ISSN 2085-4579

Implementasi Self-Service untuk Membantu Calon Pasien Rumah Sakit

Vannia Ferdina[1],Genesius Hartoko Soekoco[2], Junitania Ryanto[3], Hans Rafael[4], Eka Cahya Rahmadhani[5]

Teknik Informatika, Universitas Multimedia Nusantara, Tangerang, Indonesia
nia9508@yahoo.com[1], trinity_cross@live.com[2], jun.junitania@hotmail.com[3], hansrafael95@gmail.com[4], rahmadhani.ekacahya@gmail.com[5]

Diterima 12 Juni 2014
Disetujui 3 November 2014

Abstract— **Nowadays the need for patients to receive prompt and proper care of the hospital plays an important role for the development of the hospital itself. This encourages to the development of various applications and integrated system for enhancing the effectiveness of hospital services, one of which is a self-service method to open floor directory and to enroll oneself with a doctor at the hospital. Therefore, in this paper the authors make the design of an application to support the self-service method. The design of the application is made by using the Unified Modeling Language that is represented by use case and class diagram.**

Index Terms—**UML, use case diagram, class diagram**

I. Pendahuluan

Resepsionis merupakan bagian terdepan dalam berbagai pelayanan di ruang publik, misalnya rumah sakit. Sebagai bagian yang berinteraksi langsung dengan konsumen, resepsionis harus memberikan pelayanan prima dan cepat yang kemudian sangat mempengaruhi kesan pertama yang membentuk persepsi konsumen terhadap penyedia layanan tersebut. Namun, seringkali staf resepsionis tidak mampu memberikan pelayanan terbaiknya sehingga memunculkan keluh kesah konsumen, baik mengenai lambatnya kinerja staf, informasi yang tidak jelas, dan lain-lain. Seiring dengan berkembangnya teknologi informasi dan komunikasi, peningkatan efektifitas dan efisiensi kerja karyawan dapat semakin ditingkatkan. Salah satu teknologi yang kini kian marak digunakan adalah sistem *self-service* atau layanan mandiri. *Self-service* adalah teknologi di mana konsumen dapat melakukan sendiri aktivitas yang biasa dilayani oleh pihak staf dengan cara berinteraksi langsung dengan perangkat yang terkomputerisasi. Teknologi ini dapat diaplikasikan pada bagian resepsionis rumah sakit untuk bentuk-bentuk pelayanan tertentu, misalnya mengenai jadwal praktik dokter, direktori ruang praktik, pengambilan nomor antrian, dan lain-lain. Penerapan teknologi ini dapat mengurangi keluhan konsumen berkaitan dengan kinerja staf resepsionis secara cukup signifikan.

II. Identifikasi Masalah

Dari uraian yang penulis kemukakan pada pendahuluan, maka dapat diidentifikasi masalah-masalah sebagai berikut:

- Direktori Rumah Sakit yang belum terkomputerisasi dapat menyulitkan pasien dalam mencari informasi ruangan yang dibutuhkan dalam waktu cepat.
- Pendaftaran pasien untuk berobat ke dokter yang biasanya dilakukan manual dengan menghubungi resepsionis dapat menyulitkan pasien apabila kondisi rumah sakit dalam keadaan ramai dengan jumlah resepsionis yang terbatas.
- Mencari dokter yang sesuai dengan penyakit atau riwayat kesehatan pasien yang dilakukan secara manual dengan berkomunikasi dengan resepsionis dapat menyulitkan pasien karena terbatasnya jumlah resepsionis.
- Akses ke *database* riwayat kesehatan pasien yang hanya dilakukan oleh staf rumah sakit dapat memperlambat pembaharuan riwayat

kesehatan pasien, misalnya karena data hasil pemeriksaan pasien harus dikumpulkan terlebih dahulu ke staf yang berhubungan langsung dengan dokter kemudian diteruskan kepada staf yang bertugas untuk memperbaharui isi *database*. Proses yang demikian akan mengurangi efektivitas operasi rumah sakit.

