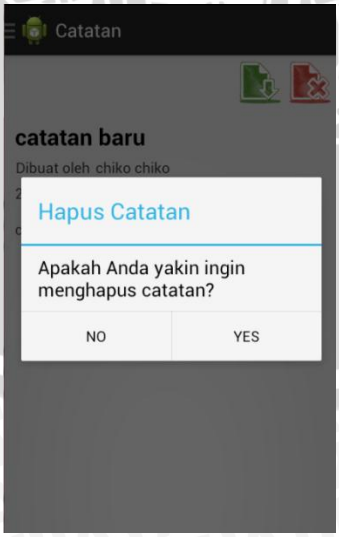
Berikut ini merupakan implementasi antarmuka *update* catatan yang akan dijelaskan pada Gambar 4.8.



Gambar 4.8. Antarmuka *update* catatan

4.1.4.8. Antarmuka Hapus Catatan

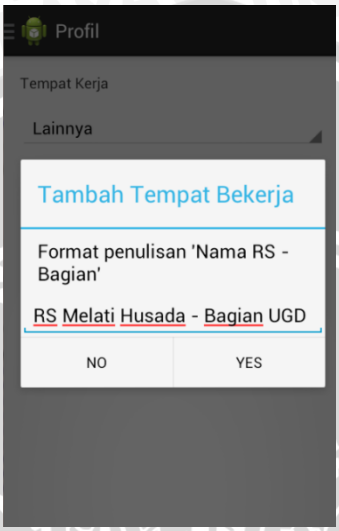
Berikut ini merupakan implementasi antarmuka hapus catatan yang akan dijelaskan pada Gambar 4.9.



Gambar 4.9. Antarmuka hapus catatan

4.1.4.9. Antarmuka Tambah Tempat Bekerja

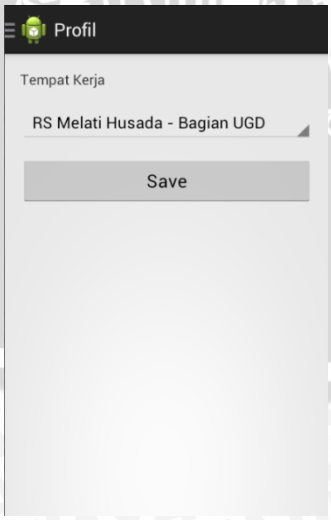
Berikut ini merupakan implementasi antarmuka tambah tempat bekerja yang akan dijelaskan pada Gambar 4.10.



Gambar 4.10. Antarmuka tambah tempat bekerja

4.1.4.10. Antarmuka Pilih Tempat Bekerja

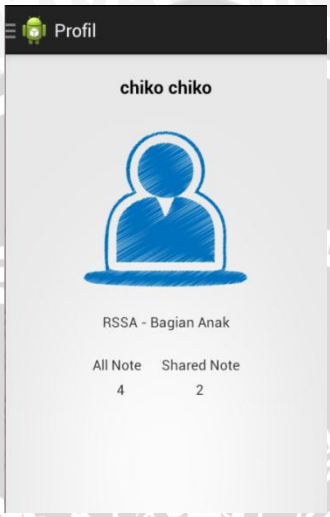
Berikut ini merupakan implementasi antarmuka pilih tempat bekerja yang akan dijelaskan pada Gambar 4.11.



Gambar 4.11. Antarmuka pilih tempat bekerja

4.1.4.11. Antarmuka Lihat Profil

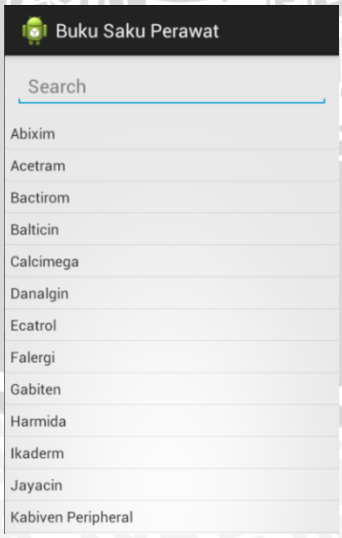
Berikut ini merupakan implementasi antarmuka lihat profil yang akan dijelaskan pada Gambar 4.12.



Gambar 4.12. Antarmuka lihat profil

4.1.4.12. Antarmuka Lihat Konten Keperawatan

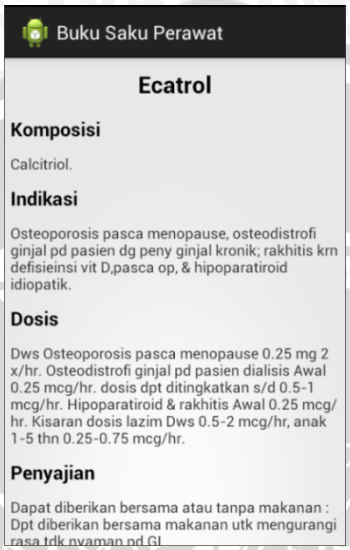
Berikut ini merupakan implementasi antarmuka lihat konten keperawatan yang akan dijelaskan pada Gambar 4.13.



Gambar 4.13. Antarmuka lihat konten keperawatan

4.1.4.13. Antarmuka Lihat Detail Konten Keperawatan

Berikut ini merupakan implementasi antarmuka lihat detail konten keperawatan yang akan dijelaskan pada Gambar 4.14.



Gambar 4.14. Antarmuka lihat detail konten keperawatan

4.1.5. Implementasi Basis Data

Sistem menggunakan basis data untuk menyimpan data yang dibutuhkan oleh sistem. Pada sistem ini, basis data terdiri dari 6 tabel yang digunakan untuk menyimpan data pengguna, catatan, tempat bekerja, obat, penyakit, dan anatomi.

Berikut ini merupakan implementasi basis data yang digunakan untuk menyimpan data pengguna yang dijelaskan pada Tabel 4.6. dan Gambar 4.15.

Tabel 4.6. Tabel *users*

Nama Kolom	Tipe Data	Keterangan	Deskripsi
uid	int	Primary key	Id pengguna (1, 2, 3, …)
id_kerja	int	Foreign Key	Id tempat bekerja pengguna (1, 2, 3, …)
name	varchar	-	Nama pengguna (aku, …)
email	varchar	-	Email pengguna (aku@gmail.com, …)
encrypted_password	varchar	-	Password pengguna yang telah dienkripsi (fT8/+l/cry

Nama Kolom	Tipe Data	Keterangan	Deskripsi
			1e5v2ZuThLpw3Y7LEyZjgxY2UzZTkx, …)
create_at	datetime	-	Tanggal dan jam pembuatan akun (2014-03-13 19:21:31, …)
update_at	datetime	-	Tanggal dan jam pembaruan akun (2014-03-17 21:34:49, …)

uid	id_kerja	name	email	encrypted_password	created_at	updated_at
1	2	chiko chiko	chiko@gmail.com	4qibyUeaNCN3oPS78O+1qeTUPzliMWRhMDEwODY4	2014-03-13 19:21:31	2014-05-22 13:31:03
2	0	Full Name	mail@domain.com	RGgg3/NOqqve8mhLrjjdfR8POdswNmQxYjNjN2Rh	2014-03-17 21:34:49	2014-03-17 21:34:49
3	3	intan ayu laksmi	intan@gmail.com	/4TlLoiWqKLY0sQLyLAmpK6x57tjMjRlZjVjNjM2	2014-04-25 10:19:27	2014-05-26 12:29:19
4	5	taufan dwi	taufan@gmail.com	6SXQZaW2B1KW0hdvD1AJQgsSIDlmYTU0YTc5YWZi	2014-04-25 20:29:01	2014-05-27 16:03:34
5	0	gentolet	gentolet@gmail.com	S5n5W53OwQ/+k3dFNtDblFy7qWYxNWI1MWYzNWIw	2014-04-27 11:36:15	2014-04-27 11:36:15
6	2	bambang	bambang@gmail.com	qFMTQMGkRHtiLDUmMC5JadGdwfl4NGJjMTY4NmEw	2014-04-27 11:51:12	2014-05-27 16:15:14
7	0	bambang	bambang gentolet	ZzIP9qkI7xOOhxMX/LbAmM06gHc5OTRmMzI3MTIm	2014-04-27 13:13:37	2014-04-27 13:13:37
8	2	mpuss chiko	mpuss@gmail.com	4yTK+8Okk5uA9qTwC3B2FyGET6IzODRjNjZlYzUx	2014-04-27 15:21:46	2014-05-27 16:34:24
9	5	muke muke	muke@gmail.com	fT8/+I/cry1e5v2ZuThLpw3Y7LEyZjgxY2UzZTkx	2014-04-27 21:41:33	2014-05-28 08:17:36
10	2	mister xyz	xyz@gmail.com	EFtop8QLb2VnYvl9T8ZqrsgUYwVhMjgxNDM1NjA4	2014-04-27 21:56:13	2014-05-28 08:38:28

Gambar 4.15. Implementasi tabel *users*

Berikut ini merupakan implementasi basis data yang digunakan untuk menyimpan data catatan yang dijelaskan pada Tabel 4.7. dan Gambar 4.16.

Tabel 4.7. Tabel *note*

Nama Kolom	Tipe Data	Keterangan	Deskripsi
id_note	int	Primary key	Id catatan (1, 2, 3, …)
id_user	int	Foreign key	Id pengguna (1, 2, 3, …)
id_kerja	int	Foreign key	Id tempat bekerja pengguna (1, 2, 3, …)
judul_note	varchar	-	Judul catatan (judul, …)
tanggal	timestamp	-	Tanggal catatan dibuat (2014-03-17 21:34:49, …)
isi_note	varchar	-	Isi catatan (isi, …)
is_share	int	-	Status pembagian catatan (1 atau 0)

id_user	id_kerja	id_note	judul_note	tanggal	isi_note	is_share
0	0	1	catatan taufan	2014-04-30 22:32:33	taufan dwi putra setiawan	0
8	0	18	coba coba	2014-05-23 14:49:52	coba coba	0
8	0	3	mpuss chiko baru	2014-04-30 23:07:41	mpuss chiko baru	0
8	0	14	bambang gentolet	2014-05-06 15:05:19	gentolelolelolet	0
0	0	5	kolonel chiko	2014-05-01 11:28:53	kolonel chiko	0
8	0	6	kolonel	2014-05-01 11:46:22	kolonel	0
8	0	7	gentolet	2014-05-01 11:55:44	bambang gentolet	0
8	0	8	gentolet	2014-05-01 11:56:21	bambang gentolet	0
8	0	9	qwerty	2014-05-01 12:10:09	qwerty uiop	0
8	0	10	qwerty	2014-05-01 12:17:15	qwerty uiop	0
4	0	11	bambang	2014-05-01 12:20:24	bambang	0
1	0	12	judul_note	2014-05-01 15:57:39	isi_note	0
1	0	13	tess terus	2014-05-01 16:11:53	tesss terus	1
11	0	15	gentolet	2014-05-12 20:14:09	gentolet	0
1	2	16	tesssku	2014-05-22 14:05:44	tesss	0
14	0	19	catatan1	2014-05-23 15:01:19	catatan1	0
14	0	20	catatan2	2014-05-23 15:02:06	catatan2	0
8	2	23	tanpa dibagi	2014-05-27 20:40:05	tanpa dibagi	0
8	2	22	chiko chiko chiko	2014-05-27 20:25:01	qewrtyyuoripyt asedreyUPTORI fsdwcgjhdsmfbvdj,sgb ...	1
4	5	24	rs hermina	2014-05-28 10:11:15	rs hermina	1
1	2	25	judul coba	2014-06-04 12:30:39	isi coba	1

Gambar 4.16. Implementasi tabel *note*

Berikut ini merupakan implementasi basis data yang digunakan untuk menyimpan data pekerjaan yang dijelaskan pada Tabel 4.8. dan Gambar 4.17.

