PENGEMBANGAN APLIKASI PERANGKAT BERGERAK: PENCARI KONSELOR PSIKOLOGI TERDEKAT BERBASIS LOKASI

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Tifo Audi Alif Putra
NIM: 165150200111168

PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2019

DAFTAR ISI

DAFTAR ISI ii
DAFTAR TABEL v
DAFTAR GAMBAR vi
DAFTAR LAMPIRAN vii
BAB 1 PENDAHULUAN 1
1.1 Latar Belakang 1
1.2 Rumusan Masalah 2
1.3 Tujuan 2
1.4 Manfaat 2
1.5 Batasan Masalah 2
1.6 Sistematika Pembahasan 2
BAB 2 LANDASAN KEPUSTAKAAN 4
2.1 Kajian Pustaka 4
2.2 Mobile Application 4
2.2.1 iOS 4
2.3 Iterative Waterfall Model 5
2.4 Location-Based Service 6
2.5 Best Practices 7
2.5.1 Swift 7
2.5.2 MVVM (Model-View-ViewModel) 7
2.5.3 MapKit 7
2.5.4 Firebase 7
2.6 Pengujian Perangkat Lunak 8
2.6.1 *BlackBox Testing* 8
2.6.2 *Usability Testing* 8
2.6.3 Skala *Likert* 8
2.7 *Standardized User Experience Percentile Rank Questionnaire for Mobile Apps* (SUPR-Qm) 9
BAB 3 METODOLOGI 10
3.1 Studi Literatur 10
3.2 Analisis Kebutuhan 11
3.3 Perancangan dan Implementasi 11
3.3.1 Perancangan 11
3.3.2 Implementasi 12
3.4 Pengujian 12
3.4.1 Evaluasi dan Demonstrasi Produk 12
3.4.2 *Retrospective* dan Perencanaan *Sprint* Selanjutnya 12
3.5 Pengujian 12
3.6 Kesimpulan 13
BAB 4 analisis kebutuhan 14
4.1 Gambaran Umum Sistem 14
4.1.1 Deskripsi Umum Sistem 14

4.1.2 Lingkungan Sistem 14
4.2 Analisis Kebutuhan Perangkat Lunak 15
4.2.1 Identifikasi Aktor 15
4.2.2 *User Story* 15
4.2.3 Kebutuhan Fungsional Sistem 16
4.2.4 Kebutuhan Non-Fungsional Sistem 17
4.2.5 Usecase Diagram 18
4.2.6 Usecase Scenario 19
BAB 5 Perancangan dan implementasi 25
5.1 Arsitektur Sistem 25
5.2 Sequence Diagram 26
5.2.1 *Sequence Diagram* - Login 26
5.2.2 *Sequence Diagram* - *Register* 27
5.2.3 *Sequence Diagram* - Melihat Lokasi Konselor Dalam Peta 28
5.2.4 *Sequence Diagram* - Melihat Daftar Konselor 28
5.2.5 *Sequence Diagram* - Melihat Profile Konselor 29
5.2.6 *Sequence Diagram* - Mengirim Permintaan Konseling 30
5.2.7 *Sequence Diagram* - Melakukan Konseling 31
5.2.8 *Sequence Diagram* - Memberi Rating ke Konselor 32
5.2.9 *Sequence Diagram* - Melihat Riwayat Konseling 33
5.2.10 *Sequence Diagram* - Membuat *Appointment* 33
5.2.11 *Sequence Diagram* - Melihat Daftar Appointment Dengan Konselor 34
5.2.12 *Sequence Diagram* - Melihat Daftar Permintaan Konseling 35
5.2.13 *Sequence Diagram* - Melihat Detail Profil Pasien 35
5.2.14 *Sequence Diagram* - Merespon Permintaan Konseling 36
5.2.15 Melihat Daftar Appointment Dari Pasien 37
5.3 Class Diagram 38
5.4 Perancangan Tabel Data 42
5.5 Perancangan Algoritme 43
5.6 Perancangan Antarmuka 44
5.6.1 Screenflow - Aplikasi Sisi Pasien 44
5.6.2 Screenflow - Aplikasi Sisi Konselor 45
5.7 Implementasi Perangkat Lunak 46
5.7.1 Spesifikasi Lingkungan Sistem 46
5.7.2 Batasan Implementasi 47
5.7.3 Implementasi Tabel Data 47
5.7.4 Implementasi Kode Program 48
5.7.5 Implementasi Antarmuka 50
BAB 6 pengujian 53
6.1 Pengujian Validasi 53
6.1.1 Login 53
6.1.2 Register 53

6.1.3 Melakukan Konseling 54
6.1.4 Membuat Appointment 54
DAFTAR REFERENSI 54

DAFTAR TABEL

DAFTAR GAMBAR

Gambar 2.1 *Lifecycle* Aplikasi iOS..... 5
Gambar 2.2 Alur Pengembangan Perangkat Lunak Menggunakan Scrum.....**Error! Bookmark not defined.**
Gambar 2.3 Arsitektur MVVM 7
Gambar 2.4 Daftar Pertanyaan SUPR-Qm 9
Gambar 3.1 Alur Penelitian Menggunakan *Scrum* 10

DAFTAR LAMPIRAN

BAB 1 PENDAHULUAN

1.1 Latar Belakang

Kesehatan mental saat ini menjadi permasalahan yang tidak bisa diremehkan. Bahkan kesehatan mental sekarang menjadi masalah yang dihadapi banyak negara di dunia, tak terkecuali Indonesia. Data dari WHO mengatakan bahwa 804.000 jumlah kematian pada tahun 2012 disebabkan oleh kasus bunuh diri, dimana rasio bunuh diri yang terjadi di Indonesia mencapai 4,3 per 100.000 penduduk (WHO, 2012). Penderita kesehatan mental paling banyak dialami oleh kalangan remaja atau mahasiswa dan memiliki dampak negatif yang serius terhadap prestasi belajar, keluarga, dan lingkungan sekitar (Chen & Jiang, 2019).

Depresi menjadi penyebab utama dalam permasalahan kesehatan mental. Depresi memengaruhi 4,4% populasi dunia dan 5% populasi di Indonesia. Selain jumlah penderita depresi yang tinggi, masalah lain pada Indonesia adalah kurangnya jumlah ketersediaan tenaga profesional kesehatan mental seperti konselor dan psikiater. WHO pun menginisiasi sebuah program bernama *Mental Health Gap Action Program* (mhGAP) yang bertujuan untuk menyediakan *internet-based intervention* untuk kesehatan mental yang dapat didistribusikan secara luas. (Arjadi, Nauta, & Bockting, 2018).

Berbagai upaya telah dilakukan oleh pemerintah Indonesia dalam menangani permasalahan kesehatan mental, salah satunya adalah menerbitkan UU no 18, tentang kesehatan mental dan perawatan orang dengan penyakit mental yang dicakup oleh cakupan kesehatan secara *universal*. Namun, banyak orang masih mengalami kesulitan dalam mengakses layanan kesehatan mental (Tristiana, Yusuf, Fitryasari, Wahyuni, & Nihayati, 2018). Di sisi lain, terdapat penelitian yang menunjukkan bahwa orang yang merasa depresi, sedih atau kesepian cenderung melampiaskan permasalahan tersebut dengan menggunakan sosial media, salah satunya adalah *Twitter* untuk sekedar bercerita atau mendapat dukungan interaksi (Mahoney et al., 2019). Hal ini menunjukkan bahwa *internet* dan teknologi informasi mampu menghasilkan sebuah solusi yang menjanjikan untuk melakukan perawatan masalah kesehatan mental yang ada di Indonesia (Arjadi et al., 2016). Mayoritas orang Indonesia menunjukkan keterbukaan untuk menggunakan *internet-based intervention* dalam menangani pemasalahan kesehatan mental, dimana sangat penting untuk dilakukan sosialisasi pemanfaatan *internet* kepada masyarakat secara luas. (Arjadi et al., 2018).

Dengan perkembangan teknologi yang semakin canggih dan digital, mampu menciptakan peluang untuk menyelesaikan permasalahan tersebut menggunakan sistem berbasis *mobile*. Dari data yang diperoleh, jumlah pengguna *smartphone* di Indonesia mencapai 41 juta pengguna dengan spesifikasi platform *Android*, sedangkan platform *iOS* mencapai 2.8 juta (Rahman, 2015). Dengan memanfaatkan fitur lokasi pengguna yang ada pada *smartphone*, maka memberi kemudahan pada pengguna untuk mengetahui lokasi pengguna lain dan dapat saling bertukar informasi. Maka dari itu, pemanfaatan sistem berbasis *mobile* dapat dilakukan untuk menghubungkan

orang yang memiliki permasalahan dengan konselor terdekat untuk mendapatkan pertolongan pertama dalam masalahnya.

1.2 Rumusan Masalah

Dari latar belakang yang sudah dijelaskan diatas, maka rumusan masalah pada penelitian ini adalah sebagai berikut :

1. Apa saja kebutuhan fungsionalitas dan non-fungsional dari aplikasi yang dikembangkan ?
2. Bagaimana hasil perancangan dari aplikasi yang dikembangkan ?
3. Bagaimana hasil implementasi dari aplikasi yang dikembangkan ?
4. Bagaimana hasil uji validasi dan usabilitas dari aplikasi yang dikembangkan?

1.3 Tujuan

Tujuan penelitian ini dilakukan adalah sebagai berikut :

1. Mengetahui apa saja kebutuhan fungsional dan non-fungsional pada aplikasi yang dikembangkan?
2. Mengetahui bagaimana hasil perancangan dari aplikasi yang dikembangkan ?
3. Mengetahui bagaimana hasil implementasi dari aplikasi yang dikembangkan ?
4. Mengetahui bagaimana hasil uji validasi dan usabilitas dari aplikasi yang dikembangkan?

1.4 Manfaat

1. Manfaat untuk penulis.

Mampu mengimplementasikan materi yang dipelajari pada perkuliahan untuk membangun aplikasi berbasis *mobile.*

2. Manfaat untuk peneliti selanjutnya.

Dapat menjadikan penelitian ini sebagai referensi atau acuan untuk membangun aplikasi berbasis *mobile.*

3. Manfaat untuk pengguna aplikasi

Dapat memberikan pertolongan pertama kepada penderita kesehatan mental dan memberikan pencegahan agar penderita segera membaik.

1.5 Batasan Masalah

Penelitian ini memiliki sejumlah batasan masalah sebagai berikut :

1. Aplikasi yang dibangun hanya menggunakan bahasa Indonesia.
2. Aplikasi harus terhubung ke internet agar bisa bekerja.
3. Aplikasi dibangun pada platform iOS dengan spesifikasi OS minimal iOS 11.

1.6 Sistematika Pembahasan

Struktur yang disusun pada proposal penelitian ini adalah sebagai berikut:

1. BAB I - Pendahuluan

   Pada bab ini berisikan latar belakang masalah yang diangkat, perumusan masalah, tujuan dari penelitian yang dilakukan, batasan-batasan yang ada pada penelitian, dan sistematika pembahasan penelitian.
2. BAB II - Landasan Kepustakaan

Pada bab ini berisikan kajian-kajian kepustakaan yang digunakan sebagai referensi untuk penelitian.

3. BAB III - Metodologi
   Pada bab ini berisikan penjelasan alur kerja penelitian yang dilakukan serta metodologi yang digunakan untuk menyelesaikan permasalahan.
4. BAB IV - Analisis Kebutuhan
   Pada bab ini menjelaskan tentang proses menggali semua kebutuhan dari sistem yang akan dibangun.
5. BAB V - Perancangan dan Implementasi
   Pada bab ini menjelaskan tentang proses perancangan untuk memulai pengembangan hingga hasil implementasi dari sistem yang dibangun.
6. BAB VI - Pengujian
   Pada bab ini menjelaskan tentang proses pengujian dari sistem yang telah selesai diimplementasikan dan memperoleh hasil kesimpulan dari sistem yang telah dibuat.
7. BAB VII - Penutup
   Pada bab ini memuat kesimpulan dari penelitian yang dilakukan dan berisi saran untuk peluang pengembangan lanjut kedepannya.