III. Pembatasan Masalah

Sistem komputerisasi direktori, registrasi, dan pembaharuan *database* riwayat kesehatan pasien sangat berkaitan dengan sistem-sistem lain di dalam sistem rumah sakit. Oleh karena itu, penulis membatasi masalah hanya pada:

- Perancangan model penampilan direktori rumah sakit
- Perancangan model aplikasi registrasi dan pencarian dokter yang sesuai dengan kebutuhan pasien berdasarkan spesialisasi dokter, daftar penyakit, dan riwayat pemeriksaan pasien di rumah sakit tersebut.

IV. Tujuan Penelitian

Berdasarkan masalah diatas, maka dapat diketahui tujuan penelitian adalah sebagai berikut:

- Untuk melakukan pemodelan aplikasi untuk menampilkan direktori rumah sakit

- Untuk melakukan pemodelan aplikasi untuk registrasi dan pencarian dokter sesuai kebutuhan pasien berdasarkan spesialisasi dokter, daftar penyakit, dan riwayat pemeriksaan pasien di rumah sakit tersebut.

V. Manfaat Penelitian

Adapun manfaat yang dapat diambil dari penelitian ini adalah sebagai berikut:

· Pemodelan aplikasi untuk menggantikan respsionis rumah sakit ini diharapkan dapat meningkatkan efektivitas dan efisiensi dalam operasi rumah sakit.

· Dengan pemodelan aplikasi yang bersifat *self-service* bagi penggunanya diharapkan dapat diterapkan guna meningkatkan kepuasan pasien terhadap pelayanan rumah sakit yang cepat dan tepat.

VI. Metode Perancangan Program

Dalam pemodelan program ini penulis menggunakan *Unified Modeling Language (UML)* dengan menggunakan *use case diagram* dan *class diagram.*

A. Unified Modeling Language Unified

Modeling Language adalah sebuah bahasa yang menggunakan grafik atau gambar untuk memvisualisasi, menspesifikasikan, membangun, dan pendokumentasian dari sebuah sistem pengembangan *software* berbasis *object-oriented*. *UML* adalah sebuah bahasa standar untuk pengembangan sebuah *software* yang dapat menyampaikan bagaimana membuat dan membentuk model-model, tetapi tidak menyampaikan apa dan kapan model yang seharusnya dibuat yang merupakan salah satu proses implementasi pengembangan *software*.

Dalam *Unified Modeling Language* dikenal tiga derajat pembungkusan atau yang lebih dikenal dengan sebutan visibility, yaitu:

· *Private*

Atribut dan operasi yang bersifat *private* tersembunyi didalam *class* sehingga *class-class* lainnya tidak dapat mengakses dan memanfaatkannya. Notasi *private* adalah tanda negatif (-).

· *Public*

Atribut dan operasi yang bersifat *public* bisa diakses dan dimanfaatkan oleh semua *class* yang ada. Notasi *public* adalah tanda positif (+).

· *Protected*

Atribut dan operasi yang bersifat *protected* hanya dapat diakses oleh anggota class yang bersangkutan serta *class-class* lain yang menjadi turunannya dalam hierarki pewarisan (*inheritance*). Notasi *protected* adalah tanda pagar (#).

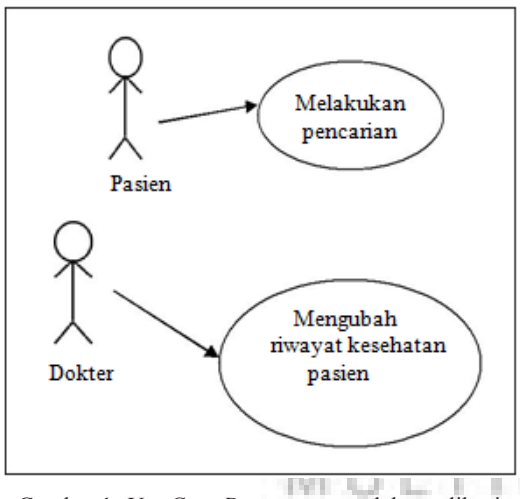
B. Use Case Diagram

Use Case Diagram, menggambarkan sekelompok *use cases* dan aktor yang disertai dengan hubungan diantaranya. Diagram *use cases* ini menjelaskan dan menerangkan kebutuhan atau *requirement* yang diinginkan atau dikehendaki pengguna, serta sangat berguna dalam menentukan struktur organisasi dan model dari pada sebuah sistem. Umumnya *use case* digambarkan dengan elips dan garis yang solid yang biasanya mengandung nama.