Tabel 4.8. Tabel pekerjaan

Nama Kolom	Tipe Data	Keterangan	Deskripsi
id_kerja	int	Primary key	Id tempat bekerja pengguna (1, 2, 3, …)
tempat_kerja	varchar	-	Tempat bekerja (RSSA – Bagian UGD, …)

id_kerja ▲	tempat_kerja
0	Belum Bekerja
2	RSSA - Bagian Anak
3	RSSA - Bagian Operasi
5	RS Hermina - Bagian Penyakit Dalam

Gambar 4.17. Implementasi tabel pekerjaan

Berikut ini merupakan implementasi basis data yang digunakan untuk menyimpan data obat-obatan yang dijelaskan pada Tabel 4.9. dan Gambar 4.18.

Tabel 4.9. Tabel obat

Nama Kolom	Tipe Data	Keterangan	Deskripsi
id_obat	int	Primary key	Id obat (1, 2, 3, …)
nama_obat	varchar	-	Nama obat (Acetram, …)
komposisi	varchar	-	Komposisi obat (Tramadol 37.5 mg dan paracetamol 325 mg, …)
indikasi	varchar	-	Indikasi obat (Nyeri akut & kronik sedang s/d berat; pemeriksaan diagnostik atau terapeutik yg menyakitkan; nyeri pasca op. Pengobatan olahraga & rehabilitasi, …)
dosis	varchar	-	Dosis penggunaan obat (Meredakan nyeri 1-2 tab tiap 4-6 jam. Maks: 8 tab/hr. Penderita dg bersihan kreatinin <30 mL/mnt 2 tab tiap 12 jam, …)
penyajian	varchar	-	Cara penyajian obat (Dapat diberikan bersama atau tanpa makanan, …)
kontra	varchar	-	Kontra indikasi obat (Intoksikasi alkohol akut; hipnotik, analgesik yg bekerja scr sentral, opiat, atau obat psikotropik, …)
perhatian	varchar	-	Perhatian penggunaan obat (Ggn ginjal. Hamil & laktasi, …)
reaksi	varchar	-	Rekasi obat (Mual, pusing, somnolen, …)
interaksi	varchar	-	Interaksi obat (MAOI non selektif, alkohol, karbamazepin, agonis-antagonis opiat, …)

id_obat	nama_obat	komposisi	indikasi	dosis	penyajian	kontra	perhatian	reaksi	interaksi
1	Abixim	Cefixime	ISK tak terkomplikasi yg disebabkan E coli & P mir...	Kaps Dws & anak dg BB>30 kg 50-100 mg 2 x/hr. Infe...	Dapat diberikan bersama atau tanpa makanan : Dpt d...	Hipersensitivitas.	Riwayat hipersensitivitas thd antibiotik sefem lai...	Ruam, urtikaria, eritema, pruritus atau demam; gra...	
2	Acetram	Tramadol 37.5 mg, paracetamol 325 mg.	Nyeri akut & kronik sedang s/d berat; pemeriksaan ...	Meredakan nyeri 1-2 tab tiap 4-6 jam. Maks: 8 tab/...	Dapat diberikan bersama atau tanpa makanan.	Intoksikasi alkohol akut; hipnotik, analgesik yg b...	Ggn ginjal. Hamil & laktasi.	Mual, pusing, somnolen.	MAOI non selektif, alkohol, karbamazepin, agonis-a...
3	Bactirom	Cefpirome sulfate.	Infeksi sal napas bwh, ISK atas & bwh terkomplikas...	ISK atas & bwh terkomplikasi, infeksi kulit & jar ...		Hipersensitif thd sefalosporin.	Monitor fungsi ginjal selama terapi bersama dg ami...	Reaksi hipersensitivitas, ggn GI, ggn fungsi ginja...	Aminoglikosida, loop diuretic.
4	Balticin	Gentamycin sulfate.	Infeksi kulit primer maupun sekunder krn bakteri y...	Oleskan pd bagian yg sakit 3-4 x/hr, bila perlu di...		Infeksi krn jamur atau virus.	Penggunaan terus menerus dpt menyebabkan resistens...	Iritasi lokal.	
5	Calcimega	Dibasic Ca phosphate 500 mg, DHA 100 mg, omega-3 f...	Suplemen Ca.	1 kaps/hr.	Sebaiknya diberikan bersama makanan.	Hiperkalsemia & hiperkalsiuria, batu ginjal, gagal...	Monitor kadar Ca serum & fungsi ginjal pd pengguna...	Hiperkalsemia & hiperkalsiuria.	Menghambat absorpsi tetrasiklin.

Gambar 4.18. Implementasi tabel obat

Berikut ini merupakan implementasi basis data yang digunakan untuk menyimpan data penyakit yang dijelaskan pada Tabel 4.10. dan Gambar 4.19.

Tabel 4.10. Tabel penyakit

Nama Kolom	Tipe Data	Keterangan	Deskripsi
id_penyakit	int	Primary key	Id penyakit (1, 2, 3, …)
nama_penyakit	longtext	-	Nama penyakit (Infark Miokard Akut (IMA), …)
definisi	longtext	-	Definisi penyakit (Infark miokard akut (*acute myocardial infarction*) merupakan kematian sel-sel miokard secara akut akibat kurangnya aliran darah yang teroksigenasi di dalam arteri koroner. Dikenal juga sebagai serangan jantung., …)
patofisiologi	longtext	-	Proses perjalanan penyakit (Cedera pada endotel arteri -> meningkatkan adhesi platelet -> respons inflamasi menyebabkan monosit dan limfosit T bermigrasi di dalam lapisan intima -> makrofag dan otot polos meluas disertai lipid, membentuk garis-garis lemak dan membentuk fibrous cap -> penipisan cap meningkatkan kerentanan untuk ruptur atau pendarahan -

Nama Kolom	Tipe Data	Keterangan	Deskripsi
			> ruptur memicu pembentuksn trombus dan vasokonstriksi -> hasil: trombus dengan arteri yang menyempit. Jika oklusi berlangsung lebih dari 20 menit akan menyebabkan IMA., …)
gambaran_klinis	longtext	-	Gejala klinis penyakit (IMA timbul sebagai nyeri dada atau rasa tidak nyaman yang berlangsung selama 20 menit atau lebih. Nyeri digambarkan sebagai tekanan, rasa seperti diikat, rasa berat, seperti terbakar, atau sensasi seperti diperas atau diremas, biasanya di dada bagian tengah atau epigastrium; keluhan ini dapat menjalar ke lengan, bahu leher, rahang, atau punggung. Rasa tidak nyaman bisa disertai kelemahan dispnea, diaforesis, atau ansietas, yang tidak hilang dengan NTG..., …)
tes_diagnostik	longtext	-	Cara melakukan tes diagnostik penyakit (- Gambaran EKG - Marker jantung (CK, mioglobin, dan troponin)), …)
penanganan	longtext	-	Cara penanganan penyakit (- Fokus pada penjalaran nyeri, sesak, dan diaforesis - Pemeriksaan EKG 12 sadapan dan lab untuk marker jantung - MONA: morfin, oksigen, NTG, dan aspirin 160-325 mg, per oral. Jika alergi aspirin, berikan ticlopidine (Ticlid) atau clopidogrel (Plavix) - Berikan oksigen tambahan untuk mempertahankan SpO2 > 90% - Berikan tablet NTG SL atau bentuk semprot - Berikan morfin IV 2-4 mg

Nama Kolom	Tipe Data	Keterangan	Deskripsi
			setiap 15 menit sampai nyeri terkontrol (pantau adanya hipotensi dan depresi pernapasan), …)

id_patofisiologi	nama_penyakit	definisi	patofisiologi	gambaran_klinis	tes_diagnostik	penanganan
1	Sindrom Koroner Akut (Acute Coronary Syndrome, ACS...	ACS merupakan istilah yang digunakan untuk menunju...	Angina takstabil menunjukkan progresi penyakit art...	ACS bermanifestasi sebagai nyeri dada, diaforesis,...	- EKG - Ekokardiogram - Marker siklus jantung (t...	- Berikan oksigen untuk mempertahankan SaO2 > 90% ...
2	Angina Takstabil	Angina takstabil merupakan nyeri dada beronset tob...	Aterosklerosis -> obstruksi arteri koroner -> penu...	Nyeri dada timbul sebagai nyeri substernal, terasa...	- EKG 12 sadapan - Laboratorium: marker jantung: ...	- Tirah baring - Lakukan pemeriksaan EKG dan labo...
3	Infark Miokard Akut (IMA)	Infark miokard akut (acute myocardial infarction) ...	Cedera pada endotel arteri -> meningkatkan adhesi ...	IMA timbul sebagai nyeri dada atau rasa tidak nyam...	- Gambaran EKG - Marker jantung (CK, mioglobin, d...	- Fokus pada penjalaran nyeri, sesak, dan diafores...

Gambar 4.19. Implementasi tabel penyakit

Berikut ini merupakan implementasi basis data yang digunakan untuk menyimpan data anatomi yang dijelaskan pada Tabel 4.11. dan Gambar 4.20.

Tabel 4.11. Tabel anatomi

Nama Kolom	Tipe Data	Keterangan	Deskripsi
id_anatomi	int	Primary key	Id anatomi (1, 2, 3, …)
nama_anatomi	varchar	-	Nama anatomi (Tengkorak, …)
definisi	varchar	-	Denifisi anatomi (Tulang kerangka kepala yang disusun menjadi dua bagian kranium(ada kalanya disebut kalvaria) terdiri atas delapan tulang dan kerangka wajah terdiri dari empat belas tulang. Rongga tengkorak mempunyai permukaan atas yang dikenal sebaagai kubah tengkorak, licin permukaa luar dan pada permukaan dalam ditandai dengan gili-gili dan lekukan supaya dapat sesuai dengan otak dan pembuluh darah. Permukaan bawah rongga dikenal sebagai dasar tengkorak atau basis kranii. Permukaan ini ditembusi banyak lubang supaya dapat

Nama Kolom	Tipe Data	Keterangan	Deskripsi
			dilalui serabut pembuluh darah, …)

id_anatomi	nama_anatomi	definisi
1	Kulit	Organ tubuh paling luar. Luas kulit org dewasa 1,5...
2	Tengkorak	Tulang kerangka kepala yang disusun menjadi dua ba...
3	Tulang Wajah	Terdapat 14 tulang wajah yang sempurna, kecuali ma...
4	Rongga Dada	Rongga dada atau toraks tersusun atas tulang dan t...
5	Gelang panggul atau tulang-tulang pelvis	Adalah penghubung antara badan dan anggota bawah. ...
6	Scapula	Atau tulang belikat membentuk bagian belakang gela...