BAB 2 LANDASAN KEPUSTAKAAN

2.1 Kajian Pustaka

Dalam penelitian ini, penulis menemukan penelitian-penelitian serupa dan relevan yang sudah pernah dilakukan sehingga menjadi acuan dan referensi untuk penulis dalam melaksanakan penelitian.

Aplikasi pencarian rute perguruan tinggi berbasis android menggunakan *Location-Based Service.* Pada penelitian ini, penulis menawarkan solusi untuk memudahkan pengguna dalam mendapatkan informasi dan menentukan rute karena data dan informasi perguruan tinggi telah didaftarkan oleh peneliti (Nugroho, Nurhayati, & Widianto, 2017).

Aplikasi *mobile* berbasis lokasi untuk mencari layanan penyedia kesehatan di Yogyakarta. Pada penelitian ini, penulis membuat sebuah sistem yang mampu mencari lokasi layanan kesehatan dengan menggunakan *Google Map API* untuk menentukan rute dan lokasi terdekat (Erna Kumalasari Nurnawati, 2014).

Pada negara Indonesia, *internet-based intervention* memiliki prospek yang menjanjikan dalam mengurangi kesenjangan kesehatan mental mengingat penggunaan internet dan smartphone terus meningkat tiap harinya. Untuk mengetahui apakah *internet-based intervention* diterima dan digunakan banyak orang, (Arjadi et al., 2018) melakukan investigasi untuk mengetahui faktor apa saja yang memengaruhi penggunaan internet-based intervention untuk kesehatan mental di Indonesia. Hasil dari penelitian ini adalah mayoritas masyarakat Indonesia memiliki keterbukaan dalam penggunakan internet-based intervention dalam menangani permasalahan kesehatan mental di Indonesia. Untuk meningkatkan adopsi intervensi berbasis internet, penting untuk terlebih dahulu mempromosikan penggunaan internet kepada lebih banyak orang di seluruh negeri, terutama bagi mereka yang saat ini sedang mengalami gangguan kesehatan mental.

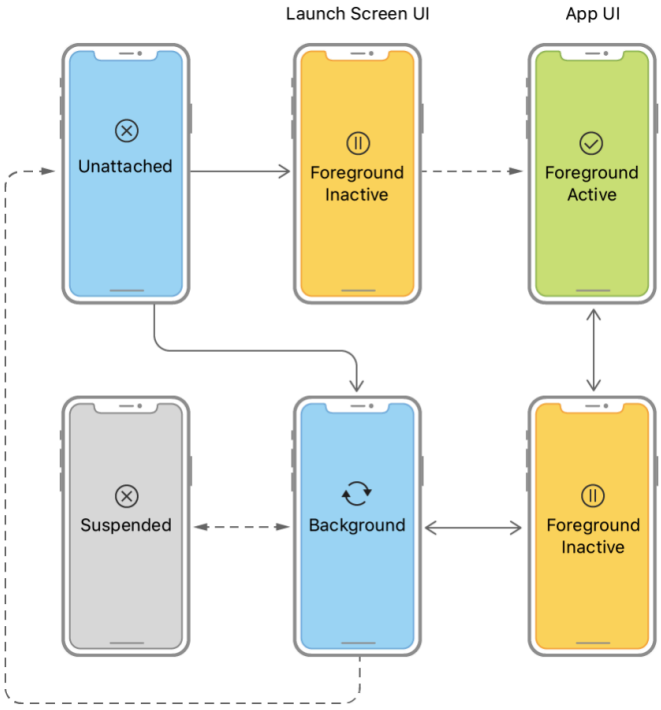
2.2 Mobile Application

Mobile application adalah perangkat lunak atau sekumpulan program yang berjalan pada *mobile device* dan menjalankan tugas tertentu untuk penggunanya. *Mobile application* memiliki keunggulan diantaranya adalah kemudahan penggunaan, *user-friendly*, murah, mudah diunduh, dan berjalan hampir pada semua level perangkat bergerak. Selain itu, *mobile application* dapat digunakan untuk kebutuhan yang luas seperti menelpon, mengirim pesan, suara, video, game, dan lain-lain (Islam & Mazumder, 2010). Namun disamping kelebihan yang dimiliki, *mobile application* memiliki beberapa kelemahan diantaranya memiliki tampilan yang relatif kecil, memori yang terbatas, kemampuan komputasi pada CPU yang terbatas, rentang transfer data yang terbatas, dan lain-lain. (Oinas-Kukkonen & Kurkela, 2003)

2.2.1 iOS

iOS merupakan sistem operasi para perangkat *mobile* yang dibuat oleh Apple. Pada penelitian ini, sistem yang akan dibangun menggunakan bahasa *Swift* dan

dibangun pada sistem operasi *iOS*. Diagram *lifeclyle* aplikasi dengan sistem operasi *iOS* digambarkan pada Gambar 2.1



Gambar 2.1 *Lifecycle* Aplikasi iOS
(Apple, 2018)

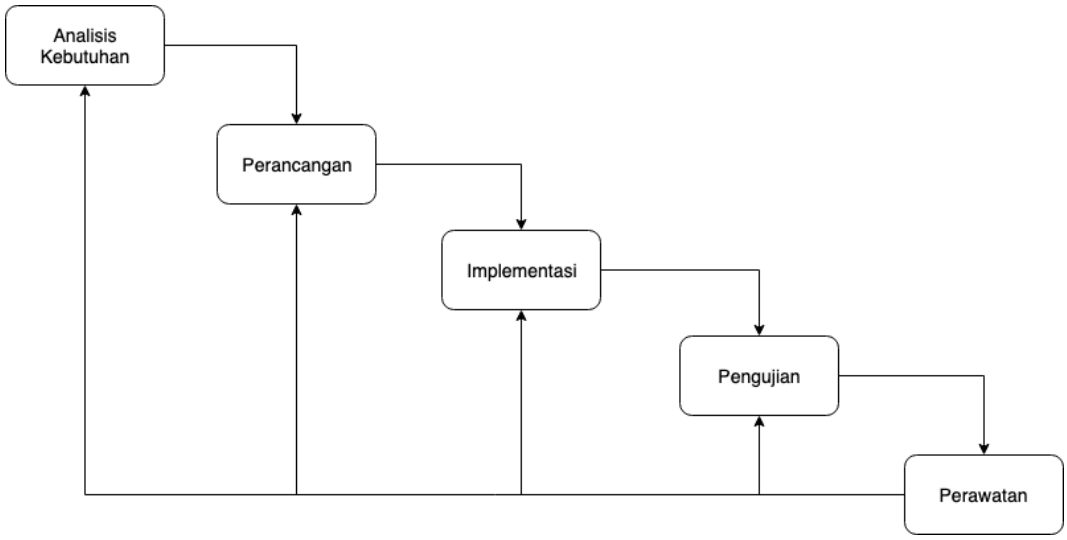
Adapun kelebihan pengembangan aplikasi pada sistem operasi iOS menurut (Avram, 2013) adalah sebagai berikut :

1. Apple sudah menyediakan standar dalam pengembangan *user-interface* sehingga pengembang perangkat lunak dapat menghemat waktu dalam melakukan pengembangan aplikasi.
2. Sistem operasi iOS hanya berjalan pada *iPhone* dan *iPad* sehingga pengembang dapat membuat aplikasi yang *responsive* dengan lebih mudah.
3. Keamanan data dan privasi pada aplikasi berbasis iOS dapat terjaga dengan baik.
4. Biaya pengembangan aplikasi berbasis iOS relatif lebih murah dan mudah.

2.3 Iterative Waterfall Model

Waterfall merupakan siklus pengembangan perangkat lunak yang bersifat linear. SDLC (*Software Development Lifecycle*) *Waterfall* dibagi menjadi dua diantaranya *Classic Waterfall* dan *Iterative Waterfall.* Kelemahan dari *Classic Waterfall* adalah sulit untuk mengatur perubahan dan perawatan pada sistem. *Iterative Waterfall* merupakan pengembangan dari *Classic Waterfall* untuk memudahkan tim dan *developer* agar mampu menghasilkan sistem dengan waktu yang efisien dan memudahkan dalam merawat sistem (Rather &

Bhatnagar, 2016). Fase pengembangan pada *Iterative Waterfall* digambarkan pada Gambar 2.2.



Gambar 2.2 SDLC Waterfall

Adapun tahap-tahap siklus pengembangan perangkat lunak menggunakan *Iterative Waterfall* adalah sebagai berikut:

1. **Analisis Kebutuhan**
   Tahap analisis kebutuhan merupakan tahap mengumpulkan kebutuhan akan sistem yang akan dibangun dengan melakukan wawancara dengan calon pengguna sistem.
2. **Perancangan**
   Tahap perancangan merupakan tahap merancang segala sesuatu yang dibutuhkan dalam membangun sistem seperti algoritma, antarmuka, dan diagram-diagram untuk membantu memudahkan tahap implementasi.
3. **Implementasi**
   Tahap implementasi merupakan tahap melakukan implementasi dari hasil perancangan yang telah selesai dibuat.
4. **Pengujian**
   Tahap pengujian merupakan tahap melakukan pengujian dari sistem yang telah selesai untuk mengidentifikasi adanya *bug* sebelum sistem digunakan oleh pengguna.
5. **Perawatan**
   Tahap perawatan adalah tahap ketika sistem telah rilis dan digunakan oleh pengguna, namun tim pengembang tetap melakukan perawatan apabila terdapat *bug* yang lolos dari tahap pengujian.

2.4 Location-Based Service

Location-based services atau layanan berbasis lokasi adalah layanan yang memperhitungkan lokasi geografis dari suatu objek (Junglas & Watson, 2008). Dengan adanya fitur *Global Positioning System* (GPS) pada perangkat *mobile* dapat membantu pengguna dalam melihat informasi lokasi suatu objek. Dalam konteks penelitian ini, peneliti menggunakan layanan berbasis lokasi dalam

aplikasi untuk menentukan dan mencari konselor psikologi terdekat dengan pengguna.

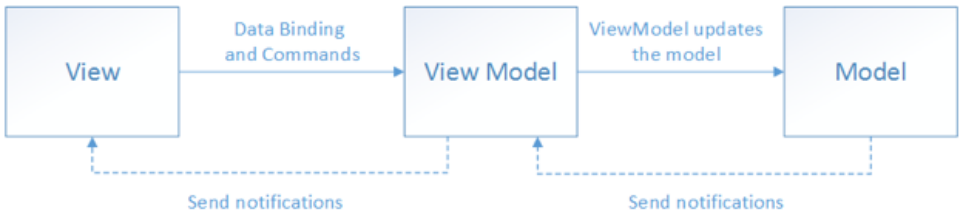
2.5 Best Practices

2.5.1 Swift

Swit adalah bahasa pemrograman baru yang dibuat oleh Apple pada tahun 2010 dan diumumkan pada tahun 2015 yagn tujuan utamanya digunakan untuk mengembangkan program diatas platform Apple (*iOS*, *macOS*, *tvOS* dan *watchOS*). *Swift* menawarkan kelebihan baru seperti *Type Safety*, *Fast* dan *Expressive*. *Type safety* yang adalah cara pemrograman baru agar program berjalan dengan cara yang aman (menghindari *null pointer*). *Swift* digunakan untum menggantikan bahasa pemrograman yang dipakai Apple sebelumnya, yaitu *Objective-C*, yang secara performa lebih cepat dari bahasa pendahulunya. (Apple, n.d.-b)

2.5.2 MVVM (Model-View-ViewModel)

MVVM merupakan arsitektur perangkat lunak yang memiliki tiga *layer* yaitu *model*, *view*, dan *view model.* Dengan adanya tiga *layer* dalam sistem maka terdapat pemisahan antara *business-logic, presentation-logic,* dan *UI-logic.* Arsitektur ini awalnya dikembangkan oleh Microsoft untuk mengembangkan aplikasi pada platform mereka, kemudian banyak diadopsi oleh pengembang pada platform lainnya. Adapun alur interaksi antar *layer* digambarkan pada Gambar 2.3.