C. Class Diagram

Class Diagram, yang memperlihatkan struktur statis dari kelas aktual didalam sistem. *Class Diagram* memiliki tiga komponen, yaitu nama *class*, atribut, dan metode atau operasi.

VII. PERANCANGAN PROGRAM

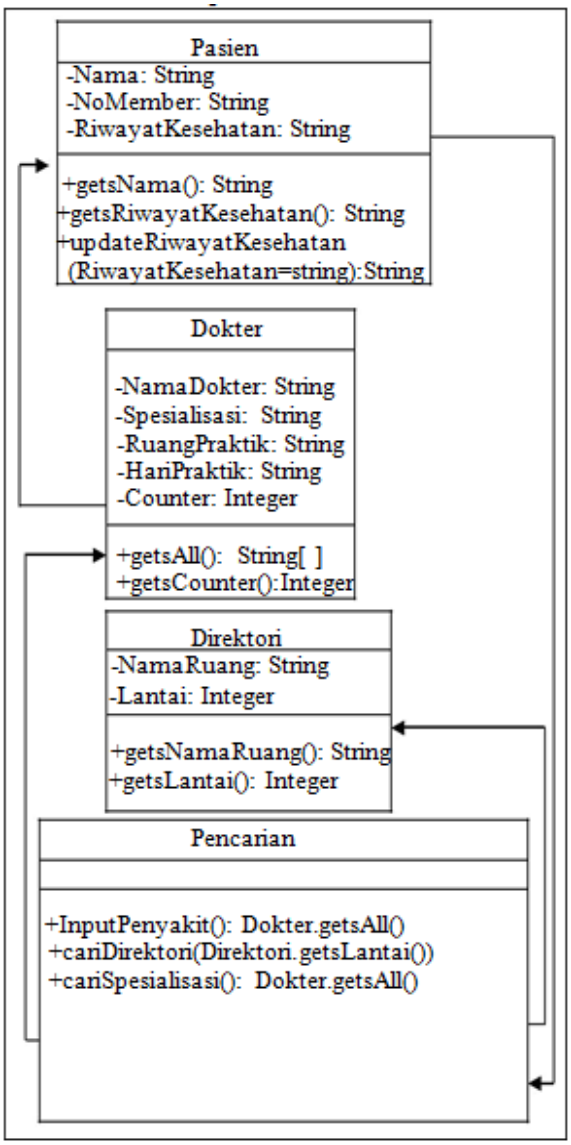


Gambar 1. *Use Case Diagram* pemodelan aplikasi

Use Case Diagram di atas menunjukkan aktor dan aktivitas yang dapat dilakukan dan diselesaikan oleh sistem. Pasien dapat melakukan pencarian, dalam hal ini adalah melihat direktori semua ruangan di setiap lantai rumah sakit dan mencari dokter yang dibutuhkan pasien yang bersangkutan.

Selain itu, pencarian dokter yang dibutuhkan dapat dilakukan berdasarkan beberapa basis pencarian, yaitu berdasarkan spesialisasi dokter, berdasarkan kategori penyakit, dan berdasarkan riwayat pemeriksaan yang tersimpan dalam *database* pasien di rumah sakit tersebut.

Dokter sebagai aktor, dapat melakukan fungsi untuk mengubah riwayat kesehatan pasien secara langsung berdasarkan hasil pemeriksaan yang dilakukannya. Sehingga setiap kali pemeriksaan selesai, dokter dapat langsung mengakses *database* riwayat kesehatan pasien yang bersangkutan untuk memasukkan hasil pemeriksaan saat itu.



Gambar 2. *Class Diagram* pemodelan aplikasi

Pseudocode fungsi getsAll() pada objek dokter:

```
Function getsAll()
    string output[1]=NamaDokter
    string output[2]=Spesialisasi
    string output[3]=RuangPraktik
    string output[4]=HariPraktik
```

string output[5]=JamPraktik
endFunction

Fungsi gets.All() pada objek dokter akan menampilkan seluruh deskripsi dokter yang bersangkutan, yaitu nama, spesialisasi, ruang praktik, hari praktik, dan jam praktik, sebagai output yang diinginkan dari pencarian dokter menggunakan fungsi InputPenyakit(), dan cariSpesialisasi().