Gambar 4.20. Implementasi tabel anatomi

4.1.6. Implementasi *Web Service*

Implementasi *web service* yang akan digunakan pada sistem dilakukan dengan mengimplementasikan rancangan *web service* yang telah dijelaskan pada bab sebelumnya.

4.1.6.1. *Web Service Login*

Berikut ini merupakan potongan program untuk melakukan proses *login* yang akan dijelaskan pada Gambar 4.21.

```
public function getUserByEmailAndPassword($email, $password) {
    $result = mysql_query("SELECT * FROM users WHERE email = '$email'")
    or die(mysql_error());
    // check for result
    $no_of_rows = mysql_num_rows($result);
    if ($no_of_rows > 0) {
        $result = mysql_fetch_array($result);
        $salt = $result['salt'];
        $encrypted_password = $result['encrypted_password'];
        $hash = $this->checkhashSSHA($salt, $password);
        // check for password equality
        if ($encrypted_password == $hash) {
            // user authentication details are correct
            return $result;
        }
    } else {
        // user not found
        return false;
    }
}

public function checkhashSSHA($salt, $password) {
```

```
        $hash = base64_encode(sha1($password . $salt, true) . $salt);
        return $hash;
}
```

Gambar 4.21. *Web service* login

Gambar 4.21. menjelaskan proses *login* yang membutuhkan data *email* dan *password* pengguna untuk melakukan proses pengecekan apakah pengguna dengan *email* dan *password* tersebut ada pada basis data. Jika proses pengecekan berhasil, maka *server* akan mengirimkan respon sukses dengan nilai 1 pada sistem yang akan digunakan sebagai tanda bahwa proses *login* berhasil. Sedangkan jika proses pengecekan gagal, maka *server* akan mengirimkan respon *error* dengan nilai 1 pada sistem yang akan digunakan sebagai tanda bahwa proses *login* gagal.

4.1.6.2. *Web Service Register*

Berikut ini merupakan potongan program untuk melakukan proses *register* yang akan dijelaskan pada Gambar 4.22. dan Gambar 4.23.

```
public function isUserExisted($email) {
      $result = mysql_query("SELECT email from users WHERE email =
      '$email'");
      $no_of_rows = mysql_num_rows($result);
      if ($no_of_rows > 0) {
            // user existed
            return true;
      } else {
            // user not existed
            return false;
      }
}
```

Gambar 4.22. *Web service* cek keberadaan pengguna

```
public function storeUser($name, $email, $password) {
      $uuid = uniqid('', true);
      $hash = $this->hashSSHA($password);
      $encrypted_password = $hash["encrypted"]; // encrypted password
      $salt = $hash["salt"]; // salt
      $result = mysql_query("INSERT INTO users(unique_id, name, email,
      encrypted_password, salt, created_at, updated_at) VALUES('$uuid',
      '$name', '$email', '$encrypted_password', '$salt', NOW(),
      NOW())");
      // check for successful store
      if ($result) {
            // get user details
```

```
            $uid = mysql_insert_id(); // last inserted id
            $result = mysql_query("SELECT * FROM users WHERE uid =
            $uid");
            // return user details
            return mysql_fetch_array($result);
        } else {
            return false;
        }
}

public function hashSSHA($password) {
        $salt = sha1(rand());
        $salt = substr($salt, 0, 10);
        $encrypted = base64_encode(sha1($password.$salt, true) .$salt);
        $hash = array("salt" => $salt, "encrypted" => $encrypted);
        return $hash;
}
```

Gambar 4.23. *Web service* register

Gambar 4.23. menjelaskan proses registrasi akun yang membutuhkan data nama, *email*, dan *password* dimana email tersebut akan dicek terlebih dahulu, apakah pengguna dengan *email* tersebut sudah ada pada basis data sesuai dengan Gambar 4.22. Jika *email* tersebut belum ada pada basis data, maka *server* akan memasukkan data tersebut ke basis data dan akan memberikan respon sukses dengan nilai 1 pada sistem yang digunakan sebagai tanda bahwa proses registrasi berhasil dilakukan. Sedangkan jika *email* tersebut ada pada basis data, maka *server* akan memberikan respon *error* dengan nilai 1 yang akan dijadikan sebagai tanda bahwa proses registrasi gagal dilakukan.

4.1.6.3. *Web Service* Lihat Catatan

Berikut ini merupakan potongan program untuk melihat daftar catatatan pengguna yang akan dijelaskan pada Gambar 4.24.

```
.
.
$id_user = $_GET['id_user'];

$result = mysql_query("select n.id_user, n.id_kerja, n.id_note,
n.judul_note, u.name, n.tanggal from note n, users u where ( u.uid =
$id_user or (n.id_kerja = (select id_kerja from users where uid =
$id_user) and is_share = 1)) and n.id_user = u.uid  ORDER BY  n.tanggal
DESC") or die(mysql_error());
```

```
if (mysql_num_rows($result) > 0) {
    $response["note"] = array();
    while ($row = mysql_fetch_array($result)) {
        $note = array();
        $note["id_user"] = $row["id_user"];
        $note["id_kerja"] = $row["id_kerja"];
        $note["id_note"] = $row["id_note"];
        $note["judul_note"] = $row["judul_note"];
        $note["name"] = $row["name"];
        $note["tanggal"] = $row["tanggal"];
        array_push($response["note"], $note);
    }
    $response["success"] = 1;
    echo json_encode($response);
}
```

Gambar 4.24. *Web service* lihat catatan

Gambar 4.24. menjelaskan proses menampilkan catatan pada basis data yang membutuhkan id pengguna dan id tempat bekerja sebagai parameter untuk melakukan pencarian catatan. Jika catatan dengan id pengguna dan id tempat bekerja ditemukan, maka *server* akan memberikan respon sukses dengan nilai 1 dan mengirimkan data id pengguna, id tempat bekerja, id catatan, judul catatan, nama pemilik, dan tanggal pembuatan catatan pada sistem sebagai tanda bahwa proses menampilkan catatan berhasil dilakukan. Sedangkan jika id pengguna dan id tempat bekerja tidak ditemukan, maka *server* akan memberikan respon sukses dengan nilai 0 dan mengirimkannya pada sistem yang akan digunakan sebagai tanda bahwa proses menampilkan catatan gagal dilakukan.

4.1.6.4. *Web Service* Tambah Catatan

Berikut ini merupakan potongan program untuk melakukan proses penambahan catatan yang akan dijelaskan pada Gambar 4.25.

```
public function storeNote($id_user, $judul_note, $isi_note, $is_share) {
     $result = mysql_query("INSERT INTO note(id_user, id_kerja,
     judul_note, tanggal, isi_note, is_share) VALUES('$id_user',
     (select id_kerja from users where uid = $id_user), '$judul_note',
     NOW(), '$isi_note', '$is_share')");
     if ($result) {
          $uid = mysql_insert_id();
          $result = mysql_query("SELECT * FROM note WHERE id_note =
          $uid");
          return mysql_fetch_array($result);
```

```
    } else {
        return false;
    }
}
```

Gambar 4.25. *Web service* tambah catatan

Gambar 4.25. menjelaskan proses penambahan catatan yang membutuhkan id pengguna, judul, isi, dan status pembagian catatan untuk melakukan penambahan catatan pada basis data. Jika catatan berhasil ditambahkan pada basis data, maka *server* akan mengirimkan respon sukses dengan nilai 1 pada sistem. Sedangkan jika catatan gagal ditambahkan pada basis data, maka *server* akan mengirimkan respon *error* dengan nilai 1 pada sistem.

4.1.6.5. *Web Service* Lihat Detail Catatan

Berikut ini merupakan potongan program untuk melihat detail catatan yang akan dijelaskan pada Gambar 4.26.

```
.
.
$id_note = $_GET['id_note'];

$result = mysql_query("SELECT n.id_user, n.id_note, n.judul_note, u.name,
n.tanggal, n.isi_note, n.is_share FROM note n, users u WHERE u.uid =
n.id_user AND n.id_note = $id_note") or die(mysql_error());

if (mysql_num_rows($result) > 0) {
    $response["note"] = array();
    while ($row = mysql_fetch_array($result)) {
        $note = array();
        $note["id_user"] = $row["id_user"];
        $note["id_note"] = $row["id_note"];
        $note["judul_note"] = $row["judul_note"];
        $note["name"] = $row["name"];
        $note["tanggal"] = $row["tanggal"];
        $note["isi_note"] = $row["isi_note"];
        $note["is_share"] = $row["is_share"];
        array_push($response["note"], $note);
    }
    $response["success"] = 1;
    echo json_encode($response);
}
.
.
```

Gambar 4.26. *Web service* lihat detail catatan

Gambar 4.26. menjelaskan proses melihat detail catatan yang membutuhkan id catatan sebagai parameter pencarian catatan pada basis data. Jika catatan dengan id catatan tersebut berhasil ditemukan, maka *server* akan mengirimkan respon sukses dengan nilai 1 dan mengirimkan id pengguna, id catatan, judul catatan, nama pemilik catatan, tanggal pembuatan catatan, isi, dan status pembagian catatan pada sistem. Sedangkan jika id catatan tidak ditemukan, maka *server* akan mengirimkan respon sukses dengan nilai 0 pada sistem.

4.1.6.6. *Web Service* Update Catatan

Berikut ini merupakan potongan program untuk melakukan update catatan yang akan dijelaskan pada Gambar 4.27.

```
.
.
$id_note = $_POST['id_note'];
$judul_note = $_POST['judul_note'];
$isi_note = $_POST['isi_note'];
$is_share = $_POST['is_share'];

$result = mysql_query("UPDATE note SET judul_note = '$judul_note',
isi_note = '$isi_note', is_share = '$is_share' WHERE id_note =
$id_note");

if ($result) {
     $response["success"] = 1;
     $response["message"] = "Note successfully updated.";
     echo json_encode($response);
}
.
.
```

Gambar 4.27. *Web service update* catatan

Gambar 4.27. menjelaskan proses *update* catatan yang membutuhkan id, judul, isi, dan status pembagian catatan sebagai parameter untuk melakukan proses *update* catatan pada basis data. Jika catatan berhasil di*update*, maka *server* akan mengirimkan respon sukses dengan nilai 1 pada sistem.