Gambar 2.3 Arsitektur MVVM
(Microsoft, 2017)

2.5.3 MapKit

MapKit merupakan *framework* yang dibuat oleh Apple untuk menampilkan peta. Pada versi iOS 5.1 atau yang terbaru, *MapKit* menggunakan *Google Mobile Maps* (GMM) untuk menyediakan data yang sudah terintegrasi (Apple, n.d.-a) . Pada penelitian ini, penulis menggunakan *MapKit* untuk menyediakan informasi lokasi konselor psikologi pada pengguna untuk memudahkan interaksi keduanya.

2.5.4 Firebase

Firebase merupakan penyedia layanan berbasis *cloud* yang dibuat oleh Google. Dengan adanya *Firebase*, pengembang perangkat lunak tidak perlu memikirkan untuk membuat *web-service* atau *API* untuk melakukan pengolahan data. Layanan *Firebase* yang digunakan pada penelitian ini adalah *Realtime Database* dan *Firebase Authentication*.

Implementasi *Firebase* pada pengembangan aplikasi perangkat bergerak memiliki keuntungan. Menurut (Mevada, 2018) keuntungan menggunakan *firebase* adalah sebagai berikut:

1. Penyimpanan data secara *real-time* dan sinkronisasi data
2. Adanya *Google Analytics* yang dapat membantu pengembang untuk melacak perilaku dari pengguna.
3. *Firebase* menyediakan fitur *crash reporting* pada pengguna sehingga pengembang dapat memperbaiki *bug* tersebut dengan cepat.
4. *Firebase Authentication* yang mempermudah pengembang dalam membuat fitur autentikasi pada aplikasi.

2.6 Pengujian Perangkat Lunak

2.6.1 *BlackBox Testing*

Pengujian validasi yang digunakan pada penelitian ini adalah menggunakan *BlackBox Testing. BlackBox Testing* adalah pengujian yang dilakukan untuk mengevaluasi fitur atau kebutuhan dari pengguna dan memastikan bahwa sistem berjalan sesuai semestinya (Larrea, 2017). *Test-Case* yang dirancang pada *blackbox testing* bergantung pada spesifikasi perangkat lunak yang telah ditetapkan.

2.6.2 *Usability Testing*

Usability Testing adalah pengujian yang dilakukan untuk menilai kualitas dari sistem oleh calon pengguna utama. Tujuan dilakukan *usability testing* adalah menemukan permasalahan yang dialami calon pengguna agar bisa diperbaiki untuk meningkatkan kenyamanan pengguna dalam mengoperasikan sistem nantinya. Tingkat *usability* dapat diketahui dengan melakukan pengukuran tentang sejauh mana tingkat kemudahan dan tingkat kepuasan dari pengguna (Farouqi, Aknuranda, & Herlambang, 2018).

Jumlah responden untuk *usability testing* dipilih 5 orang. Alasan pemilihan jumlah responden tersebut adalah 5 orang responden merupakan angka yang optimal dan efektif dimana mengacu pada keuntungan dan usaha yang dilakukan dalam pengujian (Nielson & Landauer, 1993). Alasan lain dipilih 5 orang sebagai jumlah responden adalah 95% permasalahan yang ada pada sistem akan muncul pada 5 orang pertama ketika dilakukan *usability testing* (Cao, 2015).

2.6.3 Skala *Likert*

Skala *likert* digunakan untuk mengukur pendapat seseorang terhadap sebuah pernyataan, dalam hal penelitian ini digunakan untuk mengukur tingkat kepuasan pengguna terhadap penggunaan sistem. Skala *likert* menggunakan interval yang memiliki rentang dari nilai 1 sampai 5, dimana nilai 1 merepresentasikan tidak setuju hingga nilai 5 merepresentasikan sangat setuju.

2.7 *Standardized User Experience Percentile Rank Questionnaire for Mobile Apps* (SUPR-Qm)

Table 1. Items Considered in Study 1

Abbreviation	Item	% N/A
Fun	It's fun using the app.	2%
Enjoy	I enjoy using the app.	2%
Happy	Using the app makes me happy.	2%
Exciting	It's exciting to use the app.	3%
Bored	I use the app when I am bored.	2%
LTR	How likely are you to recommend the app to a friend or colleague?	0%
MeetReq	The app's capabilities meet my requirements.	0%
MobileSite	The app offers features its mobile website doesn't.	22%
Crash	The app rarely crashes or causes problems on my phone.	0%
Bugs	The app runs without bugs or errors.	0%
Freq	I would like to use the app frequently.	0%
Misuse	The app does not misuse my information.	4%
Trust	I trust the app with my personal information.	2%
Appagain	I plan to use the app again soon.	1%
EasyNav	It is easy to navigate within the app.	1%
EasyUse	The app is easy to use.	0%
Attractive	I find the app to be attractive.	0%
Clean	The app has a clean and simple presentation.	1%
Discover	I like discovering new features on the app.	3%
Cantlive	I can't live without the app on my phone.	1%
Delightful	The app is delightful.	0%
LearnFriends	I talk about things I do or learn on the app with my friends.	3%
CxFriends	I am able to connect or communicate with friends directly from the app.	9%

Gambar 2.4 Daftar Pertanyaan SUPR-Qm

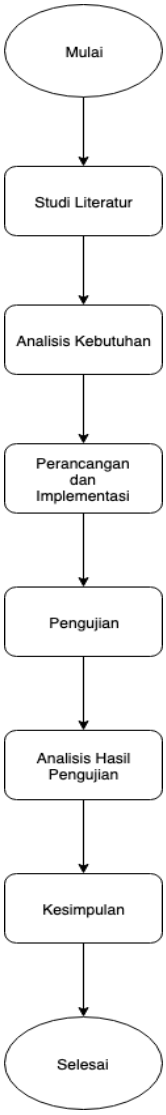
(Sauro & Zarolia, 2017)

SUPR-Qm adalah instrumen pengukuran yang dikembangkan untuk melakukan pengujian usabilitas dari aplikasi perangkat bergerak. Pengembangan SUPR-Qm berawal dari kebutuhan pengukuran usabilitas untuk aplikasi perangkat bergerak yang awalnya hanya tersedia untuk aplikasi berbasis *desktop* dan *website* dengan menggunakan SUPR-Q. SUPR-Q digunakan untuk menangkap konstruksi pengalaman pengguna yang luas dan komponen yang lebih spesifik dari semua sistem berbasis *website* maupun *desktop* (Sauro & Zarolia, 2017).

SUPR-Qm sendiri berbentuk daftar pertanyaan yang terdiri dari 16 buah dimana pertanyaan tersebut digunakan untuk mengukur tingkat usabilitas dari aplikasi perangkat bergerak dengan menggunakan skala *likert* sebagai komponen nilainya. Daftar pertanyaan dari SUPR-Qm digambarkan pada Gambar 2.4

BAB 3 METODOLOGI

Dalam sebuah penelitian diperlukan metodologi untuk menjadi landasan dan pedoman agar penelitian yang dilakukan terstruktur rapi. Metode yang digunakan pada penelitian yang dilakukan adalah *Waterfall*. Alur penelitian yang akan dilakukan digambarkan pada Gambar 3.1.



Gambar 3.1 Alur Penelitian Menggunakan *Waterfall*

3.1 Studi Literatur

Pada studi literatur, peneliti menemukan teori dan referensi penelitian yang relevan untuk memudahkan dalam melakukan pengembangan sistem. Adapun teori dan penelitian yang relevan sehingga menjadi referensi ketika penelitian ini dilakukan. Aplikasi pencarian rute perguruan tinggi berbasis android menggunakan *Location-Based Service.* Pada penelitian ini, penulis menawarkan solusi untuk memudahkan pengguna dalam mendapatkan informasi dan

menentukan rute karena data dan informasi perguruan tinggi telah didaftarkan oleh peneliti. (Nugroho et al., 2017).

Aplikasi *mobile* berbasis lokasi untuk mencari layanan penyedia kesehatan di Yogyakarta. Pada penelitian ini, penulis membuat sebuah sistem yang mampu mencari lokasi layanan kesehatan dengan menggunakan *Google Map API* untuk menentukan rute dan lokasi terdekat. (Erna Kumalasari Nurnawati, 2014)

Pada negara Indonesia, *internet-based intervention* memiliki prospek yang menjanjikan dalam mengurangi kesenjangan kesehatan mental mengingat penggunaan internet dan smartphone terus meningkat tiap harinya. Untuk mengetahui apakah *internet-based intervention* diterima dan digunakan banyak orang, (Arjadi et al., 2018) melakukan investigasi untuk mengetahui faktor apa saja yang memengaruhi penggunaan internet-based intervention untuk kesehatan mental di Indonesia. Hasil dari penelitian ini adalah mayoritas masyarakat Indonesia memiliki keterbukaan dalam penggunakan internet-based intervention dalam menangani permasalahan kesehatan mental di Indonesia. Untuk meningkatkan adopsi intervensi berbasis internet, penting untuk terlebih dahulu mempromosikan penggunaan internet kepada lebih banyak orang di seluruh negeri, terutama bagi mereka yang saat ini sedang mengalami gangguan kesehatan mental.

3.2 Analisis Kebutuhan

Pada tahap analisis kebutuhan, peneliti akan menjelaskan tiap kebutuhan yang digunakan untuk membangun aplikasi perangkat bergerak dan proses pengembangan perangkat lunak menggunakan *waterfall* akan dimulai. Analisis kebutuhan terdiri dari mendefinisikan kebutuhan fungsional dan non-fungsional.

3.3 Perancangan dan Implementasi

Pada tahap perancangan dan implementasi, dipastikan bahwa *product backlog* telah selesai dibuat sehingga implementasi dari tiap kebutuhan dapat segera dilakukan. Ketika tahap implementasi sudah selesai sesuai dengan *sprint* yang ditentukan, diharapkan semua p*roduct backlog* telah selesai dibuat sehingga bisa tahap pengembangan bisa dilanjutkan ke tahap pengujian dan *delivery* ke calon pengguna. Penjelasan masing-masing dari tahap perancangan dan implementasi adalah sebagai berikut :

3.3.1 Perancangan

Pada tahap perancangan, peneliti melakukan perancangan dasar terhadap sistem seperti membuat *class-diagram* untuk memodelkan arsitektur sistem dan *activity-diagram* yang digunakan untuk menggambarkan alur kerja sistem secara keseluruhan dari setiap *use-case* yang sudah ditetapkan. Kemudian dilanjutkan dengan membuat struktur data yang digunakan pada *Realtime Database* sebagai layanan penyimpanan data yang digunakan, dan tipe autentikasi yang digunakan sistem pada *Firebase Authentication*.

Setelah tahap perancangan dasar sudah selesai, maka dilanjutkan dengan melakukan perancangan algoritma dari tiap *use-case* yang sudah ditetapkan. Hal

ini dilakukan agar memudahkan peneliti dalam melanjutkan ke tahap implementasi nantinya.

3.3.2 Implementasi

Setelah tahap perancangan sudah selesai, maka dilakukan tahap implementasi ke pengembangan sistem. Hasil perancangan yang didapat menjadi pedoman untuk implementasi dilakukan mulai dari pembuatan *class* atau *interface* yang telah ditetapkan pada *class-diagram,* hingga integrasi sistem terhadap *Realtime Database* dan *Firebase Authentication*. Selanjutnya implementasi dari algoritma yang sudah ditetapkan dimana pada penelitian ini sistem dibangun pada perangkat bergerak dengan sistem operasi *iOS* menggunakan bahasa pemrograman *Swift*. Pada tahap akhir implementasi dilakukan perancangan *user-interface* menggunakan *framework* bawaan bernama UIKit yang telah disediakan oleh Apple.