Class Diagram di atas menunjukkan hubungan antar *class* dan *visibility* masing-masing atribut dan operasi. Pasien dapat mencari dokter berdasarkan dua cara, yaitu melalui spesialisasi dokter dalam fungsi cariSpesialisasi(), kategori-kategori penyakit melalui fungsi InputPenyakit(), serta riwayat kesehatan pasien.

Semua pencarian dokter baik berdasarkan spesialisasi dokter, kategori penyakit, atau riwayat kesehatan pasien, akan diakhiri dengan menampilkan deskripsi dokter (melalui fungsi getsAll()) yang disesuaikan dengan hari dan jam disaat pasien melakukan pencarian. Artinya, dokter beserta deskripsi yang ditampilkan saat itu hanyalah dokter yang sedang praktik di hari dan jam tersebut, yang dibaca dari sistem. Setelah nama-nama dokter ditampilkan, pasien dapat memilih dokter mana yang dituju untuk memeriksakan dirinya. Setelah memilih dokter, pasien akan diminta untuk memasukkan nomor kartu anggota rumah sakit. Nomor tersebut akan dicocokkan dengan nomor-nomor kartu anggota yang terdaftar di rumah sakit. Jika nomor tersebut ditemukan dalam database rumah sakit, maka akan ditampilkan pesan bahwa pasien berhasil melakukan pendaftaran untuk dokter tersebut. Nilai yang tersimpan dalam fungsi counter() akan bertambah satu dan akan ditampilkan sebagai nomor antrian pasien. Pasien yang tidak memiliki kartu anggota tidak dapat melakukan pendaftaran untuk berobat melalui aplikasi ini. Hal ini dimaksudkan agar penggunaan aplikasi ini terkontrol dan menghindari pihak-pihak yang tidak bertanggungjawab melakukan registrasi untuk pemeriksaan.

Jika pasien memilih melakukan pencarian dokter berdasarkan penyakit, fungsi InputPenyakit() akan menampilkan daftar nama-nama penyakit . Jika pasien tidak menemukan nama penyakit yang mereka derita dalam daftar nama-nama penyakit tersebut ataupun riwayat penyakitnya, pasien akan disarankan mengunjungi dokter umum.

Sistem akan tetap mencari nama-nama dokter umum yang berpraktik dihari tersebut untuk ditampung dalam *array* yang diperuntukkan untuk menampung daftar dokter yang akan direkomendasikan kepada pasien.

Pencarian dokter berdasarkan riwayat kesehatan pasien yang bersangkutan yang tersimpan dalam *database* rumah sakit tersebut sebagai alternatif kedua dilakukan dengan memanggil fungsi getsRiwayatKesehatan() dalam objek pasien. Pasien terlebih dahulu diminta memasukkan nomor anggota mereka. Sistem kemudian membaca dan mengambil nama-nama dokter yang pernah menangani pasien tersebut. Nama-nama tersebur kemudian dicocokkan satu per satu dengan nama-nama dan deskripsi dokter (melalui fungsi getsAll()) yang ada dalam database rumah sakit yang berpraktik di hari tersebut.

Jika pasien menemukan nama penyakit yang mereka derita dalam daftar nama-nama penyakit, pasien kemudian memilih salah satu dari daftar nama penyakit tersebut. Fungsi akan mencocokkan spesialisasi dokter untuk penyakit tersebut dengan spesialisasi dokter yang berpraktik di hari dan jam tersebut dan kemudian menampilkan rekomendasi dokter beserta dengan nama dan deskripsi dokter tersebut melalui fungsi getsAll(). Setelah muncul nama dan keterangan dokter-dokter yang direkomendasikan, pasien dapat memilih dokter mana yang akan dituju untuk melakukan pemeriksaan.