4.1.6.7. *Web Service* Hapus Catatan

Berikut ini merupakan potongan program untuk melakukan hapus catatan yang akan dijelaskan pada Gambar 4.28.

```
.
.
$id_note = $_POST['id_note'];

$result = mysql_query("DELETE FROM note WHERE id_note = $id_note");

if (mysql_affected_rows() > 0) {
    $response["success"] = 1;
    $response["message"] = "Note successfully deleted.";
    echo json_encode($response);
}
.
.
```

Gambar 4.28. *Web service* hapus catatan

Gambar 4.28. menjelaskan proses hapus catatan yang membutuhkan id catatan sebagai parameter untuk melakukan proses penghapusan catatan dari basis data. Jika catatan berhasil dihapus, maka *server* akan mengirimkan respon sukses dengan nilai 1 pada sistem. Sedangkan jika catatan gagal dihapus, maka *server* akan mengirimkan respon sukses dengan nilai 0 pada sistem.

4.1.6.8. *Web Service* Lihat Tempat Bekerja

Berikut ini merupakan potongan program untuk menampilkan daftar tempat bekerja yang akan dijelaskan pada Gambar 4.29.

```
.
.
$result = mysql_query("SELECT * FROM pekerjaan") or die(mysql_error());
if (mysql_num_rows($result) > 0) {
    $response["pekerjaan"] = array();
    while ($row = mysql_fetch_array($result)) {
        // temp user array
        $pekerjaan = array();
        $pekerjaan["id_kerja"] = $row["id_kerja"];
        $pekerjaan["tempat_kerja"] = $row["tempat_kerja"];
        array_push($response["pekerjaan"], $pekerjaan);
    }
    $response["success"] = 1;
    echo json_encode($response);
}
.
```

```
.
```

Gambar 4.29. *Web service* lihat tempat bekerja

Gambar 4.29. menjelaskan proses menampilkan tempat bekerja dengan mengambil data tempat bekerja pada basis data. Jika data tempat bekerja ada, maka *server* akan mengirimkan respon sukses dengan nilai 1 dan mengirimkan id tempat bekerja dan tempat bekerja pada sistem. Sedangkan jika data tempat bekerja tidak ada, maka *server* akan mengirimkan respon sukses dengan nilai 0 pada sistem.

4.1.6.9. *Web Service* Tambah Tempat Bekerja

Berikut ini merupakan potongan program untuk melakukan proses penambahan tempat bekerja yang akan dijelaskan pada Gambar 4.30.

```
.
.
$tempat_kerja = $_POST["tempat_kerja"];
$query = "INSERT INTO pekerjaan(tempat_kerja) VALUES('$tempat_kerja')";
$result = mysql_query($query) or die(mysql_error());
if ($result) {
      $response["success"] = 1;
      $response["message"] = "Workplace created successfully!";
}
.
.
```

Gambar 4.30. *Web service* tambah tempat bekerja

Gambar 4.30. menjelaskan proses penambahan tempat bekerja yang membutuhkan nama tempat bekerja sebagai parameter untuk melakukan proses penambahan tempat bekerja pada basis data. Jika tempat bekerja berhasil ditambahkan, maka *server* akan mengirimkan respon sukses dengan nilai 1 pada sistem. Sedangkan jika tempat bekerja gagal ditambahkan, maka *server* akan mengirimkan respon sukses dengan nilai 0 pada sistem.

4.1.6.10. *Web Service* Pilih Tempat Bekerja

Berikut ini merupakan potongan program untuk melakukan pemilihan tempat bekerja yang akan dijelaskan pada Gambar 4.31.

```
.
.
$id_user = $_GET['uid'];
$tempat_kerja = $_GET['tempat_kerja'];

$result = mysql_query("UPDATE users SET id_kerja = (select id_kerja from
pekerjaan where tempat_kerja = '$tempat_kerja') WHERE uid = $id_user");

if ($result) {
     $response["success"] = 1;
     $response["message"] = "User successfully updated.";
     echo json_encode($response);
}
.
.
```

Gambar 4.31. *Web service* pilih tempat bekerja

Gambar 4.31. menjelaskan proses pemilihan tempat bekerja yang membutuhkan id pengguna dan id tempat bekerja sebagai parameter untuk memperbarui data pengguna. Jika proses memperbarui berhasil dilakukan, maka *server* akan mengirimkan respon sukses dengan nilai 1 pada sistem. Sedangkan jika proses memperbarui gagal dilakukan, maka *server* akan mengirimkan respon sukses dengan nilai 0 pada sistem.

4.1.6.11. *Web Service* Lihat Profil

Berikut ini merupakan potongan program untuk melihat profil pengguna yang akan dijelaskan pada Gambar 4.32.

```
.
.
$id_user = $_GET['id_user'];
$id_kerja = $_GET['id_kerja'];

$result = mysql_query("SELECT u.uid, u.name, p.tempat_kerja, count(u.uid)
as jum_catatan, (select count(id_user) from note where id_user = $id_user
and is_share = 1) as jum_share FROM note n, users u, pekerjaan p where
n.id_user = u.uid and u.uid = $id_user and u.id_kerja = p.id_kerja") or
die(mysql_error());

if (mysql_num_rows($result) > 0) {
    $response["profile"] = array();
    while ($row = mysql_fetch_array($result)) {
        $profile = array();
        $profile["uid"] = $row["uid"];
        $profile["name"] = $row["name"];
        $profile["tempat_kerja"] = $row["tempat_kerja"];
        $profile["jum_catatan"] = $row["jum_catatan"];
        $profile["jum_share"] = $row["jum_share"];
```

```
        array_push($response["profile"], $profile);
    }
    $response["success"] = 1;
    echo json_encode($response);
}
.
.
```

Gambar 4.32. *Web service* lihat profil

Gambar 4.32. menjelaskan proses menampilkan profil yang membutuhkan id pengguna dan id tempat bekerja sebagai parameter untuk melakukan pencarian data pengguna pada basis data. Jika data ditemukan, maka *server* akan mengirimkan respon sukses dengan nilai 1 dan mengirimkan id, nama, tempat bekerja pengguna, jumlah catatan yang dimiliki, dan dibagiakan ke pengguna lain yang memiliki tempat bekerja sama pada sistem. Sedangkan jika data tidak ditemukan, maka *server* akan mengirimkan respon sukses dengan nilai 0 pada sistem.

4.1.6.12. *Web Service* Lihat Konten Keperawatan

Implementasi *web service* lihat konten keperawatan dibagi menjadi 4, yaitu obat, penyakit, pemeriksaan diagnostik berdasarkan nama penyakit, dan anatomi.

1. Obat

Berikut ini merupakan potongan program untuk menampilkan data obat yang akan dijelaskan pada Gambar 4.33.

```
.
.
$result = mysql_query("SELECT * FROM obat") or die(mysql_error());
if (mysql_num_rows($result) > 0) {
    $response["obat"] = array();
    while ($row = mysql_fetch_array($result)) {
        $obat = array();
        $obat["id_obat"] = $row["id_ obat"];
        $obat["nama_obat"] = $row["nama_obat"];
        $obat["komposisi"] = $row["komposisi"];
        $obat["indikasi"] = $row["indikasi"];
        $obat["dosis"] = $row["dosis"];
        $obat["penyajian"] = $row["penyajian"];
        $obat["kontra"] = $row["kontra"];
        $obat["perhatian"] = $row["perhatian"];
        $obat["reaksi"] = $row["reaksi"];
        $obat["interaksi"] = $row["interaksi"];
        array_push($response["obat"], $obat);
```

```
    }
    $response["success"] = 1;
    echo json_encode($response);
}
.
.
```

Gambar 4.33. *Web service* lihat data obat

Gambar 4.33. menjelaskan proses menampilkan data obat dari obat pada basis data. Jika data obat ada, maka *server* akan mengirimkan respon sukses dengan nilai 1 mengirimkan id, nama, komposisi, indikasi, dosis, cara pemberian, kontra indikasi, perhatian, reaksi, dan interaksi obat pada sistem. Sedangkan jika data obat tidak ada, maka *server* akan mengirimkan respon sukses dengan nilai 0 pada sistem.

2. Penyakit

Berikut ini merupakan potongan program untuk menampilkan data penyakit yang akan dijelaskan pada Gambar 4.34.

```
.
.
$result = mysql_query("SELECT id_penyakit, nama_penyakit, definisi,
mikrobiologi, manifes_klinis FROM penyakit") or die(mysql_error());
if (mysql_num_rows($result) > 0) {
    $response["penyakit"] = array();
    while ($row = mysql_fetch_array($result)) {
        $penyakit = array();
        $penyakit["id_penyakit"] = $row["id_ penyakit"];
        $penyakit["nama_penyakit"] = $row["nama_penyakit"];
        $penyakit["definisi"] = $row["definisi"];
        $penyakit["patofisiologi"] = $row["patofisiologi"];
        $penyakit["gambaran_klinis"] = $row["gambaran_klinis"];
        $penyakit["tes_diagnostik"] = $row["tes_diagnostik"];
        $penyakit["penanganan"] = $row["penanganan"];
        array_push($response["penyakit"], $penyakit);
    }
    $response["success"] = 1;
    echo json_encode($response);
}
.
.
```

Gambar 4.34. *Web service* lihat data penyakit

Gambar 4.34. menjelaskan proses menampilkan data penyakit dari basis data. Jika data penyakit ada, maka *server* akan mengirimkan respon sukses dengan

nilai 1 mengirimkan id, nama, definisi, patofisiologi, gambaran klinis, tes diagnostic, dan penanganan penyakit pada sistem. Sedangkan jika data penyakit tidak ada, maka *server* akan mengirimkan respon sukses dengan nilai 0 pada sistem.

3. Anatomi

Berikut ini merupakan potongan program untuk menampilkan data anatomi yang akan dijelaskan pada Gambar 4.35.

```
.
.
$id_anatomi = $_GET['id_anatomi'];

$result = mysql_query("SELECT * FROM anatomi") or die(mysql_error());
if (mysql_num_rows($result) > 0) {
    $response["anatomi"] = array();
    while ($row = mysql_fetch_array($result)) {
        $anatomi = array();
        $anatomi["id_anatomi"] = $row["id_ anatomi"];
        $anatomi["nama_anatomi"] = $row["nama_anatomi"];
        $anatomi["definisi"] = $row["definisi"];
        array_push($response["anatomi"], $anatomi);
    }
    $response["success"] = 1;
    echo json_encode($response);
}
.
```

Gambar 4.35. *Web service* lihat data anatomi

Gambar 4.35. menjelaskan proses menampilkan data anatomi yang dari basis data. Jika data anatomi ada, maka *server* akan mengirimkan respon sukses dengan nilai 1 mengirimkan id, nama, definisi, dan fungsi anatomi pada sistem. Sedangkan jika data anatomi tidak ada, maka *server* akan mengirimkan respon sukses dengan nilai 0 pada sistem.

4.1.7. Implementasi Kode Program

Implementasi kode program dilakukan berdasarkan pada perancangan sistem yang telah dijelaskan pada bab sebelumnya.