3.4 Pengujian

Tahap berikutnya adalah melakukan pengujian dari implementasi yang sudah selesai dilakukan. Evaluasi dilakukan agar memastikan fitur-fitur yang telah diimplementasi tidak memiliki *bug* atau masalah lainnya. Tujuan lain dari tahap evaluasi adalah memastikan fitur aplikasi sudah semuanya selesai diimplementasikan.

3.4.1 Evaluasi dan Demonstrasi Produk

Evaluasi yang dilakukan pada penelitian ini adalah *Blackbox Testing*. *BlackBox Testing* dilakukan untuk memastikan semua fitur berjalan sesuai yang diharapkan. Apabila terdapat *bug* pada implementasi yang telah selesai dilakukan, maka kebutuhan implementasi tersebut kembali dikerjakan pada *sprint* yang masih berjalan.

3.4.2 *Retrospective* dan Perencanaan *Sprint* Selanjutnya

Ketika sprint telah selesai dilaksanakan, maka dilakukan tahap *retrospective* yaitu melakukan evaluasi pada *sprint* yang telah dilaksanakan untuk mengetahui hasil dan kesimpulan yang didapat. Ketika proses *retrospective* telah selesai, maka umpan balik dari *sprint* sebelumnya akan menjadi perbaikan pada *sprint* yang akan dilakukan selanjutnya.

3.5 Pengujian

Tahap pengujian pada penelitian ini adalah melakukan pengujian dari implementasi yang sudah selesai dilakukan. Pengujian dilakukan agar memastikan fitur-fitur yang telah diimplementasi tidak memiliki *bug* atau masalah lainnya. Pada penelitian ini juga dilakukan pengujian *usability* yang dilakukan untuk mendapatkan umpan balik terhadap *user-experience* dari aplikasi yang dibangun. Dalam penelitian ini, jumlah responden yang ditetapkan untuk *usability testing* adalah 5 orang. Pengujian ini dilakukan dengan menggunakan SUPR-Qm sebagai instrumen yang terdiri dari daftar pertanyaan dan skala *likert* sebagai skala yang digunakan untuk penilaiannya.

3.6 Kesimpulan

Tahap ini adalah mengambil kesimpulan setelah semua sprint yang ditentukan telah selesai dilaksanakan. Kesimpulan diambil berdasarkan hasil analisis dan hasil dari pengujian aplikasi yang telah dibuat. Kemudian hasil kesimpulan menjadi saran evaluasi kepada penelitian yang lebih lanjut.

BAB 4 ANALISIS KEBUTUHAN

Pada bab ini memuat proses analisis kebutuhan terhadap calon pengguna aplikasi yang berguna untuk mendapatkan kebutuhan sistem sehingga proses pengembangan dapat dimulai. Kebutuhan sistem yang dibuat mengacu pada *user story* dari hasil wawancara yang dilakukan terhadap calon pengguna dan konselor.

4.1 Gambaran Umum Sistem

Penelitian yang dilakukan adalah mengembangkan aplikasi perangkat bergerak untuk mencari konselor psikologi terdekat dengan pengguna dan dibangun pada *platform iOS*. Gambaran umum sistem terdiri dari deskripsi umum sistem dan lingkungan sistem.

4.1.1 Deskripsi Umum Sistem

Deskripsi umum pada sistem yang dibangun adalah aplikasi perangkat bergerak yang dikembangkan pada *platform iOS.* Aplikasi yang dikembangkan bertujuan untuk memudahkan pengguna dalam mencari konselor psikologi terdekat dengan pengguna sehingga pengguna dapat melakukan konseling secara *online* dan mempermudah pengguna untuk bertemu dengan konselor secara langsung. Setelah selesai melakukan konseling, pengguna dapat memberi *rating,* kritik dan saran kepada konselor.

Selain dari sisi pengguna, aplikasi dapat digunakan oleh konselor psikologi yang bertujuan untuk memudahkan konselor psikologi dalam melakukan konseling dengan pengguna. Konselor psikologi dapat melihat siapa saja pengguna yang ingin melakukan konseling. Kemudian konselor psikologi memiliki pilihan untuk menerima atau menolak permintaan dari pengguna untuk konselor. Konselor psikologi juga dapat melihat profil dari pengguna sebelum menerima permintaan konseling dari pengguna.

4.1.2 Lingkungan Sistem

Aplikasi ini dikembangkan menggunakan bahasa pemrograman *swift* dan *Xcode* sebagai IDE (*Integrated Development Environment*). Peneliti memanfaatkan *MapKit,* sebuah *framework* buatan *Apple* untuk dapat menampilkan peta pada aplikasi dan menggunakan *CLLocation* untuk mendapatkan informasi lokasi pengguna dan juga konselor psikologi. Untuk pengolahan data dan menangani *request* dari aplikasi, peneliti menggunakan *Firebase* dimana layanan yang digunakan adalah *firebase authentication* dan *realtime database.*

4.2 Analisis Kebutuhan Perangkat Lunak

4.2.1 Identifikasi Aktor

Definisi aktor adalah semua orang atau sistem eksternal yang memiliki peran secara langsung terhadap sistem. Aktor pada aplikasi ini terdiri dari *guest, user,* dan konselor. Penjelasan aktor dan tiap deskripsinya dapat dilihat pada Tabel 4.1.

Tabel 4.1 Daftar Aktor

Aktor	Deskripsi
Guest	Pada dasarnya, *guest* adalah pengguna namun belum memiliki akun atau belum melakukan autentikasi. Aktor *guest* dapat melihat aplikasi namun memerlukan proses autentikasi untuk mengakses fitur-fitur tertentu.
Pasien	Pasien adalah pengguna yang telah teridentifikasi melalui proses autentikasi pada aplikasi. Aktor pasien memiliki wewenang untuk mengakses semua fitur dari aplikasi yang dikembangkan.
Konselor	Aktor konselor adalah aktor yang telah teridentifikasi sebagai konselor setelah melakukan proses autentikasi.

4.2.2 *User Story*

Setelah melakukan wawancara secara pribadi dengan pengguna dan wawancara dengan konselor psikologi, hasil dari wawancara tersebut dapat dijadikan *user story* sebagai dasar untuk mendefinisikan semua fitur yang ada pada aplikasi. *User story* adalah daftar ekspetasi dari pengguna yang dapat membantu pengguna menyelesaikan masalah atau tujuannya melalui aplikasi yang dikembangkan. Dalam pembuatan *user story*, peneliti melakukan wawancara secara pribadi dengan calon pengguna terkait permasalahan kesehatan mental, serta ekspetasi ketika melakukan konseling dengan konselor psikologi dan wawancara dengan konselor psikologi untuk menggali kebutuhan aplikasi dari sisi konselor. Adapun *user story* pengguna yang telah didapatkan berdasarkan hasil wawancara dapat dilihat pada Tabel 4.2, dan *user story* konselor pada Tabel 4.3.

Tabel 4.2 User Story Pengguna

No	User Story Pasien
1.	Sebagai pasien saya dapat mencari konselor psikologi terdekat dengan lokasi saya.
2.	Sebagai pasien saya ingin melihat lokasi konselor psikologi yang terdekat dengan saya melalui peta.
3.	Sebagai pasien saya ingin dapat melihat profil konselor psikologi sebelum memulai konseling.
4.	Sebagai pasien saya ingin dapat memberi rating kepada konselor setelah melakukan konseling.
5.	Sebagai pasien saya ingin dapat melihat daftar riwayat konseling saya.
6.	Sebagai pasien saya ingin dapat membuat *appointment* dengan konselor.

Tabel 4.3 User Story Konselor

No	User Story Konselor
1.	Sebagai konselor saya ingin dapat melihat daftar permintaan konseling dari pengguna yang masuk dan dapat memilih untuk menerima permintaan konseling atau menolaknya.
2.	Sebagai konselor saya ingin konseling dilakukan melalui *chatting.*
3.	Sebagai konselor saya dapat melihat profil pengguna yang mengirim permintaan konseling.
4.	Sebagai konselor saya ingin melihat daftar *appointment* dari pasien ke saya.

4.2.3 Kebutuhan Fungsional Sistem

Kebutuhan fungsional sistem adalah daftar yang mendeskripsikan ekspektasi layanan yang seharusnya dilakukan oleh sistem (Faisandier, 2012). Pada penelitian ini, kebutuhan fungsional sistem dibuat berdasarkan *user story* yang telah didefinisikan. Dalam membuat daftar kebutuhan fungsional, tiap kebutuhan diberi kode sebagai identitas untuk memudahkan peneliti dalam mereferensikanya dalam seluruh proses penelitian. Daftar kebutuhan fungsional sistem untuk pengguna dapat dilihat pada Tabel 4.4 dan kebutuhan fungsional sistem untuk konselor dapat dilihat pada

Tabel 4.5.

Tabel 4.4 Kebutuhan Fungsional - Pasien

No	Kode Fungsi	Deskripsi Kebutuhan
1.	SS-F-001	Sistem dapat menyediakan tempat untuk pasien melakukan *login* dengan mengisi email dan kata sandi pada tempat yang disediakan.
2.	SS-F-002	Sistem dapat menyediakan tempat untuk pasien melakukan *register* dengan mengisi nama, tanggal lahir, jenis kelamin, email dan kata sandi pada tempat yang disediakan.
3.	SS-F-003	Sistem dapat menampilkan peta yang berisi lokasi pasien dan lokasi konselor psikologi terdekat dengan

		pasien.
4.	SS-F-004	Sistem dapat menampilkan daftar semua konselor psikologi terdekat dengan pasien.
5.	SS-F-005	Sistem dapat menampilkan detail profil konselor psikologi yang dipilih pasien.
6.	SS-F-006	Sistem dapat memfasilitasi pasien untuk mengirim permintaan konseling dengan konselor psikologi yang dipilih pasien.
7.	SS-F-007	Sistem dapat menampilkan halaman *chatting* antara pasien dan konselor dalam melakukan konseling secara *online.*
8.	SS-F-008	Sistem dapat memfasilitasi pasien untuk memberi rating kepuasan terhadap konselor setelah selesai melakukan konseling.
9.	SS-F-009	Sistem dapat menampilkan halaman riwayat konseling antara pasien dan konselor psikologi.
10.	SS-F-010	Sistem dapat memfasilitasi pasien untuk membuat *appointment* dengan konselor psikologi.

Tabel 4.5 Kebutuhan Fungsional - Konselor

1.	SS-KF-001	Sistem dapat menampilkan seluruh permintaan konseling dari pasien untuk konselor psikologi.
2.	SS-KF-002	Sistem dapat memfasilitasi konselor untuk menerima atau menolak permintaan konseling dari pasien.
3.	SS-KF-003	Sistem dapat menampilkan profil pasien yang mengirim permintaan konseling untuk konselor psikologi.
4.	SS-KF-004	Sistem dapat menampilkan daftar *appointment* dari pasien.

4.2.4 Kebutuhan Non-Fungsional Sistem

Kebutuhan non-fungsional adalah kebutuhan yang menetapkan atribut pada sistem perangkat lunak seperti batasan lingkungan dan implementasi pada sistem, peforma sistem, keandalan sistem, dan perawatan pada sistem (Bajpai & Gorthi, 2012). Kebutuhan non-fungsional yang dibuat pada aplikasi yang dikembangkan pada penelitian ini mencakup *usability* yang diukur dengan melakukan pengujian kepada pengguna untuk melakukan uji skenario terhadap aplikasi yang sudah ditetapkan.