Pencarian dokter yang dibutuhkan pasien berdasarkan riwayat kesehatan pasien yang tersimpan dalam database rumah sakit tersebut membutuhkan pembaharuan database riwayat kesehatan secara cepat dan berkesinambungan. Oleh karena itu, objek dokter dapat melakukan fungsi untuk memperbaharui database secara langsung sesudah melakukan pemeriksaan terhadap pasien melalui fungsi UpdateRiwayat Kesehatan(RiwayatKesehatan=String) yang memiliki parameter berupa riwayat kesehatan pasien sebelumnya (RiwayatKesehatan).

Fungsi cariSpesialisasi() berfungsi untuk menampilkan daftar pilihan-pilihan spesialisasi dokter yang tersedia. Pasien kemudian memilih

salah satu kategori spesialisasi yang diinginkan, kemudian akan ditampilkan semua nama dokter dengan spesialisasi yang telah dipilih yang praktik di hari tersebut. Kemudian pasien dapat memilih nama dokter di mana pasien akan memeriksakan diri.

Sedangkan menu untuk melihat direktori atau peta rumah sakit dapat diakses oleh seluruh pengguna tanpa perlu memiliki kartu anggota rumah sakit. Jika pengguna memilih menu ini, maka pengguna akan diminta untuk memilih denah lantai berapa yang pengguna ingin ketahui, menggunakan fungsi cariDirektori(Direktori. getsLantai()). Input pengguna tersebut kemudian akan digunakan untuk menampilkan lantai (melalui fungsi getsLantai()). Setelah memilih lantai, aplikasi akan menampilkan seluruh ruangan di lantai tersebut (melalui fungsi getsRuang()). Direktori rumah sakit tidak menampilkan detail setiap ruang praktik, misalnya nama dokter yang praktik di ruangan tersebut. Direktori rumah sakit akan menampilkan blok-blok ruangan rumah sakit untuk masing-masing spesialisasi dokter yang sudah tersimpan di dalam *database* denah rumah sakit.

VIII. Pengujian Rancangan Program

Berdasarkan perancangan program yang penulis rancang, penulis melakukan pengujian dengan metode *desk-checking* untuk setiap fungsi sebagai berikut :

A. *Pengujian Fungsi Internal Masing-Masing Class*

Tabel 1. Pengujian fungsi internal setiap *class*

No	*Class*	Fungsi	Parameter	Return	Status
1	Pasien	getsNama	-	Nama	√
		getsRiwayat Kesehatan	-	RiwayatKese-hatan	√
2.	Dokter	getsAll	-	NamaDokter	√
				Spesialisasi	√
				RuangPraktik	√
				HariPraktik	√
				JamPraktik	√
		getsCounter	-	Counter	√
3.	Direktori	getsNama Ruang	-	NamaRuang	√
		getsLantai	-	Lantai	√

Pengujian fungsi internal masing-masing *class* dimaksudkan untuk menguji fungsi-fungsi tertentu pada masing-masing *class* yang banyak digunakan dalam fungsi-fungsi yang berhubungan dengan *class* lain untuk menampilkan atribut-atribut dalam *class* itu sendiri. Status menunjukkan apakah fungsi tersebut mengembalikan nilai atau atribut yang diminta. Tanda "√" menunjukkan bahwa fungsi tersebut berjalan dengan benar.

Class Pasien memiliki fungsi getsNama tanpa parameter untuk menampilkan atribut Nama, dan fungsi getsRiwayatKesehatan untuk menampilkan atribut RiwayatKesehatan dalam fungsi tersebut. Sedangkan fungsi Update Riwayat Kesehatan (Riwayat Kesehatan=String) akan diuji pada hubungan antar fungsi pada tabel 2.

Class Dokter memiliki fungsi getsAll tanpa parameter untuk menampilkan atribut NamaDokter, Spesialisasi, RuangPraktik, HariPraktik, dan JamPraktik, serta fungsi getsCounter untuk menampilkan atribut Counter dalam fungsi tersebut.

Class Direktori memiliki fungsi getsNamaRuang tanpa parameter untuk menampilkan atribut NamaRuang, dan fungsi getsLantai untuk menampilkan atribut Lantai dalam fungsi tersebut.

Class Pencarian memiliki 4 fungsi yang semuanya menunjuk pada *class* lain. Oleh karena itu, pengujian fungsi-fungsi didalamnya dicantumkan dalam tabel 2.