4.1.7.1. Kode Program *Login*

Berikut ini merupakan potongan program untuk melakukan proses *login* yang akan dijelaskan pada Gambar 4.36.

```
.
.
@Override
protected JSONObject doInBackground(String... args) {
     UserFunctions user_function = new UserFunctions();
     JSONObject json = user_function.loginUser(user, pass);
     return json;
}

@Override
protected void onPostExecute(JSONObject json) {
        try {
            if (json.getString(KEY_SUCCESS) != null) {
                String success = json.getString(KEY_SUCCESS);
                if(Integer.parseInt(success) == 1){
                    uid = json.getString(KEY_UID);
                    session.createLoginSession(uid, user);
                    if(posisi.equalsIgnoreCase("catatan")){
                        fragment = new CatatanFragment();
                    } else if(posisi.equalsIgnoreCase("profil")){
                        if(Integer.parseInt(id_kerja) == 0){
                            fragment = new Pekerjaan();
                        } else{
                            fragment = new Profil();
                        }
                    } else{
                        fragment = new CategoryGrid();
                    }

                    if (fragment != null) {
                        FragmentManager fragmentManager =
                        getFragmentManager();
                        fragmentManager.beginTransaction()
                        .replace(R.id.frame_container,
                        fragment).commit();
                    } else {
                        Log.e("MainActivity", "Error in creating
                        fragment");
                    }
                }else{
                    Toast.makeText(getActivity(), "Invalid
                    email/password", Toast.LENGTH_SHORT).show();
                }
            }
        } catch (JSONException e) {
            e.printStackTrace();
        }
}
.
.
```

Gambar 4.36. Kode program *login*

```
public UserFunctions(){
```

```
        parser = new JSONParser();
}

public JSONObject loginUser(String email, String password){
        List<NameValuePair> params = new ArrayList<NameValuePair>();
        params.add(new BasicNameValuePair("tag", tag_login));
        params.add(new BasicNameValuePair("email", email));
        params.add(new BasicNameValuePair("password", password));

        JSONObject json = parser.makeHttpRequest(url_login, "POST",
        params);

        return json;
}
```

Gambar 4.37. Kode program *user function*

Gambar 4.36. merupakan kode yang digunakan untuk melakukan proses *login*. Data yang diperlukan untuk proses login yaitu *email* dan *password* yang diperoleh dari kolom *username* dan *password* yang telah disediakan oleh aplikasi. Sistem akan meminta *server* untuk melakukan pencarian data *email* dan *password* tersebut pada basis data seperti yang dijelaskan pada Gambar 4.37. Jika *server* mengirimkan respon sukses dengan nilai 1, maka proses login berhasil dilakukan. Sedangkan jika *server* mengirimkan respon *error* dengan nilai 1, maka proses login gagal dilakukan dan sistem akan menampilkan pesan kesalahan pada pengguna.

4.1.7.2. Kode Program *Register*

Berikut ini merupakan potongan program untuk melakukan proses *register* yang akan dijelaskan pada Gambar 4.38.

```
class RegisterProcess extends AsyncTask<String, String, JSONObject>{
        .
        .

        @Override
        protected JSONObject doInBackground(String... args) {
        // TODO Auto-generated method stub
            UserFunctions user_function = new UserFunctions();

           JSONObject json = user_function.registerUser(full, user, pass);

            return json;
        }

        @Override
```

```
        protected void onPostExecute(JSONObject json) {
                try {
                  if (json.getString(KEY_SUCCESS) != null) {
                       String success = json.getString(KEY_SUCCESS);
                       if(Integer.parseInt(success) == 1){
                       // user successfully registred
                       // Store user details in SQLite Database
                       DatabaseHandler db = new
                       DatabaseHandler(getActivity());
                       JSONObject json_user = json.getJSONObject("user");
                       UserFunctions logout = new UserFunctions();
                       logout.logoutUser(getActivity());
                       db.addUser(json_user.getString(KEY_NAME),
                       json_user.getString(KEY_EMAIL),
                       json.getString(KEY_UID),
                       json_user.getString(KEY_CREATED_AT));
                       fragment = new LoginUser();

                       Bundle bundle = new Bundle();

                       bundle.putString("posisi", "register");

                       fragment.setArguments(bundle);

                       if (fragment != null) {
                           FragmentManager fragmentManager =
                           getFragmentManager();
                           fragmentManager.beginTransaction().replace(R.id.f
                           rame_container, fragment).commit();
                       } else {
                           Log.e("MainActivity", "Error in creating
                           fragment");
                       }
                       .
                       .
    }
```

Gambar 4.38. Kode program *register*

```
public UserFunctions(){
     parser = new JSONParser();
}

public JSONObject registerUser(String name, String email, String
password){
     List<NameValuePair> params = new ArrayList<NameValuePair>();
     params.add(new BasicNameValuePair("tag", tag_register));
     params.add(new BasicNameValuePair("name", name));
     params.add(new BasicNameValuePair("email", email));
     params.add(new BasicNameValuePair("password", password));
     JSONObject json = parser.makeHttpRequest(url_register, "POST",
     params);

     return json;
}
```

Gambar 4.39. Kode program *user function*

Gambar 4.38. merupakan kode yang digunakan untuk melakukan proses regestrasi akun. Data yang diperlukan untuk proses login yaitu nama, *email*, dan *password* yang diperoleh dari kolom nama, *email*, dan *password* yang telah disediakan oleh aplikasi. Sistem akan meminta *server* untuk melakukan pencarian data nama, *email*, dan *password* tersebut pada basis data seperti yang dijelaskan pada Gambar 4.39. Jika data tidak ada, maka *server* akan mengirimkan respon sukses dengan nilai 1 dan *server* akan memasukkan data registrasi penggun pada basis data. Sedangkan jika data ada, maka *server* akan mengirimkan respon *error* dengan nilai 1 dan sistem akan menampilkan pesan kesalahan pada pengguna.

4.1.7.3. Kode Program Lihat Catatan

Berikut ini merupakan potongan program untuk melihat daftar catatan yang akan dijelaskan pada Gambar 4.40.

```
.
.
lv.setOnItemClickListener(new OnItemClickListener() {

   @Override
   public void onItemClick(AdapterView<?> arg0, View arg1, int arg2, long
   arg3) {
       fragment = new DetailCatatan();

       String get_id_note = ((TextView)
       arg1.findViewById(R.id.id_note)).getText().toString();

       Bundle bundle = new Bundle();
       bundle.putString("id_note", get_id_note);
       fragment.setArguments(bundle);

       FragmentManager fragmentManager = getFragmentManager();
       fragmentManager.beginTransaction()
       .replace(R.id.frame_container, fragment).commit();
       }
});
.
.
protected ArrayList<Note> doInBackground(String... args) {
   // TODO Auto-generated method stub
   List<NameValuePair> params = new ArrayList<NameValuePair>();
   params.add(new BasicNameValuePair(tag_id_user, id_user));
   JSONObject json = parser.makeHttpRequest(url, "GET", params);
   try {
```

```
        int success = json.getInt(tag_success);
        if(success == 1){
          note = json.getJSONArray(tag_products);
          for(int i = 0; i < note.length(); i++){
            JSONObject data = note.getJSONObject(i);
            String id_note = data.getString(tag_id_note);
            String judul_note = data.getString(tag_judul_note);
            String tanggal = data.getString(tag_tanggal);
            String nama = data.getString(tag_name);
            Note catatan = new Note();
            catatan.setIdNote(id_note);
            catatan.setJudulNote(judul_note);
            catatan.setTanggal(tanggal);
            catatan.setNama(nama);
            note_list.add(catatan);
    .
    .
}
```

Gambar 4.40. Kode program lihat catatan

Gambar 4.40 merupakan kode yang digunakan untuk melakukan proses pencarian catatan yang dibuat oleh pengguna pengguna lain tetapi memiliki tempat bekerja sama. Data yang diperlukan untuk proses pencarian data catatan yaitu id pengguna dan id tempat bekerja. Data id pengguna dan id kerja pengguna tersebut akan dikirimkan ke *server* sebagai parameter untuk mencari data catatan. Jika *server* mengirimkan respon sukses dengan nilai 1, maka proses pencarian berhasil dilakukan dan sistem akan menampilakan daftar catatan. Sedangkan jika *server* mengirimkan respon sukses dengan nilai 0, maka proses pencarian gagal dilakukan dan sistem akan menampilakan pesan kesalahan.

4.1.7.4. Kode Program Tambah Catatan

Berikut ini merupakan potongan program untuk melakukan proses penambahan catatan yang akan dijelaskan pada Gambar 4.41.

```
class AddNoteProcess extends AsyncTask<String, String, JSONObject>{

    String judul, isi;

    @Override
    protected void onPreExecute() {
       super.onPreExecute();

       judul_note = (EditText)
       getView().findViewById(R.id.judul_add_note);
```

```
    isi_note = (EditText) getView().findViewById(R.id.isi_add_note);
    share =(CheckBox) getView().findViewById(R.id.is_share);

    judul = judul_note.getText().toString();
    isi = isi_note.getText().toString();

    if(share.isChecked()){
          is_share = "1";
    } else{
          is_share = "0";
    }

}

@Override
protected JSONObject doInBackground(String... args) {
    UserFunctions user_function = new UserFunctions();

    JSONObject json = user_function.addNote(id_user, judul, isi,
    is_share);

    try {
          Log.d("key", json.getString(KEY_SUCCESS));
    } catch (JSONException e) {
          e.printStackTrace();
    }

    return json;
}

@Override
protected void onPostExecute(JSONObject json) {
    try {
          Log.d("key", json.getString(KEY_SUCCESS));

          if (json.getString(KEY_SUCCESS) != null) {
              String success = json.getString(KEY_SUCCESS);
          if(Integer.parseInt(success) == 1){
              fragment = new CatatanFragment();

          if (fragment != null) {
                FragmentManager fragmentManager =
                getFragmentManager();
                fragmentManager.beginTransaction()
                .replace(R.id.frame_container, fragment).commit();
          } else {
                Log.e("MainActivity", "Error in creating fragment");
          }

        } else{
                Toast.makeText(getActivity(), "Error occured in
                saving note", Toast.LENGTH_SHORT).show();
          }
.
.
     public void AddNoteCaller(View view){
```

```
            new AddNoteProcess().execute();
        }
```

Gambar 4.41. Kode program tambah catatan

```
private static String url_add_note = "http://perawat.meximas.com/";
private static String tag_add_note = "add_note";

public UserFunctions(){
    parser = new JSONParser();
}

public JSONObject addNote(String id_user, String judul_note, String
isi_note, String is_share){
    List<NameValuePair> params = new ArrayList<NameValuePair>();
    params.add(new BasicNameValuePair("tag", tag_add_note));
    params.add(new BasicNameValuePair("id_user", id_user));
    params.add(new BasicNameValuePair("judul_note", judul_note));
    params.add(new BasicNameValuePair("isi_note", isi_note));
    params.add(new BasicNameValuePair("is_share", is_share));
    JSONObject json = parser.makeHttpRequest(url_add_note, "POST",
    params);

    return json;
}
```

Gambar 4.42. Kode program *user function*

Gambar 4.41. merupakan kode yang digunakan untuk melakukan proses penambahan catatan. Data yang diperlukan untuk proses penambahan catatan yaitu id pengguna, judul, isi dan status pembagian catatan yang diperoleh dari kolom yang telah disediakan oleh sistem. Sistem akan meminta *server* untuk memasukkan data id pengguna, judul, isi dan status pembagian catatan tersebut pada basis data seperti yang dijelaskan pada Gambar 4.42. Jika *server* mengirimkan respon sukses dengan nilai 1 menandakan bahwa proses penambahan catatan berhasil dilakukan. Sedangkan jika *server* mengirimkan respon sukses dengan nilai 0 menandakan bahwa proses penambahan catatan gagal dilakukan dan sistem akan menampilkan pesan kesalahan pada pengguna.