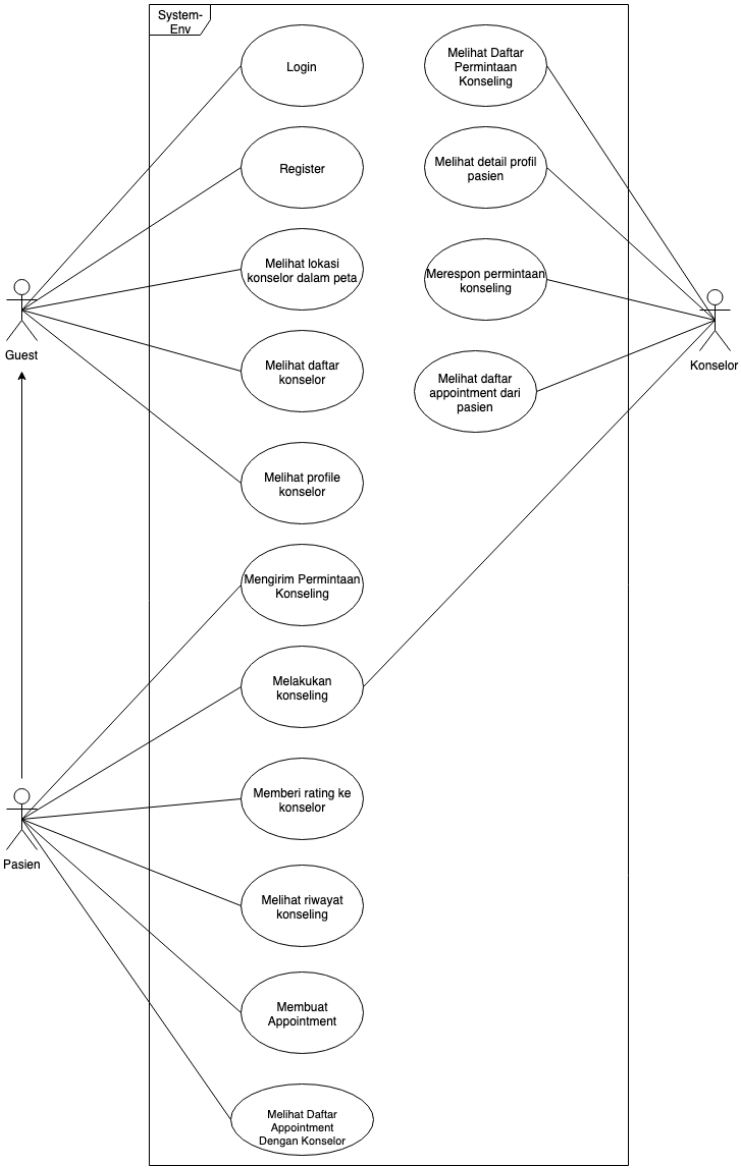
4.2.5 Usecase Diagram

Usecase diagram adalah diagram yang menggambarkan interaksi secara abstrak antara aktor dengan sistem (Shivaram & Handigund, 2015). Pada penelitian ini terdapat tiga aktor dengan masing-masing peran yang berbeda. *Usecase* diagram pada penelitian ini dapat dilihat pada Gambar 4.1. Pada *usecase* diagram yang telah dibuat, terdapat tiga aktor yang terlibat didalam sistem yaitu *guest,* pasien*,* dan konselor.

Aktor *guest* dapat melakukan *login* dengan mengisi *email* dan *password* pada tempat yang telah disediakan oleh sistem. Kemudian pada *usecase register,* aktor *guest* dapat melakukannya dengan mengisi nama lengkap, tanggal lahir, jenis kelamin, *email*, dan *passwordnya*. Pada *usecase* melihat lokasi konselor dalam peta, aktor *guest* dapat melihat semua lokasi konselor yang terdekat dengannya. Pada *usecase* melihat daftar konselor, aktor *guest* dapat melihat semua konselor dalam sistem yang berbentuk *list* dan diurutkan berdasarkan jarak antara aktor dengan konselor. Pada *usecase* melihat profil konselor, aktor *guest* dapat memilih konselor kemudian sistem menampilkan detail informasi mengenai konselor tersebut.

Aktor pasien merupakan spesialisasi dari aktor *guest*, dimana aktor pasien dapat melakukan semua aktivitas atau *usecase* yang dimiliki pada aktor *guest* ditambah dengan *usecase* khusus pada aktor pasien itu sendiri. Pada *usecase* mengirim permintaan konseling, aktor pasien dapat mengirim permintaan untuk melakukan konseling secara online dengan konselor yang dipilihnya. Pada *usecase* melakukan konseling secara online,setelah aktor pasien mengirim permintaan konseling kepada konselor dan konselor tersebut menerima permintaanya maka aktor pasien dapat melakukan konseling secara *online* dengan konselor tersebut. Pada *usecase* memberi rating ke konselor, aktor pasien dapat memberikan rating kepuasan terhadap konselor setelah selesai melaksanakan konseling. Pada *usecase* melihat daftar riwayat konseling, aktor pasien dapat melihat semua riwayat konseling yang pernah dilakukan sebelumnya dengan konselor.

Aktor konselor memiliki tiga *usecase* pada aplikasi yang dikembangkan. Pada *usecase* melihat daftar permintaan konseling, aktor konselor dapat melihat semua permintaan konseling yang masuk kedalam akunnya. Pada *usecase* melihat detail profil pasien, aktor konselor dapat melihat detail informasi profil dari pasien yang mengirim permintaan untuk melakukan konseling. Pada *usecase* merespon permintaan konseling, aktor konselor dapat menerima permintaan konseling maupun menolak permintaan konseling dari pasien.



Gambar 4.1 Usecase Diagram

4.2.6 Usecase Scenario

Usecase scenario mendefinisikan skenario aktivitas aktor dalam melaksanakan *usecase.* Berikut merupakan *usecase scenario* yang dikembangkan pada penelitian yang dilakukan.

Tabel 4.6 Usecase Scenario - Login

Usecase	Login
Actor	*Guest*
Target	*Guest* telah teridentifikasi sebagai *user* dalam sistem
Pre-Condition	-
Main Flow	1. *Guest* mengisi email dan password pada field yang disediakan dan menekan tombol login. 2. Sistem melakukan validasi email dan

	password dari *guest*, kemudian menampilkan halaman utama.
Alternative Flow	Apabila sistem gagal melakukan validasi email dan password pengguna, maka sistem akan menampilkan pesan error.
Post Condition	*Guest* sudah teridentifikasi sebagai pasien dan masuk kedalam halaman utama sistem.

Tabel 4.7 Usecase Scenario - Register

Usecase	Register
Actor	*Guest*
Target	*Guest* memiliki akun yang terdaftar
Pre-Condition	-
Main Flow	1. Sistem menampilkan halaman login. 2. *Guest* menekan tombol navigasi ke halaman *register.* 3. Sistem menampilkan halaman register. 4. *Guest* mengisi data diri yang terdiri dari nama lengkap, tanggal lahir, jenis kelamin, email, dan password. Kemudian *guest* menekan tombol register. 5. Sistem memvalidasi data diri dari *guest,* kemudian menampilkan ke halaman utama.
Alternative Flow	Apabila sistem gagal memvalidasi data diri, maka sistem akan menampilkan pesan error.
Post Condition	*Guest* memiliki akun yang terdaftar dan masuk kedalam halaman utama sistem.

Tabel 4.8 Usecase Scenario - Melihat lokasi konselor dalam peta

Usecase	Melihat lokasi konselor dalam peta
Actor	*Guest*
Target	*Guest* dapat melihat semua lokasi konselor dalam peta.
Pre-Condition	-
Main Flow	1. *Guest* membuka *tab* beranda pada sistem 2. Sistem menampilkan semua lokasi konselor dalam peta
Alternative Flow	-
Post Condition	Sistem telah menampilkan semua lokasi konselor dalam peta.

Tabel 4.9 Usecase Scenario - Melihat daftar konselor

Usecase	Melihat daftar konselor
Actor	*Guest*

Target	*Guest* melihat daftar konselor terdekat.
Pre-Condition	-
Main Flow	1. *Guest* menekan tombol tampilkan semua konselor 2. Sistem menampilkan daftar konselor terdekat dengan pengguna.
Alternative Flow	-
Post Condition	Sistem telah menampilkan daftar konselor terdekat.

Tabel 4.10 Usecase Scenario - Melihat profile konselor

Usecase	Melihat profil konselor
Actor	*Guest*
Target	*Guest* dapat melihat detail informasi profil konselor yang dipilih.
Pre-Condition	-
Main Flow	1. *Guest* memilih konselor. 2. Sistem menampilkan detail informasi profil konselor
Alternative Flow	-
Post Condition	Sistem telah menampilkan detail informasi profil dari konselor yang dipilih.

Tabel 4.11 Usecase Scenario - Mengirim permintaan konseling

Usecase	Mengirim permintaan konseling
Actor	Pasien
Target	Pasien dapat mengirim permintaan konseling kepada konselor yang dipilih..
Pre-Condition	-
Main Flow	1. Pasien menekan tombol kirim permintaan konseling pada konselor yang dipilih 2. Sistem menampilkan halaman menunggu konfirmasi konselor
Alternative Flow	-
Post Condition	Sistem telah mengirim permintaan konseling dari pasien kepada konselor yang telah dipilih.

Tabel 4.12 Usecase Scenario - Melakukan konseling

Usecase	Melakukan konseling
Actor	Pasien
Target	Pasien melakukan konseling secara online dengan *chatting* dengan konselor.
Pre-Condition	-
Main Flow	1. Sistem menampilkan halaman *chatting.*

	2. Pasien mengirim pesan. 3. Sistem menampilkan pesan dari pasien 4. Pasien menekan tombol selesai konseling.
Alternative Flow	-
Post Condition	Pasien telah melakukan konseling secara online dengan konselor.

Tabel 4.13 Usecase Scenario - Memberikan rating kepada konselor

Usecase	Memberi rating kepada konselor
Actor	Pasien
Target	Pasien memberi rating kepada konselor setelah selesai melakukan konseling.
Pre-Condition	-
Main Flow	1. Sistem menampilkan halaman rating. 2. Pasien memberi rating dan menekan tombol selesai
Alternative Flow	-
Post Condition	*Sistem* kembali menampilkan halaman utama sistem

Tabel 4.14 Usecase Scenario - Membuat Appointment

Usecase	Membuat *Appointment*
Actor	Pasien
Target	Pasien dapat membuat *appointment* dengan konselor psikologi
Pre-Condition	-
Main Flow	1. Sistem menampilkan halaman *appointment* 2. Pasien membuat *appointment* dari waktu yang tersedia pada hari ini 3. Sistem menampilkan pesan sukses membuat *appointment*
Alternative Flow	-
Post Condition	Sistem telah mengirim permintaan konseling dari pasien kepada konselor yang telah dipilih.

Tabel 4.15 Usecase Scenario - Melihat Riwayat Konseling

Usecase	Melihat riwayat konseling
Actor	Pasien
Target	Pasien dapat melihat semua riwayat konseling.
Pre-Condition	-
Main Flow	1. Pasien melihat riwayat konseling 2. Sistem menampilkan daftar riwayat konseling dari pasien.

Alternative Flow	-
Post Condition	Sistem telah menampilkan daftar riwayat konseling dari pasien.

Tabel 4.16 Usecase Scenario - Melihat Daftar Appointment Dengan Konselor

Usecase	Melihat Daftar *Appointment* dengan Konselor
Actor	Pasien
Target	Pasien dapat melihat daftar *appointment* dengan konselor.
Pre-Condition	-
Main Flow	1. Pasien melihat daftar *appointment* 2. Sistem menampilkan daftar *appointment* dengan konselor
Alternative Flow	-
Post Condition	Sistem telah menampilkan daftar *appointment* dengan konselor.

Tabel 4.17 Usecase Scenario - Melihat daftar permintaan konseling

Usecase	Melihat daftar permintaan konseling
Actor	Konselor
Target	Konselor dapat melihat semua permintaan konseling dari pasien.
Pre-Condition	-
Main Flow	1. Konselor membuka *tab* konseling 2. Sistem menampilkan daftar permintaan konseling dari pasien.
Alternative Flow	-
Post Condition	Sistem telah menampilkan semua permintaan konseling dari pasien.

Tabel 4.18 Usecase Scenario - Melihat detail profil user

Usecase	Melihat detail profil pasien
Actor	Konselor
Target	Konselor melihat detail informasi dari profil pasien yang mengirim permintaan konseling.
Pre-Condition	-
Main Flow	1. Konselor memilih permintaan konseling 2. Sistem menampilkan detail profil dari pasien.
Alternative Flow	-
Post Condition	Sistem telah menampilkan detail informasi profile dari pasien.