B. *Pengujian Hubungan Fungsi Antar Class*

Tabel 2. Pengujian hubungan fungsi antar *class*

Class	Fungsi	Parameter	Return	Status
Pasien	Pencarian.InputPenyakit	-	Dokter.getsAll	√
	Pencarian.cariDirektori	Direktori. getsLantai	Direktori. getsNamaRuang()	√
	Pencarian.cariSpesialisasi	-	Dokter.getsAll	√
Pencarian	Direktori.getsLantai	-	Direktori. getsNamaRuang()	√
	Dokter.getsAll	-	NamaDokter	√
			Spesialisasi	√
			RuangPraktik	√
			HariPraktik	√
			JamPraktik	√
Dokter	Pasien. UpdateRiwayatKesehatan	Pasien. Riwayat Kesehatan	Pasien. RiwayatKesehatan	√

Pengujian hubungan fungsi antar *class* dilakukan untuk mengetahui apakah interaksi fungsi antar *class* berjalan dengan benar. Status menunjukkan apakah fungsi tersebut mengembalikan nilai atau atribut yang diminta. Tanda "√" menunjukkan bahwa fungsi tersebut berjalan dengan benar.

Class Pasien memiliki hubungan dengan *class* Pencarian yaitu menuju fungsi InputGejala(), InputPenyakit(), cariDirektori(Direktori. getsLantai()), dan cariSpesialisasi(). Fungsi InputGejala, InputPenyakit, dan cariSpesialisasi tidak memiliki parameter dan ketiganya mengembalikan nilai berupa fungsi getsAll() pada objek Dokter. Fungsi getsAll() pada objek dokter telah diuji pada tabel 1.

Fungsi cariDirektori memiliki parameter fungsi getsLantai() pada objek Direktori dan mengembalikan nilai berupa fungsi gets. NamaRuang pada objek Direktori pula. Fungsi getsLantai() dan getsNamaRuang() pada objek Direktori telah diuji pada tabel 1.

Class Pencarian memiliki hubungan dengan *class* Direktori (yaitu menuju fungsi getsLantai() dan getsNamaRuang()), dan dengan *class* Dokter (yaitu menuju fungsi getsAll()). Fungsi getsLantai() dan getsNamaRuang() pada objek Direktori telah diuji pada tabel 1, serta fungsi getsAll() pada objek Dokter juga telah diuji pada tabel 1.

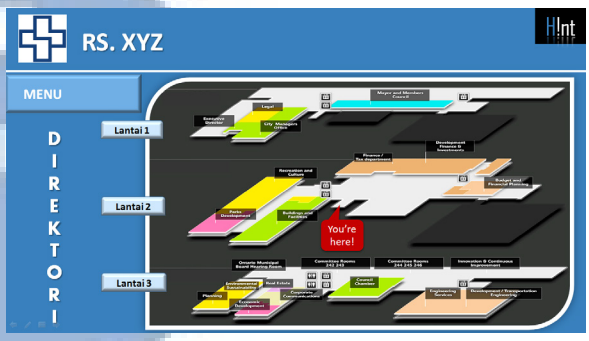
Class Dokter memiliki hubungan dengan *class* Pasien, yaitu menuju fungsi UpdateRiwayatKesehatan dengan parameter RiwayatKesehatan. Fungsi UpdateRiwayatKesehatan akan mengembalikan nilai berupa RiwayatKesehatan pada *class* Pasien tersebut.

Semua hubungan fungsi antar *class* memiliki status "√" yang berarti semuanya dapat berjalan dan mengembalikan nilai sebagaimana mestinya.

C. Graphical User Interface

Berdasarkan program yang penulis rancang, penulis juga menyertakan tampilan tatap muka, sebagai gambaran sistem yang dirancang.