4.1.7.5. Kode Program Lihat Detail Catatan

Berikut ini merupakan potongan program untuk melihat detail catatan yang akan dijelaskan pada Gambar 4.43.

```
    .
```

```
.
@Override
protected JSONObject doInBackground(String... args) {
   List<NameValuePair> params = new ArrayList<NameValuePair>();
   params.add(new BasicNameValuePair(tag_id_note, id_note));
   JSONObject json = parser.makeHttpRequest(url, "GET", params);

   try {
    if (json.getString(tag_success) != null) {
     int success = json.getInt(tag_success);
     if(success == 1){
         note = json.getJSONArray(tag_products);
         data = note.getJSONObject(0);
     } else{
         Toast.makeText(getActivity(), "Error occured in loading note
         detail", Toast.LENGTH_SHORT).show();
}
```

Gambar 4.43. Kode program lihat detail catatan

Gambar 4.43. merupakan kode yang digunakan untuk melakukan proses melihat detail catatan. Data yang diperlukan untuk proses melihat detail catatan yaitu id catatan yang diperoleh dari daftar catatan yang dipilih oleh pengguna sebelumnya. Sistem akan mengirimkan data tersebut ke *server* sebagai parameter dalam mencari detail catatan yang berada pada basis data. Jika *server* mengirimkan respon sukses dengan nilai 1 menandakan bahwa proses pencarian berhasil dilakukan dan sistem akan menampilkan data detail catatan kepada pengguna. Sedangkan jika *server* mengirimkan respon sukses dengan nilai 0, maka sistem akan menampilkan pesan kesalahan pada pengguna.

4.1.7.6. Kode Program *Update* Catatan

Berikut ini merupakan potongan program untuk melakukan proses *update* catatan yang akan dijelaskan pada Gambar 4.44.

```
.
.
protected String doInBackground(String... args) {
    judul = judul_note.getText().toString();
    isi = isi_note.getText().toString();
    if(is_share.isChecked()){
       share = "1";
    } else{
       share = "0";
    }
```

```
Log.d("Get Text", judul + isi + share);

List<NameValuePair> params = new ArrayList<NameValuePair>();
params.add(new BasicNameValuePair(tag_id_note, id_note));
params.add(new BasicNameValuePair(tag_judul_note, judul));
params.add(new BasicNameValuePair(tag_isi_note, isi));
params.add(new BasicNameValuePair(tag_is_share, share));

JSONObject json = parser.makeHttpRequest(url_update, "POST", params);

try {

  if(Integer.parseInt(json.getString(tag_success)) == 1){

    fragment = new DetailCatatan();

    Bundle bundle = new Bundle();
    bundle.putString("id_note", id_note);
    fragment.setArguments(bundle);
    if (fragment != null) {
      FragmentManager fragmentManager = getFragmentManager();
      fragmentManager.beginTransaction()
      .replace(R.id.frame_container, fragment).commit();
    } else {
        Log.e("MainActivity", "Error in creating fragment");
    }
.
.
```

Gambar 4.44. Kode program *update* catatan

Gambar 4.44. merupakan kode yang digunakan untuk melakukan proses *update* catatan. Data yang diperlukan untuk proses *update* catatan yaitu id, judul, isi, dan status pembagian catatan yang diperoleh dari kolom yang disediakam oleh aplikasi. Sistem akan mengirimkan data tersebut ke *server* sebagai parameter untuk memperbarui data catatan pada basis data. Tetapi, sebelum mengirimkan data tersebut ke *server*, sistem akan melakukan pengecekan apakah id pengguna sama dengan id pemilik catatan. Jika sama, maka proses *update* dapat dilakukan. Jika tidak, maka proses *update* tidak dapat dilakukan. Jika *server* mengirimkan respon sukses dengan nilai 1 menandakan bahwa proses *update* berhasil dilakukan.

4.1.7.7. Kode Program Hapus Catatan

Berikut ini merupakan potongan program untuk melakukan proses penghapusan catatan yang akan dijelaskan pada Gambar 4.45.

```
.
.
protected String doInBackground(String... args) {
    List<NameValuePair> params = new ArrayList<NameValuePair>();
    params.add(new BasicNameValuePair(tag_id_note, id_note));
    JSONObject json = parser.makeHttpRequest(url_delete, "POST", params);

    Log.d("Delete Product", json.toString());
    try {

      if(Integer.parseInt(json.getString(tag_success)) != 0){
       fragment = new CatatanFragment();

       if (fragment != null) {
         FragmentManager fragmentManager = getFragmentManager();
         fragmentManager.beginTransaction().replace(R.id.frame_container,
         fragment).commit();
       } else {
          Log.e("MainActivity", "Error in creating fragment");
       }
    }
    } catch (JSONException e) {
       e.printStackTrace();
    }
    .
    .
```

Gambar 4.45. Kode program hapus catatan

Gambar 4.45. merupakan kode yang digunakan untuk melakukan proses hapus catatan. Data yang diperlukan untuk proses hapus catatan yaitu id catatan yang diperoleh dari daftar catatan yang dipilih oleh pengguna sebelumnya. Sistem akan mengirimkan data tersebut ke *server* sebagai parameter untuk menghapus data catatan pada basis data. Jika *server* mengirimkan respon sukses dengan nilai 1 menandakan bahwa proses hapus catatan berhasil dilakukan dan sistem akan menampilkan data detail catatan kepada pengguna. Sedangkan jika *server* mengirimkan respon sukses dengan nilai 1, maka sistem akan menampilkan pesan kesalahan pada pengguna.

4.1.7.8. Kode Program Lihat Tempat Bekerja

Berikut ini merupakan potongan program untuk menampilkan daftar tempat bekerja yang akan dijelaskan pada Gambar 4.46.

```
.
.
private void populateSpinner() {
    List<String> lables = new ArrayList<String>();

    for (int i = 0; i < work_list.size(); i++) {
      lables.add(work_list.get(i).getTempat());
    }

    lables.add("Lainnya");

    ArrayAdapter<String> adapter = new
    ArrayAdapter<String>(getActivity(),android.R.layout.simple_spinner_it
    em, lables);
    adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropd
    own_item);
    spinner_work.setAdapter(adapter);
}
.
.
protected ArrayList<Work> doInBackground(String... args) {
    List<NameValuePair> params = new ArrayList<NameValuePair>();
    JSONObject json = parser.makeHttpRequest(url_get_work, "GET",
    params);
    try {
      int success = json.getInt(tag_success);
      if(success == 1){
        workplace = json.getJSONArray(tag_pekerjaan);

        for(int i = 0; i < workplace.length(); i++){
          JSONObject data = workplace.getJSONObject(i);
          String id_kerja = data.getString(tag_id_kerja);
          String tempat_kerja = data.getString(tag_tempat_kerja);
          Work kerja = new Work();
          kerja.setId(id_kerja);
          kerja.setTempat(tempat_kerja);
          work_list.add(kerja);

        }
       }
    } catch (JSONException e) {
.
.
```

Gambar 4.46. Kode program menampilkan daftar tempat bekerja

Gambar 4.46. merupakan kode yang digunakan untuk menampilkan daftar tempat bekerja. Sistem akan meminta *server* untuk mendapatkan data tempat bekerja yang berada pada basis data. Jika *server* mengirimkan respon sukses dengan nilai 1 menandakan bahwa data ditemukan dan sistem akan menampilkan daftar tempat kerja kepada pengguna dalam bentuk *spinner*. Sedangkan jika *server* mengirimkan respon sukses dengan nilai 0, maka sistem akan menampilkan pesan kesalahan pada pengguna.

4.1.7.9. Kode Program Tambah Tempat Bekerja

Berikut ini merupakan potongan program untuk melakukan proses penambahan tempat bekerja yang akan dijelaskan pada Gambar 4.47.

```
.
.
protected String doInBackground(String... args) {
    String tempat = add_work.getText().toString();
    List<NameValuePair> params = new ArrayList<NameValuePair>();
    params.add(new BasicNameValuePair(tag_tempat_kerja, tempat));
    JSONObject json = parser.makeHttpRequest(url_create_work, "POST",
    params);

    try {
      int success = json.getInt(tag_success);
      if(success == 1){
        isWorkCreated = true;
      }
    } catch (JSONException e) {
    .
.
```

Gambar 4.47. Kode program tambah tempat bekerja

Gambar 4.47. merupakan kode yang digunakan untuk melakukan proses penambahan tempat kerja. Setelah pengguna melihat daftar tempat kerja yang sebelumnya telah ditampilkan oleh sistem dan pengguna tidak menemukan tempat bekerja pada *spinner*, maka pengguna bisa membuat tempat bekerja baru dengan nama yang sesuai dengan tempat bekerja pengguna sekarang. Data yang diperlukan untuk proses penambahan tempat bekerja, yaitu nama tempat kerja yang diperoleh dari masukkan pengguna. Sistem akan mengirimkan data tersebut ke *server* yang nantinya akan dimasukkan ke basis data. Jika *server* mengirimkan

respon sukses dengan nilai 1 menandakan bahwa proses penambahan berhasil dilakukan dan sistem akan menampilkan daftar tempat bekerja pada pengguna. Jika *server* mengirimkan respon sukses dengan nilai 1, maka sistem akan menampilkan pesan kesalahan pada pengguna.