Tabel 4.19 Usecase Scenario - Melihat Daftar *Appointment* Dari Pasien

Usecase	Melihat daftar *appointment* dari pasien
Actor	Konselor
Target	Konselor dapat melihat daftar *appointment* dari pasien.
Pre-Condition	-
Main Flow	1. Konselor membuka halaman daftar *appointment* 2. Sistem menampilkan halaman *appointment*
Alternative Flow	-
Post Condition	Sistem telah menampilkan semua permintaan konseling dari pasien.

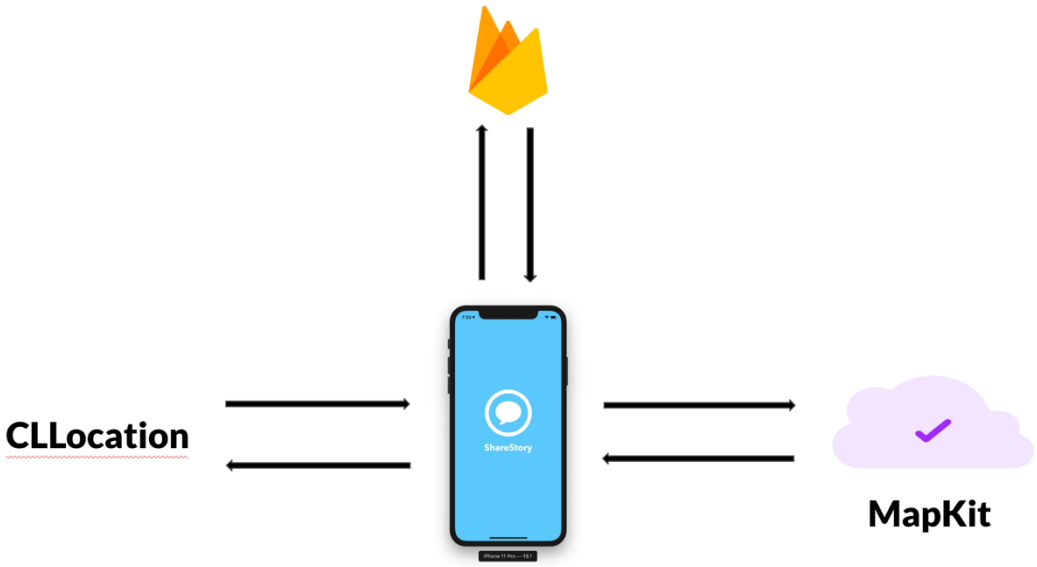
Tabel 4.20 Usecase Scenario - Merespon permintaan konseling

Usecase	Merespon permintaan konseling
Actor	Konselor
Target	Konselor memberi respon kepada permintaan konseling yang dikirim pasien.
Pre-Condition	-
Main Flow	1. Konselor menekan tombol untuk merespon permintaan konseling dari pasien. 2. Sistem menampilkan halaman *chatting.*
Alternative Flow	Apabila konselor menolak permintaan dari pasien, maka sistem menampilkan pesan *pop-up* berisi alasan penolakan dari konselor.
Post Condition	Sistem telah mengirim respon dari konselor kepada pasien.

BAB 5 PERANCANGAN DAN IMPLEMENTASI

Bab ini membahas tentang perancangan yang dilakukan untuk membuat gambaran dari sistem yang akan dibuat. Hasil dari perancangan akan diimplementasikan dan dibahas pada bab implementasi. Tahap perancangan terdiri dari perancangan arsitektur sistem, *sequence diagram*, *class diagram,* perancangan basis data, dan diakhiri dengan membuat rancangan antarmuka.

5.1 Arsitektur Sistem



Gambar 5.1 Arsitektur Sistem

Perancangan arsitektur sistem pada aplikasi yang dikembangkan digambarkan pada Gambar 5.1. Pada arsitektur sistem yang dibangun terdapat tiga komponen *library* utama yaitu *MapKit*, *CLLocation*, dan *Firebase*. *MapKit* adalah *framework* buatan Apple yang digunakan untuk menampilkan peta. Dalam hal ini, MapKit digunakan untuk menampilkan lokasi pasien dan konselor psikologi terdekat dengan pasien. *CLLocation* adalah *native-library* yang digunakan untuk mengatur dan memonitor lokasi terkini dari pasien dan mendapatkan alamat fisik dari konselor psikologi. *Firebase* adalah layanan berbasis *cloud* yang dibuat oleh Google. Pada penelitian ini, layanan Firebase yang digunakan adalah *Realtime Database* yang bertujuan untuk melakukan transaksi pertukaran data dengan sisi aplikasi serta memanfaatkan fitur *realtime* untuk membuat layanan *chatting* pada sisi aplikasi.

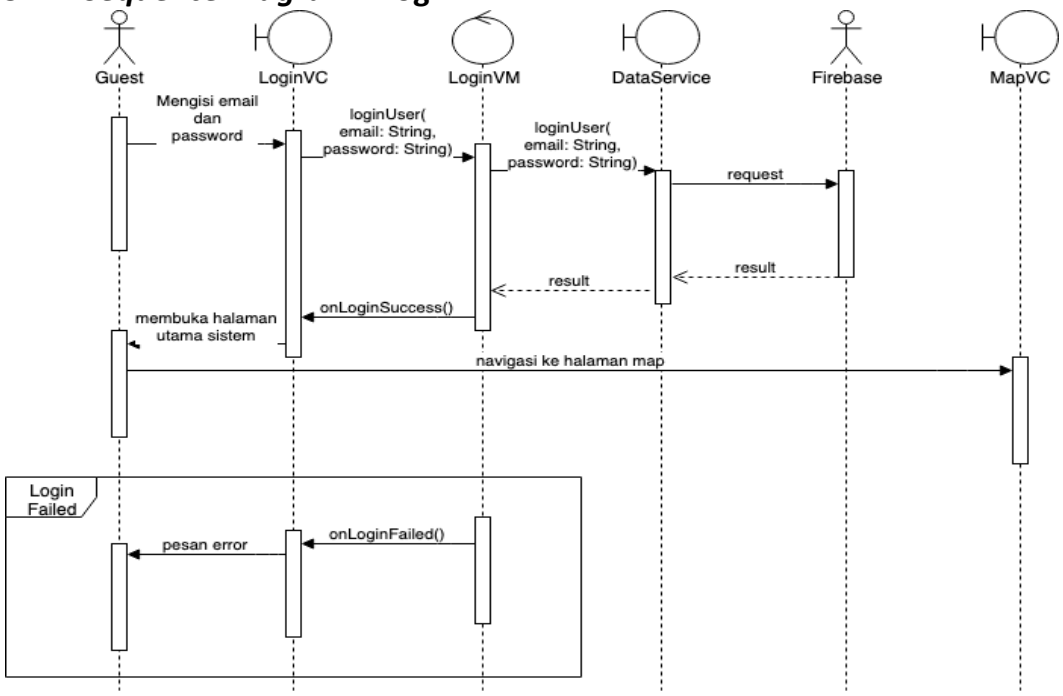
Selain menggunakan ketiga *library* tersebut, arsitektur yang digunakan untuk mengorganisir tiap modul dan fitur adalah MVVM (*Model-View-ViewModel*). *Model* memiliki tugas untuk membuat representasi data objek. *ViewModel* memiliki tugas untuk menyediakan data yang siap dikonsumsi untuk *view.* Dan *view* memiliki tugas untuk menampilkan data dan menerima respon atau *event* dari pengguna. Dalam penerapannya, terdapat penambahan satu layer yaitu

Service, dimana sebuah objek *singleton* yang memiliki tugas untuk melakukan *request* ke *Firebase* untuk melakukan pertukaran data.

5.2 Sequence Diagram

Sequence diagram merupakan diagram yang menggambarkan urutan aktifitas antara aktor dengan komponen pada sistem pada masing-masing kebutuhan fungsional. Berikut merupakan hasil perancangan *sequence diagram* dari kebutuhan fungsional yang telah didefinisikan.

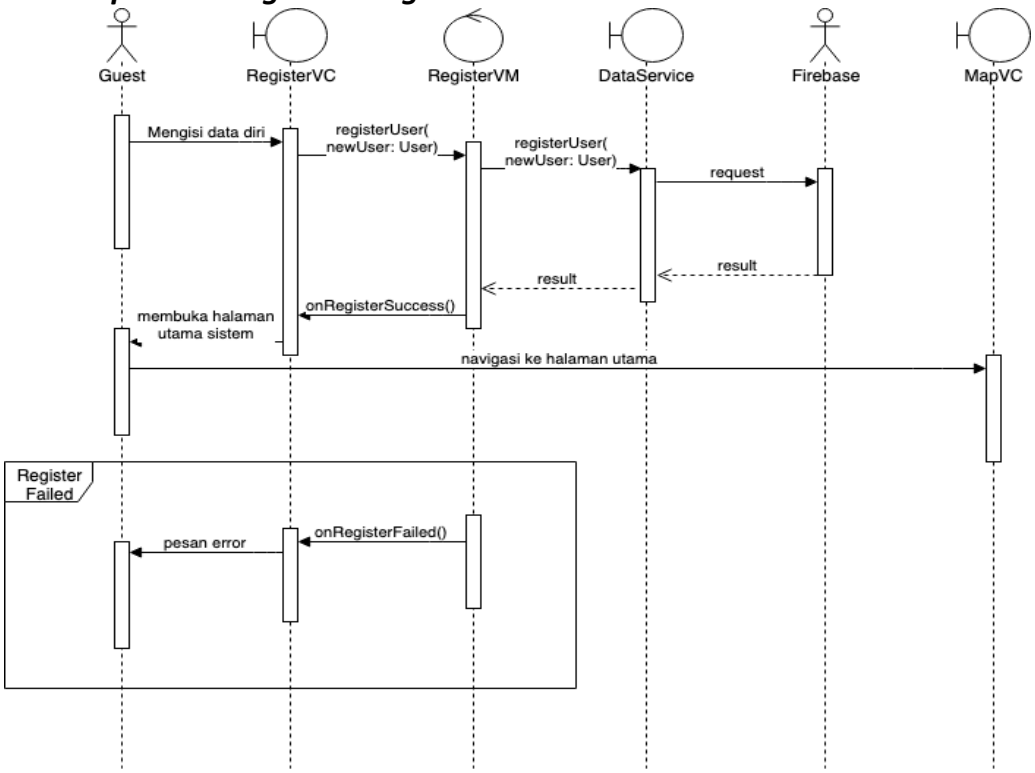
5.2.1 *Sequence Diagram* - Login



Gambar 5.2 Sequence Diagram - Login

Sequence diagram login dari aktor *Guest* digambarkan pada Gambar 5.2. Aktor *guest* mengisi email dan password pada tempat yang disediakan pada halaman *LoginVC.* Kemudian pada *class LoginVC* memanggil fungsi `loginUser(email: String, password: String)` yang ada pada *class LoginVM.* Dari *LoginVM* kemudian memanggil fungsi `loginUser(email: String, password: String)` yang ada pada *class DataService.* Pada *class DataService* melakukan *request* kepada Firebase dan mendapatkan *result* berupa hasil verifikasi login dari aktor *guest.* Apabila proses login berhasil maka *LoginVM* memanggil fungsi `onLoginSuccess()` pada *class LoginVC* dan mengarahkan aktor untuk berpindah ke halaman *MapVC.* Apabila proses login gagal maka sistem menampilkan pesan *error.*