Gambar 3. Tampilan menu utama



Gambar 4. Tampilan direktori

Bagian direktori digunakan untuk melihat lokasi terkini pengguna, dan direktori ruangan dari rumah sakit

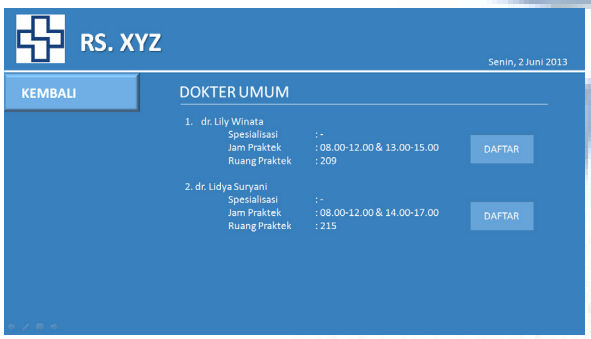
Bagian pencarian dokter terdiri dari 3 pilihan, yaitu pencarian berdasarkan spesialisasi, yang bila diklik akan muncul daftar dokter spesialisasi tertentu yang berjaga pada hari itu saja, berdasarkan jenis penyakit tertentu, dan berdasarkan jadwal dokter yang berpraktik di hari tersebut.

Gambar 5. Tampilan pencarian dokter berdasarkan spesialisasi

Gambar 6. Tampilan pencarian dokter berdasarkan jenis penyakit

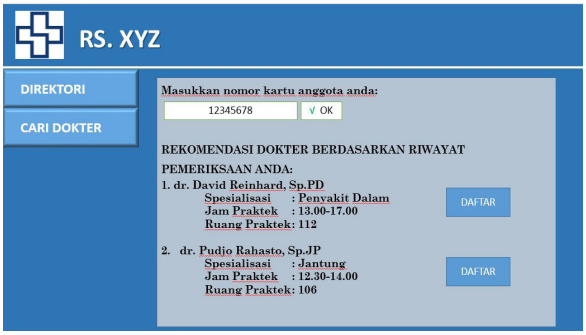
Pencarian dokter berdasarkan jadwal saja memiliki tampilan yang sama dengan pencarian dokter berdasarkan spesialisasi maupun berdasarkan jenis penyakit tertentu. Dalam fitur ini akan ditampilkan seluruh dokter yang berpraktik di hari tersebut.

Jika pasien memilih salah satu dokter pada tombol spesialisasi maupun penyakit, akan muncul pilihan dokter dalam kategori tersebut yang berpraktik pada hari itu dan pasien dapat melakukan pendaftaran seperti pada gambar di bawah ini.



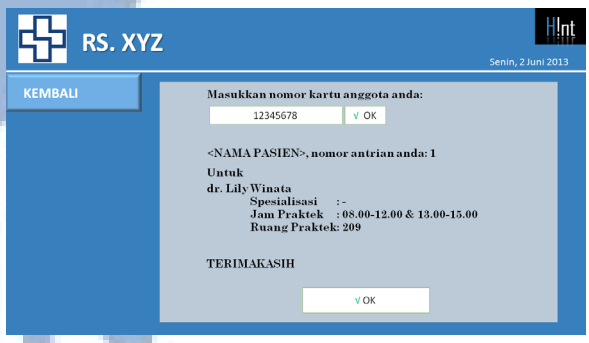
Gambar 7. Tampilan daftar dokter untuk kategori tertentu (contoh: dokter umum)

Dalam fitur pencarian berdasarkan jenis penyakit, terdapat pilihan untuk melakukan pencarian berdasarkan riwayat kesehatan pasien yang bersangkutan. Pasien terlebih dahulu diminta untuk memasukkan nomor anggota mereka untuk selanjutnya dilakukan pencarian terhadap nama-nama dokter yang pernah menangani pasien tersebut.



Gambar 8. Tampilan rekomendasi dokter berdasarkan riwayat kesehatan pasien

Ketika pasien menekan tombol DAFTAR, pasien akan melihat tampilan untuk memasukkan nomor anggota mereka, kecuali untuk pencarian berdasarkan riwayat kesehatan (karena sebelumnya sudah diminta memasukkan nomor anggota).



Gambar 9. Tampilan pendaftaran

Pada bagian pendaftaran ini, pasien diarahkan untuk memasukkan nomor kartu anggota untuk verifikasi data pasien, dan akan diberitahu nomor antriannya.

D. Pengujian Integrasi dengan Enterprise Resource Planning Rumah Sakit

Pengujian dilakukan dengan mengambil referensi dari sistem yang sudah dibuat dan diimplementasikan oleh PatientWay. PatientWay adalah perusahaan pengembang teknologi *self-service* untuk pasien rumah sakit yang kurang lebih serupa dengan aplikasi kami.