4.1.7.10. Kode Program Pilih Tempat Bekerja

Berikut ini merupakan potongan program untuk memilih tempat bekerja yang akan dijelaskan pada Gambar 4.48.

```
protected ArrayList<Work> doInBackground(String... args) {
   Log.d("nama pekerjaan", tempat_kerja);
   Log.d("id user", id_user);
   List<NameValuePair> params = new ArrayList<NameValuePair>();
   params.add(new BasicNameValuePair("uid", id_user));
   params.add(new BasicNameValuePair(tag_tempat_kerja, tempat_kerja));

   JSONObject json = parser.makeHttpRequest(url_update_work, "GET",
   params);

   try {
      int success = json.getInt(tag_success);
      Log.d("is_success", success + "");
      Log.d("message", json.getString("message"));
      if(success == 1){
         fragment = new Profil();

         if (fragment != null) {
           FragmentManager fragmentManager = getFragmentManager();

           fragmentManager.beginTransaction().replace(R.id.frame_containe
           r, fragment).commit();
         } else {
            Log.e("MainActivity", "Error in creating fragment");
         }
       }
   } catch (JSONException e) {
   .
.
```

Gambar 4.48. Kode program pilih tempat bekerja

Gambar 4.48. merupakan kode yang digunakan untuk melakukan proses pilih tempat bekerja. Setelah pengguna melihat daftar tempat kerja yang sebelumnya telah disajikan oleh sistem, pengguna bisa memilih salah satu tempat bekerja pada *spinner*, maka sistem akan memperbarui data pengguna yang berada

pada basis data. Data yang diperlukan untuk memperbarui data pengguna yaitu id pengguna dan id tempat bekerja. Sistem akan mengirimkan data tersebut ke *server* yang nantinya akan digunakan sebagai parameter untuk memperbarui data pengguna yang berada pada basis data. Jika *server* mengirimkan respon sukses dengan nilai 1 menandakan bahwa proses memperbarui data pengguna berhasil dilakukan dan sistem akan menampilkan profil pengguna. Sdeangkan jika *server* mengirimkan respon sukses dengan nilai 0, maka sistem akan menampilkan pesan kesalahan pada pengguna.

4.1.7.11. Kode Program Lihat Profil

Berikut ini merupakan potongan program untuk melihat profil yang akan dijelaskan pada Gambar 4.49.

```
.
.
protected JSONObject doInBackground(String... args) {
   List<NameValuePair> params = new ArrayList<NameValuePair>();
   params.add(new BasicNameValuePair("id_user", id_user));
   JSONObject json = parser.makeHttpRequest(url_get_profil, "GET",
   params);

   try {
    if (json.getString(tag_success) != null) {
       int success = json.getInt(tag_success);
       if(success == 1){
              profil = json.getJSONArray(tag_profile);

              data = profil.getJSONObject(0);
       } else{
              Toast.makeText(getActivity(), "Error occured in loading
              profile", Toast.LENGTH_SHORT).show();
       }
    }
   } catch (JSONException e) {
    .
.
```

Gambar 4.49. Kode program lihat profil

Gambar 4.49. merupakan kode yang digunakan untuk menampilkan profil pengguna. Data yang diperlukan untuk proses menampilkan profil pengguna yaitu id penggunadan id tempat bekerja. Sistem akan mengirimkan data tersebut ke *server* sebagai parameter untuk melakukan pencarian data profil

pengguna pada basis data. Jika *server* mengirimkan respon sukses dengan nilai 1 menandakan bahwa data ditemukan dan sistem akan menampilkan profil pengguna. Sedangkan jika *server* mengirimkan respon sukses dengan nilai 0, maka sistem akan menampilkan pesan kesalahan pada pengguna.

4.1.7.12. Kode Program Lihat Konten Keperawatan

Implementasi kode program lihat konten keperawatan dibagi menjadi 4, yaitu obat, penyakit, pemeriksaan diagnostik, dan anatomi.

1. Obat

Berikut ini merupakan potongan program untuk melihat daftar obat yang akan dijelaskan pada Gambar 4.50.

```
.
.
lv.setOnItemClickListener(new OnItemClickListener() {

   @Override
   public void onItemClick(AdapterView<?> arg0, View arg1, int arg2, long
   arg3) {
       Intent in = new Intent(getApplicationContext(), DetailObat.class);
       String get_id_obat = ((TextView)
       view.findViewById(R.id.id_drug)).getText().toString();
       String get_nama_obat = ((TextView)
       view.findViewById(R.id.nama_drug)).getText().toString();
       String get_komposisi = ((TextView)
       view.findViewById(R.id.komposisi)).getText().toString();
       String get_indikasi = ((TextView)
       view.findViewById(R.id.indikasi)).getText().toString();
       String get_dosis = ((TextView)
       view.findViewById(R.id.dosis)).getText().toString();
       String get_penyajian = ((TextView)
       view.findViewById(R.id.penyajian)).getText().toString();
       String get_kontra = ((TextView)
       view.findViewById(R.id.kontra)).getText().toString();
       String get_perhatian = ((TextView)
       view.findViewById(R.id.perhatian)).getText().toString();
       String get_reaksi = ((TextView)
       view.findViewById(R.id.reaksi)).getText().toString();
       String get_interaksi = ((TextView)
       view.findViewById(R.id.interaksi)).getText().toString();
       in.putExtra(tag_id_obat, get_id_obat);
       in.putExtra(tag_nama_obat, get_nama_obat);
       in.putExtra(tag_komposisi, get_komposisi);
       in.putExtra(tag_indikasi, get_indikasi);
       in.putExtra(tag_dosis, get_dosis);
       in.putExtra(tag_penyajian, get_penyajian);
       in.putExtra(tag_kontra, get_kontra);
       in.putExtra(tag_perhatian, get_perhatian);
       in.putExtra(tag_reaksi, get_reaksi);
       in.putExtra(tag_interaksi, get_interaksi);
```

```
        startActivity(in);
    }
});
.
.
protected ArrayList<Note> doInBackground(String... args) {
    // TODO Auto-generated method stub
    List<NameValuePair> params = new ArrayList<NameValuePair>();
    JSONObject json = parser.makeHttpRequest(url, "GET", params);
    try {
        int success = json.getInt(tag_success);
        if(success == 1){
            obat = json.getJSONArray(tag_obat);
            for(int i = 0; i < obat.length(); i++){
                JSONObject data = obat.getJSONObject(i);
                String id_obat = data.getString(tag_id_obat);
                String nama_obat = data.getString(tag_nama_obat);
                String komposisi = data.getString(tag_komposisi);
                String indikasi = data.getString(tag_indikasi);
                String dosis = data.getString(tag_dosis);
                String penyajian = data.getString(tag_penyajian);
                String kontra = data.getString(tag_kontra);
                String perhatian = data.getString(tag_perhatian);
                String reaksi = data.getString(tag_reaksi);
                String interaksi = data.getString(tag_interaksi);

                Drugs drug = new Drugs();

                drug.setIdObat(id_obat);
                drug.setNamaObat(nama_obat);
                drug.setKomposisi(komposisi);
                drug.setIndikasi(indikasi);
                drug.setDosis(dosis);
                drug.setPenyajian(penyajian);
                drug.setKontra(kontra);
                drug.setPerhatian(perhatian);
                drug.setReaksi(reaksi);
                drug.setInteraksi(interaksi);

                drug_list.add(drug);
    .
    .
}
```

Gambar 4.50. Kode program lihat obat

Pada gambar 4.50. merupakan kode yang digunakan untuk menampilkan daftar obat. Sistem akan meminta *server* untuk mendapatkan data obat yang berada pada basis data. Jika *server* mengirimkan respon sukses dengan nilai 1 menandakan bahwa data ditemukan dan sistem akan menampilkan daftar obat. Sedangkan jika *server* mengirimkan respon sukses dengan nilai 0, maka sistem akan menampilkan pesan kesalahan pada pengguna.

2. Penyakit

Berikut ini merupakan potongan program untuk melihat daftar penyakit yang akan dijelaskan pada Gambar 4.51.

```
.
.
lv.setOnItemClickListener(new OnItemClickListener() {

      @Override
      public void onItemClick(AdapterView<?> arg0, View arg1, int arg2,
      long arg3) {
          Intent in = new Intent(getApplicationContext(),
          DetailPenyakit.class);
          String get_id_penyakit = ((TextView)
          view.findViewById(R.id.id_penyakit)).getText().toString();
          String get_nama_penyakit = ((TextView)
          view.findViewById(R.id.nama_penyakit)).getText().toString();
          String get_definisi = ((TextView)
          view.findViewById(R.id.definisi)).getText().toString();
          String get_patofisiologi = ((TextView)
          view.findViewById(R.id.patofisiologi)).getText().toString();
          String get_gambaran_klinis = ((TextView)
          view.findViewById(R.id.gambaran_klinis)).getText().toString();
          in.putExtra(tag_id_penyakit, get_id_penyakit);
          in.putExtra(tag_nama_penyakit, get_nama_penyakit);
          in.putExtra(tag_definisi, get_definisi);
          in.putExtra(tag_patofisiologi, get_patofisiologi);
          in.putExtra(tag_gambaran_klinis, get_gambaran_klinis);
          startActivity(in);
      }
});
.
.
protected ArrayList<Note> doInBackground(String... args) {
    List<NameValuePair> params = new ArrayList<NameValuePair>();
    JSONObject json = parser.makeHttpRequest(url, "GET", params);
    try {
      int success = json.getInt(tag_success);
      if(success == 1){
          penyakit = json.getJSONArray(tag_penyakit);
          for(int i = 0; i < penyakit.length(); i++){
             JSONObject data = penyakit.getJSONObject(i);
             String id_penyakit = data.getString(tag_id_penyakit);
             String nama_penyakit = data.getString(tag_nama_penyakit);
             String definisi = data.getString(tag_definisi);
             String patofisiologi = data.getString(tag_patofisiologi);
             String gambaran_klinis =
             data.getString(tag_gambaran_klinis);

             Disease disease = new Disease();

             disease.setIdPenyakit(id_penyakit);
             disease.setNamaPenyakit(nama_penyakit);
             disease.setDefinisi(definisi);
             disease.setPatofisiologi(patofisiologi);
             disease.setGambaranKlinis(gambaran_klinis);

             disease_list.add(disease);
```

```
.
.
}
```

Gambar 4.51. Kode program lihat penyakit

Gambar 4.51. merupakan kode yang digunakan untuk menampilkan daftar penyakit. Sistem akan meminta *server* untuk mendapatkan data penyakit yang berada pada basis data. Jika *server* mengirimkan respon sukses dengan nilai 1 menandakan bahwa data ditemukan dan sistem akan menampilkan daftar penyakit. Sedangkan jika *server* mengirimkan respon sukses dengan nilai 0, maka sistem akan menampilkan pesan kesalahan pada pengguna.