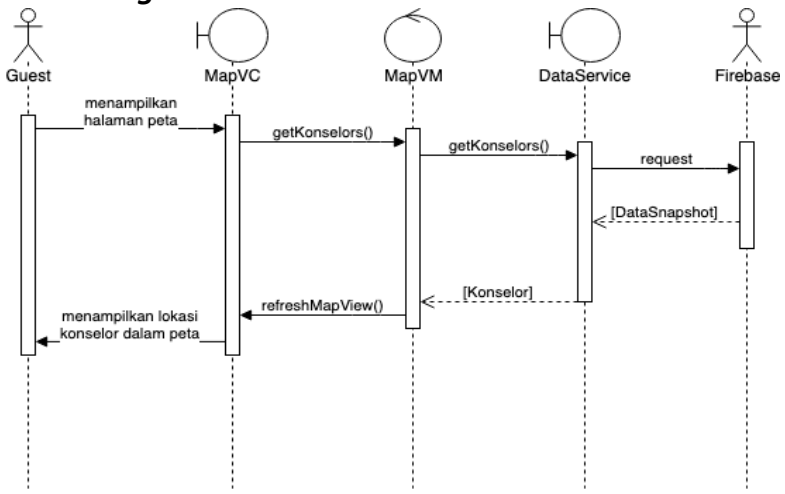
5.2.2 *Sequence Diagram - Register*



Gambar 5.3 Sequence Diagram - Register

Sequence diagram register dari aktor *Guest* digambarkan pada Gambar 5.3. Aktor *guest* mengisi data diri pada tempat yang disediakan pada halaman *RegisterVC.* Kemudian pada *class RegisterVC* memanggil fungsi `registerUser(newUser: User)` yang ada pada *class registerVM.* Dari *registerVM* kemudian memanggil fungsi `registerUser(newUser: User)` yang ada pada *class DataService.* Pada *class DataService* melakukan *request* kepada Firebase dan mendapatkan *result* berupa hasil verifikasi register dari aktor *guest.* Apabila proses login berhasil maka *RegisterVM* memanggil fungsi `onRegisterSuccess()` pada *class RegisterVC* dan mengarahkan aktor untuk berpindah ke halaman *MapVC.* Apabila proses register gagal maka sistem menampilkan pesan *error.*

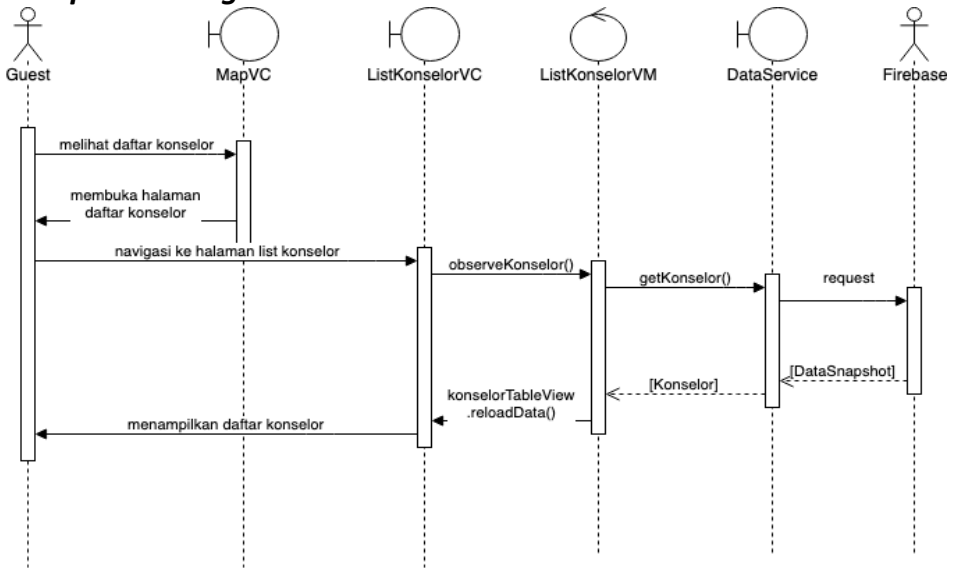
5.2.3 *Sequence Diagram* - Melihat Lokasi Konselor Dalam Peta



Gambar 5.4 Sequence Diagram - Melihat Lokasi Konselor Dalam Peta

Sequence diagram melihat lokasi konselor dalam peta oleh aktor *guest* digambarkan pada Gambar 5.4. Sistem menampilkan halaman *MapVC* pada aktor yang kemudian memanggil fungsi `getKonselors()` pada objek *MapVM*. Dari *MapVM* kemudian memanggil fungsi `getKonselors()` yang ada pada objek *DataService*. Kemudian objek *DataService* melakukan *request* kepada Firebase untuk mendapatkan data konselor. Ketika *DataService* telah selesai melakukan *request* dan mengembalikan data menuju objek *MapVM* maka dilakukan pemanggilan fungsi `refreshMapView()` yang ada pada halaman *MapVC.*

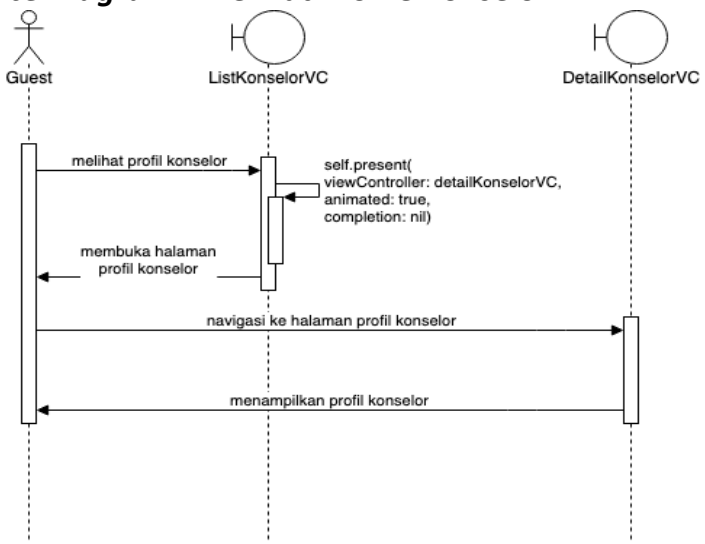
5.2.4 *Sequence Diagram* - Melihat Daftar Konselor



Gambar 5.5 Sequence Diagram - Melihat Daftar Konselor

Sequence diagram melihat daftar konselor dari aktor *guest* digambarkan pada Gambar 5.5. Pada halaman *MapVC*, aktor *guest* menekan tombol yang tersedia untuk melihat daftar konselor dan berpindah halaman ke *ListKonselorVC*. Kemudian memanggil fungsi `observeKonselor()` yang ada pada objek *ListKonselorVM*. Dari objek *ListKonselorVM* kemudian memanggil fungsi `getKonselor()` yang ada pada objek *DataService*. Setelah itu objek *DataService* melakukan request ke Firebase dan menghasilkan data konselor yang dikembalikan ke objek *ListKonselorVM*. Data yang dihasilkan membuat halaman *ListKonselorVC* melakukan `reloadData()` untuk menampilkan data konselor kepada aktor.

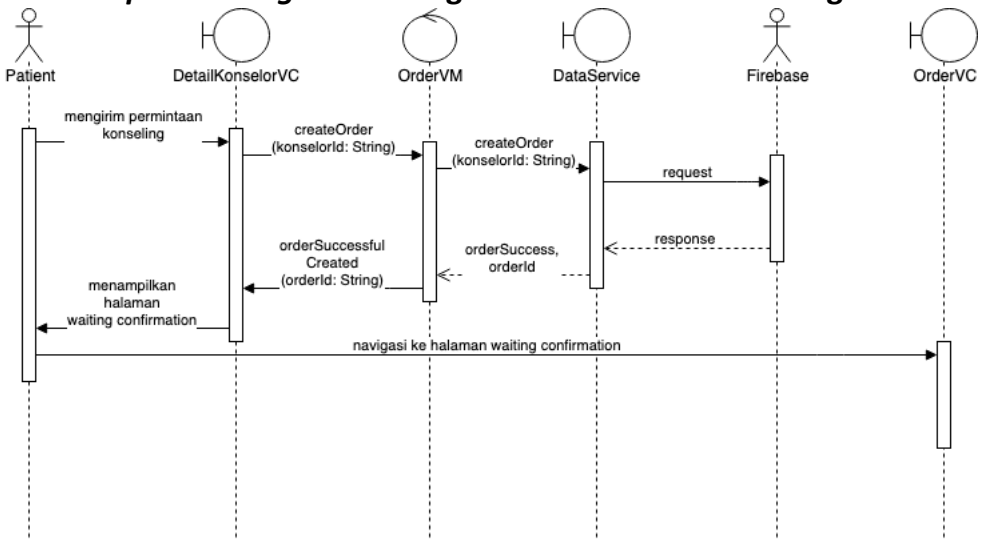
5.2.5 *Sequence Diagram* - Melihat Profile Konselor



Gambar 5.6 Sequence Diagram - Melihat Profile Konselor

Sequence diagram melihat profile konselor oleh aktor *guest* digambarkan pada Gambar 5.6. Pada halaman *ListKonselorVC*, aktor guest memilih konselor yang tersedia untuk dapat melihat detail profilnya. Kemudian pada sistem membuka halaman profil konselor yang bernama *DetailKonselorVC*.

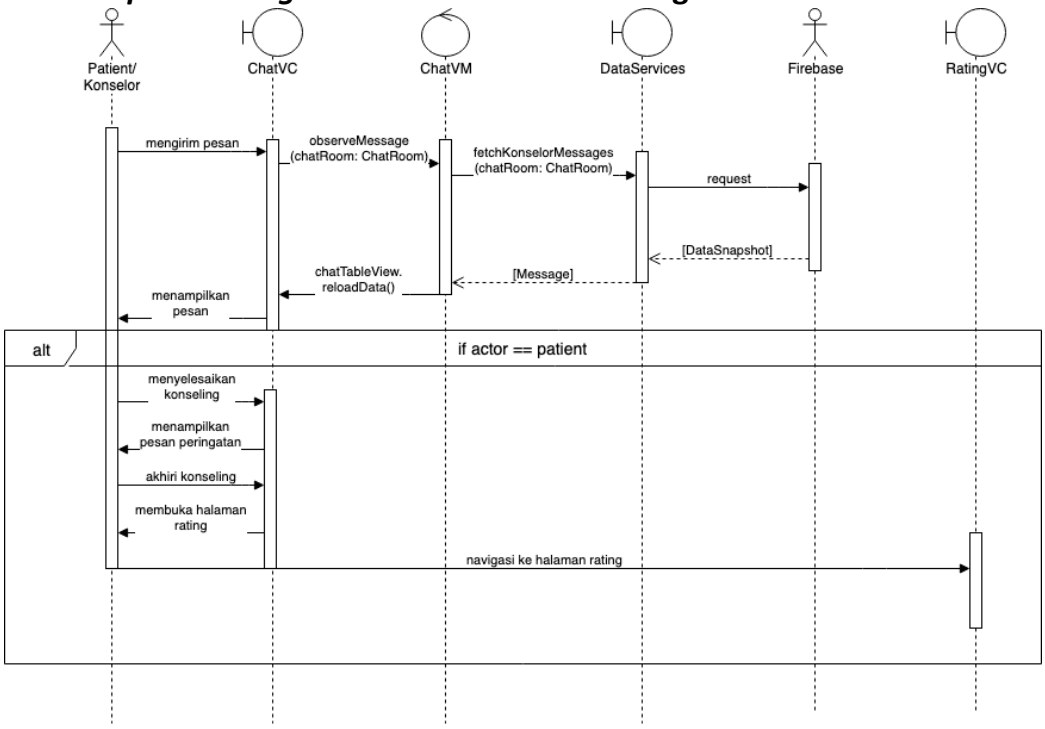
5.2.6 *Sequence Diagram* - Mengirim Permintaan Konseling



Gambar 5.7 Sequence Diagram - Mengirim Permintaan Konseling

Sequence diagram mengirim permintaan konseling oleh aktor pasien digambarkan pada Gambar 5.7. Pasien menekan tombol untuk mengirim permintaan konseling pada halaman *DetailKonselorVC*. Kemudian memanggil fungsi `createOrder(konselorId: String)` pada objek *OrderVM*. *OrderVM* kemudian memanggil fungsi `createOrder(konselorId: String)` pada objek *DataService*. Setelah pemanggilan fungsi tersebut maka objek *DataService* melakukan *request* ke Firebase dan mendapatkan respon berupa `orderSuccess` dan `orderId`. Kemudian objek *OrderVM* memanggil fungsi `orderSuccessfulCreated(orderId: String)` pada *DetailKonselorVC* dan menampilkan halaman menunggu konfirmasi dari konselor.