Sistem hanya dihubungkan dengan bagian database rumah sakit, karena aplikasi ini ditujukan untuk menggantikan fungsi resepsionis dan kemudahan untuk mengupdate riwayat pasien

oleh dokter, yang berhubungan dengan database.

Bagian pendaftaran adalah komponen kunci dari infrastruktur TI sebuah rumah sakit dan akan perlu diawasi oleh Departemen TI. PatientWay telah memastikan bahwa dampak kios pendaftaran minimal.

Bantuan departemen TI terbatas pada daftar berikut:

- Pasokan akses jaringan Ethernet
- Penyediaan server, termasuk sistem operasi, untuk mengelola Kios. Sebuah server virtual dapat diterima selama kinerja tidak terpengaruh oleh proses lainnya.
- Penyediaan akun pengguna di sistem ADT untuk server kios (dan staf PatientWay untuk pengujian). ini
- perlu dikonfigurasi untuk akses simultan.

E. Pengujian Kelayakan Sistem

Pengujian dilakukan dengan mengambil referensi dari sistem yang sudah dibuat dan diimplementasikan oleh PatientWay. PatientWay adalah perusahaan pengembang teknologi *self-service* untuk pasien rumah sakit yang kurang lebih serupa dengan aplikasi kami.

Sistem PatientWay telah diimplementasikan di banyak badan kesehatan di Amerika Serikat, salah satunya adalah di *Southlake Regional Health Centre*. Sistem ini telah berjalan di pusat kesehatan ini selama 1 tahun dan memberikan dampak poitif yang nyata, yaitu:

- Pengurangan 30% pada staf pendaftaran
- 50% pengurangan kesalahan data pendaftaran
- Pengurangan 60% dalam waktu untuk melihat petugas pendaftaran
- 72% serapan oleh pasien
- $ 400K penghematan operasional tahunan
- Penghematan waktu yang cukup signifikan untuk check-in ke rumah sakit

IX. KESIMPULAN

Dari pengujian rancangan aplikasi yang telah dilakukan, dapat disimpulkan bahwa semua hubungan antar *class*, fungsi, dan atributnya berjalan dengan benar.Aplikasi dapat digunakan untuk membuka denah rumah sakit per lantai dan melakukan pencarian dan pendaftaran untuk pemeriksaan kepada dokter.

Aplikasi ini juga sudah dilengkapi dengan *Graphical User Interface* yang *user-friendly*, integrasi yang minimal dengan sistem manajemen rumah sakit, dan memberikan dampak positif terhadap rumah sakit secara signifikan.

Oleh karena itu, rancangan atau pemodelan aplikasi untuk membantu resepsionis rumah sakit dinyatakan berhasil dan berjalan sebagaimana-mestinya.

UCAPAN TERIMA KASIH

Penulis berterima kasih kepada Bapak Yustinus Eko Soelistio, S.Kom., M.M. selaku dosen Metode Perancangan Program yang telah mendampingi penulis dalam menyelesaikan penelitian ini dan kepada tim penilai jurnal Fakultas TIK Universitas Multimedia Nusantara yang telah memberikan panduan dalam memperbaiki hasil penelitian ini.

DAFTAR PUSTAKA

[1] Robertson, Lesley Anne, *Simple Program Design Step-by-Step Approach*, Fifth Ed, Sydney: Nelson Australia Pty Limited, 2007.

[2] Aji, Bayu, "Unified Modeling Languange (UML)", *Gunadarma e-Learning*, pp 1-15, 10 April 2014, http://bayuaji.staff.gunadarma.ac.id/Downloads/files/32096/UML.pdf

[3] Lawrence, Jay, "Case Study: Hospital generates $400K in savings, reduces registration staff by 30% with patient self-service technologies", 9 November 2014, http://www.patientway.com/case-study-hospital-generates-400k-savings-reduces-registration-staff-30-patient-selfservice-technologies

[4] PatientWay, "Impact on your IT Department", 9 November 2014, http://www.patientway.com/wp-content/uploads/2013/08/Impact-on-your-IT-Department-130313.pdf