3. Pemeriksaan Diagnostik

Berikut ini merupakan potongan program untuk melihat daftar pemeriksaan diagnostik berdasarkan nama penyakit yang akan dijelaskan pada Gambar 4.52.

```
.
.
lv.setOnItemClickListener(new OnItemClickListener() {

      @Override
      public void onItemClick(AdapterView<?> arg0, View arg1, int arg2,
      long arg3) {
            Intent in = new Intent(getApplicationContext(),
            DetailDiagnostik.class);

            String get_id_penyakit = ((TextView)
            view.findViewById(R.id.id_penyakit)).getText().toString();
            String get_nama_penyakit = ((TextView)
            view.findViewById(R.id.nama_penyakit)).getText().toString();
            String get_tes_diagnostik = ((TextView)
            view.findViewById(R.id.tes_diagnostik)).getText().toString();
            String get_penanganan = ((TextView)
            view.findViewById(R.id.penanganan)).getText().toString();
            in.putExtra(tag_id_penyakit, get_id_penyakit);
            in.putExtra(tag_nama_penyakit, get_nama_penyakit);
            in.putExtra(tag_tes_diagnostik, get_tes_diagnostik);
            in.putExtra(tag_penanganan, get_penanganan);
            startActivity(in);
      }
});
.
.
protected ArrayList<Note> doInBackground(String... args) {
    List<NameValuePair> params = new ArrayList<NameValuePair>();
    JSONObject json = parser.makeHttpRequest(url, "GET", params);
    try {
      int success = json.getInt(tag_success);
```

```
        if(success == 1){
            penyakit = json.getJSONArray(tag_penyakit);
            for(int i = 0; i < penyakit.length(); i++){
                JSONObject data = penyakit.getJSONObject(i);
                String id_penyakit = data.getString(tag_id_penyakit);
                String nama_penyakit = data.getString(tag_nama_penyakit);
                String tes_diagnostik = data.getString(tag_tes_diagnostik);
                String penanganan = data.getString(tag_penanganan);

                Disease disease = new Disease();

                disease.setIdPenyakit(id_penyakit);
                disease.setNamaPenyakit(nama_penyakit);
                disease.setTesDiagnostik(tes_diagnostik);
                disease.setPenanganan(penanganan);

                disease_list.add(disease);
.
.
}
```

Gambar 4.52. Kode program lihat pemeriksaan diagnostik

Gambar 4.52. merupakan kode yang digunakan untuk menampilkan daftar pemeriksaan diagnostik berdasarkan nama penyakit. Sistem akan meminta *server* untuk mendapatkan data pemeriksaan diagnostik berdasarkan nama penyakit yang berada pada basis data. Jika *server* mengirimkan respon sukses dengan nilai 1 menandakan bahwa data ditemukan dan sistem akan menampilkan daftar pemeriksaan diagnostik berdasarkan nama penyakit. Sedangkan jika *server* mengirimkan respon sukses dengan nilai 0, maka sistem akan menampilkan pesan kesalahan pada pengguna.

4. Anatomi

Berikut ini merupakan potongan program untuk melihat daftar anatomi yang akan dijelaskan pada Gambar 4.53.

```
.
.
lv.setOnItemClickListener(new OnItemClickListener() {

   @Override
   public void onItemClick(AdapterView<?> arg0, View arg1, int arg2, long
   arg3) {
         Intent in = new Intent(getApplicationContext(),
         DetailAnatomi.class);

         String get_id_anatomi = ((TextView)
         view.findViewById(R.id.id_anatomi)).getText().toString();
         String get_nama_anatomi = ((TextView)
```

```
        view.findViewById(R.id.nama_anatomi)).getText().toString();
        String get_definisi = ((TextView)
        view.findViewById(R.id.definisi_anatomi)).getText().toString();

        in.putExtra(tag_id_anatomi, get_id_anatomi);
        in.putExtra(tag_nama_anatomi, get_nama_anatomi);
        in.putExtra(tag_definisi, get_definisi);
        startActivity(in);
    }
});
.
.
protected ArrayList<Note> doInBackground(String... args) {
    List<NameValuePair> params = new ArrayList<NameValuePair>();
    JSONObject json = parser.makeHttpRequest(url, "GET", params);
        try {
          int success = json.getInt(tag_success);
          if(success == 1){
             anatomi = json.getJSONArray(tag_anatomi);
             for(int i = 0; i < anatomi.length(); i++){
                JSONObject data = anatomi.getJSONObject(i);
                String id_anatomi = data.getString(tag_id_anatomi);
                String nama_anatomi = data.getString(tag_nama_anatomi);
                String definisi = data.getString(tag_definisi);

                Anfis anfis = new Anfis();

                anfis.setIdSnatomi(id_anatomi);
                anfis.setNamaAnatomi(nama_anatomi);
                anfis.setDefinisi(definisi);

                anfis_list.add(anfis);
.
.
```

Gambar 4.53. Kode program lihat anatomi

Gambar 4.53. merupakan kode yang digunakan untuk menampilkan daftar anatomi. Sistem akan meminta *server* untuk mendapatkan data anatomi yang berada pada basis data Jika *server* mengirimkan respon sukses dengan nilai 1 menandakan bahwa data ditemukan dan sistem akan menampilkan daftar anatomi. Sedangkan jika *server* mengirimkan respon sukses dengan nilai 0, maka sistem akan menampilkan pesan kesalahan pada pengguna.

4.1.7.13. Kode Program Lihat Detail Konten Keperawatan

Implementasi kode program lihat detail konten keperawatan dibagi menjadi 4, yaitu detail obat, detail penyakit, detail pemeriksaan diagnostik, dan detail anatomi.

1. Detail Obat

Berikut ini merupakan potongan program untuk melihat detail obat yang akan dijelaskan pada Gambar 4.54.

```
.
.
Intent in = getIntent();
get_id_obat = in.getStringExtra(tag_id_obat);
get_nama_obat = in.getStringExtra(tag_nama_obat);
get_komposisi = in.getStringExtra(tag_komposisi);
get_dosis = in.getStringExtra(tag_dosis);
get_penyajian = in.getStringExtra(tag_penyajian);
get_indikasi = in.getStringExtra(tag_indikasi);
get_kontra = in.getStringExtra(tag_kontra);
get_perhatian = in.getStringExtra(tag_perhatian);
get_reaksi = in.getStringExtra(tag_reaksi);
get_interaksi = in.getStringExtra(tag_interaksi);
.
.
@Override
protected JSONObject doInBackground(String... args) {
   id_obat.setText(get_id_obat);
   nama_obat.setText(get_nama_obat);
   komposisi.setText(get_komposisi);
   indikasi.setText(get_indikasi);
   dosis.setText(get_dosis);
   penyajian.setText(get_penyajian);
   kontra.setText(get_kontra);
   perhatian.setText(get_perhatian);
   reaksi.setText(get_reaksi);
   interaksi.setText(get_interaksi);
}
```

Gambar 4.54. Kode program lihat detail obat

Gambar 4.54. merupakan kode yang digunakan untuk menampilkan detail obat. Data yang diperlukan untuk proses melihat detail obat, yaitu id obat yang diperoleh dari daftar obat yang dipilih oleh pengguna sebelumnya. Sistem akan mengirimkan data tersebut ke *server* sebagai parameter dalam mencari detail obat pada basis data. Jika *server* mengirimkan respon sukses dengan nilai 1 menandakan bahwa data ditemukan dan sistem akan menampilkan detail obat. Sedangkan jika *server* mengirimkan respon sukses dengan nilai 0, maka sistem akan menampilkan pesan kesalahan pada pengguna.

2. Detail Penyakit

Berikut ini merupakan potongan program untuk melihat detail penyakit yang akan dijelaskan pada Gambar 4.55.

```
.
.
Intent in = getIntent();
get_id_penyakit = in.getStringExtra(tag_id_penyakit);
get_nama_penyakit = in.getStringExtra(tag_nama_penyakit);
get_definisi = in.getStringExtra(tag_definisi);
get_patofisiologi = in.getStringExtra(tag_patofisiologi);
get_gambaran_klinis = in.getStringExtra(tag_gambaran_klinis);
.
.
@Override
protected JSONObject doInBackground(String... args) {
   id_penyakit.setText(get_id_penyakit);
   nama_penyakit.setText(get_nama_penyakit);
   definisi.setText(get_definisi);
   patofisiologi.setText(get_patofisiologi);
   gambaran_klinis.setText(get_gambaran_klinis);
}
```

Gambar 4.55. Kode program lihat detail penyakit

Gambar 4.55. merupakan kode yang digunakan untuk menampilkan detail penyakit. Data yang diperlukan untuk proses melihat detail penyakit, yaitu id penyakit yang diperoleh dari daftar penyakit yang dipilih oleh pengguna sebelumnya. Sistem akan mengirimkan data tersebut ke *server* sebagai parameter dalam mencari detail penyakit pada basis data. Jika *server* mengirimkan respon sukses dengan nilai 1 menandakan bahwa data ditemukan dan sistem akan menampilkan detail penyakit. Sedangkan jika *server* mengirimkan respon sukses dengan nilai 0, maka sistem akan menampilkan pesan kesalahan pada pengguna.

3. Detail Pemeriksaan Diagnostik

Berikut ini merupakan potongan program untuk melihat detail pemeriksaan diagnostik yang akan dijelaskan pada Gambar 4.56.

```
.
.
Intent in = getIntent();
get_id_penyakit = in.getStringExtra(tag_id_penyakit);
get_nama_penyakit = in.getStringExtra(tag_nama_penyakit);
get_tes_diagnostik = in.getStringExtra(tag_tes_diagnostik);
```

```
get_penanganan = in.getStringExtra(tag_penanganan);
.
.
@Override
protected JSONObject doInBackground(String... args) {
   id_penyakit.setText(get_id_penyakit);
   nama_penyakit.setText(get_nama_penyakit);
   tes_diagnostik.setText(get_tes_diagnostik);
   penanganan.setText(get_penanganan);
}
```

Gambar 4.56. Kode program lihat detail pemeriksaan diagnostik

Gambar 4.56. merupakan kode yang digunakan untuk menampilkan detail pemeriksaan diagnostik. Data yang diperlukan untuk proses melihat detail pemeriksaan diagnostik, yaitu id penyakit yang diperoleh dari daftar pemeriksaan diagnostik berdasarkan nama penyakit yang dipilih oleh pengguna sebelumnya. Sistem akan mengirimkan data tersebut ke *server* sebagai parameter dalam mencari detail pemeriksaan diagnostik pada basis data. Jika *server* mengirimkan respon sukses dengan nilai 1 menandakan bahwa data ditemukan dan sistem akan menampilkan detail pemeriksaan diagnostik. Sedangkan jika *server* mengirimkan respon sukses dengan nilai 0, maka sistem akan menampilkan pesan kesalahan pada pengguna.

4. Detail Anatomi

Berikut ini merupakan potongan program untuk melihat detail anatomi yang akan dijelaskan pada Gambar 4.57.

```
.
.
Intent in = getIntent();
get_id_anatomi = in.getStringExtra(tag_id_anatomi);
get_nama_anatomi = in.getStringExtra(tag_nama_anatomi);
get_definisi = in.getStringExtra(tag_definisi);
.
.
@Override
protected JSONObject doInBackground(String... args) {
   id_anatomi.setText(get_id_anatomi);
   nama_anatomi.setText(get_nama_anatomi);
   definisi.setText(get_definisi);
}
```

Gambar 4.57. Kode program lihat detail anatomi

Gambar 4.57. merupakan kode yang digunakan untuk menampilkan detail anatomi. Data yang diperlukan untuk proses melihat detail anatomi, yaitu id anatomi yang diperoleh dari daftar anatomi yang dipilih oleh pengguna sebelumnya. Sistem akan mengirimkan data tersebut ke *server* sebagai parameter dalam mencari detail anatomi pada basis data. Jika *server* mengirimkan respon sukses dengan nilai 1 menandakan bahwa data ditemukan dan sistem akan menampilkan detail anatomi. Sedangkan jika *server* mengirimkan respon sukses dengan nilai 0, maka sistem akan menampilkan pesan kesalahan pada pengguna.