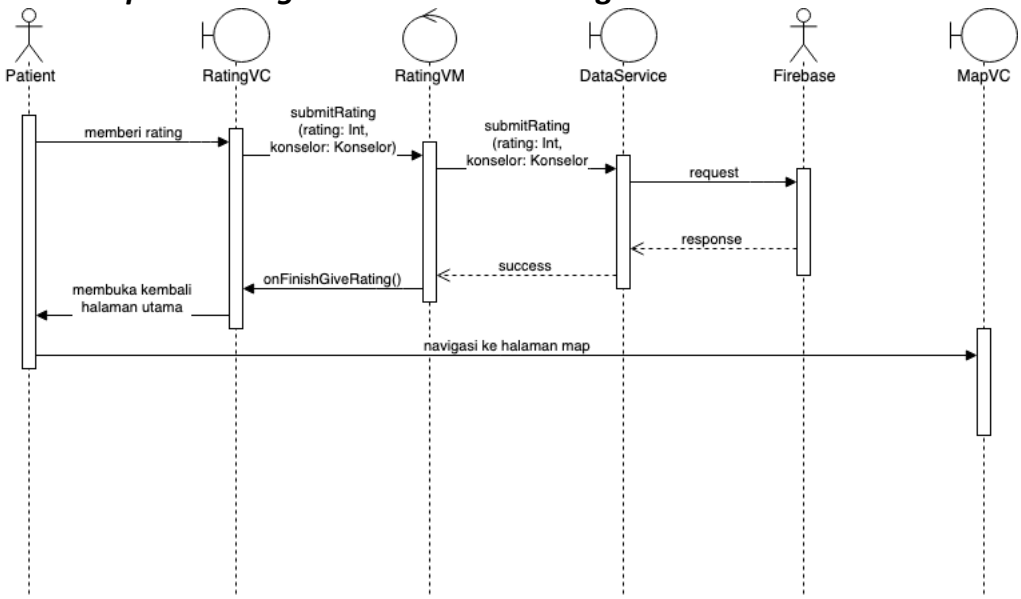
5.2.7 *Sequence Diagram* - Melakukan Konseling



Gambar 5.8 Sequence Diagram - Melakukan Konseling

Sequence diagram melakukan konseling oleh aktor pasien digambarkan pada Gambar 5.8. Aktor pasien mengirim pesan pada tempat yang disediakan pada halaman *ChatVC* yang kemudian memanggil fungsi `observeMessage(chatRoom: ChatRoom)` pada objek *ChatVM*. Kemudian objek *ChatVM* memanggil fungsi `fetchKonselorMessages(chatRoom: ChatRoom)` pada objek *DataServices*. Objek *DataService* kemudian melakukan *request* kepada Firebase dan mendapatkan respon berupa *array* bertipe data *Message*. Kemudian objek *ChatVM* melakukan *trigger* pada *chatTableView* untuk memanggil fungsi `reloadData()` yang berguna untuk menambahkan pesan kepada halaman *ChatVC*. Setelah menampilkan pesan maka aktor pasien dapat menyelesaikan konseling dengan menekan tombol selesai pada halaman *ChatVC*. Kemudian *ChatVC* menampilkan pesan pop-up untuk mengonfirmasi aksi dari aktor pasien. Kemudian aktor pasien menekan tombol akhiri konseling dan sistem membuka halaman rating yaitu *RatingVC*.

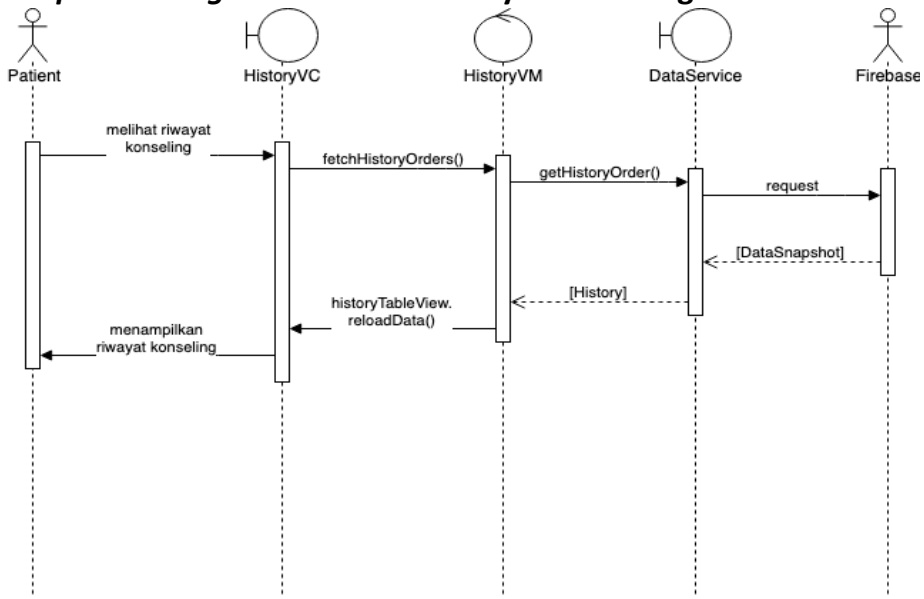
5.2.8 *Sequence Diagram* - Memberi Rating ke Konselor



Gambar 5.9 Sequence Diagram - Memberi Rating ke Konselor

Sequence diagram memberi rating ke konselor oleh aktor pasien digambarkan pada Gambar 5.9. Setelah selesai melakukan konseling dengan konselor, maka pasien dapat memberi rating pada tempat yang disediakan pada halaman *RatingVC*. Kemudian memanggil fungsi `submitRating(rating: Int, konselor: Konselor)` yang ada pada objek *RatingVM*. Kemudian objek *RatingVM* memanggil fungsi `submitRating(rating: Int, konselor: Konselor)` pada objek *DataService*. Objek *DataService* melakukan *request* ke Firebase dan mendapatkan respon bahwa pasien sukses memberi rating ke konselor. Kemudian objek *RatingVM* memanggil fungsi `onFinishGiveRating()` pada *RatingVC* dan sistem navigasi ke halaman *MapVC*.

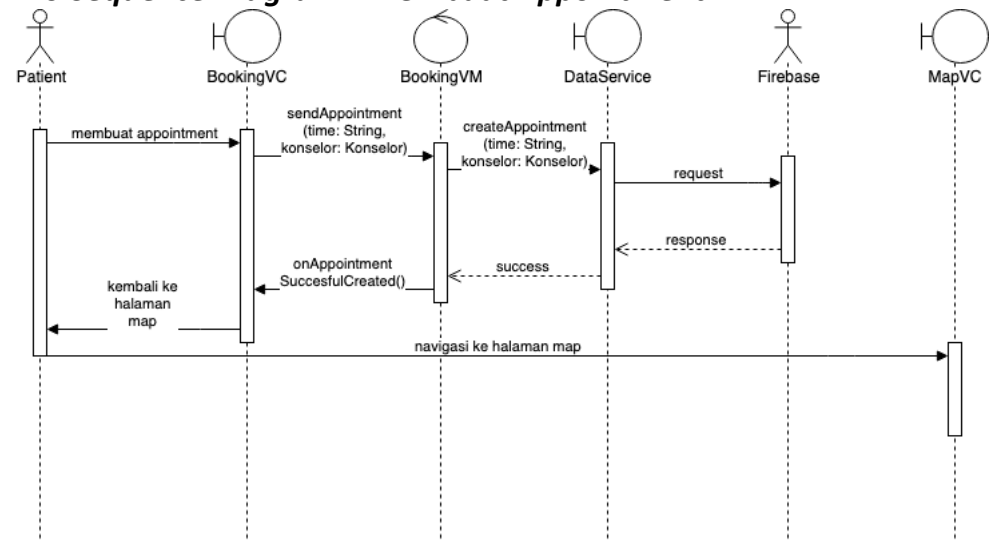
5.2.9 *Sequence Diagram* - Melihat Riwayat Konseling



Gambar 5.10 Sequence Diagram - Melihat Riwayat Konseling

Sequence diagram melihat riwayat konseling dari aktor pasien digambarkan pada Gambar 5.10. Aktor pasien melihat riwayat konseling pada halaman *HistoryVC* yang kemudian memanggil fungsi `fetchHistoryOrders()` dari objek *HistoryVM*. Kemudian objek *HistoryVM* memanggil fungsi `getHistoryOrder()` pada objek *DataService*. Setelah itu objek *DataService* melakukan *request* ke Firebase dan mendapatkan respon berupa *array* bertipe data *History*. Kemudian objek *HistoryVM* melakukan *trigger* kepada *historyTableView* untuk memanggil fungsi `reloadData()` agar halaman *HistoryVC* menampilkan data riwayat konseling kepada pasien.

5.2.10 *Sequence Diagram* - Membuat *Appointment*

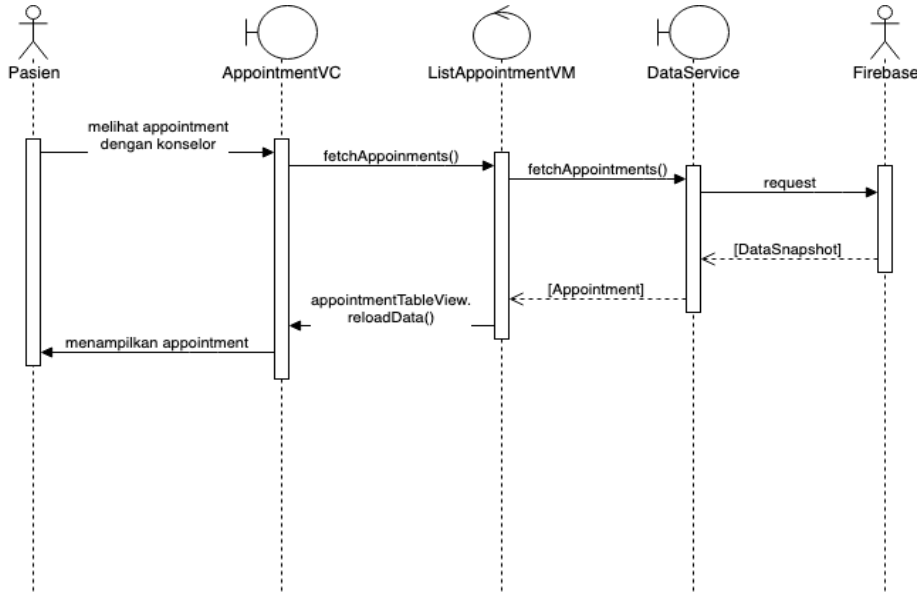


Gambar 5.11 Sequence Diagram - Membuat Appointment

Sequence diagram membuat *appointment* dari aktor pasien digambarkan pada Gambar 5.11. Aktor pasien membuat *appointment* pada halaman

BookingVC dan memilih jam yang tersedia pada hari ini. Kemudian dari *BookingVC* memanggil fungsi `sendAppointment(time: String, konselor: Konselor)` pada objek *BookingVM*. Kemudian objek *BookingVM* memanggil fungsi `createAppointment(time: String, konselor: Konselor)` pada objek *DataService*. Setelah itu objek *DataService* melakukan *request* ke Firebase dan mendapatkan respon bahwa *appointment* berhasil dibuat sehingga objek *BookingVM* memanggil fungsi `onAppointmentSuccesfulCreated()` pada *BookingVC*. Kemudian sistem kembali menampilkan halaman *MapVC* sebagai halaman utama.

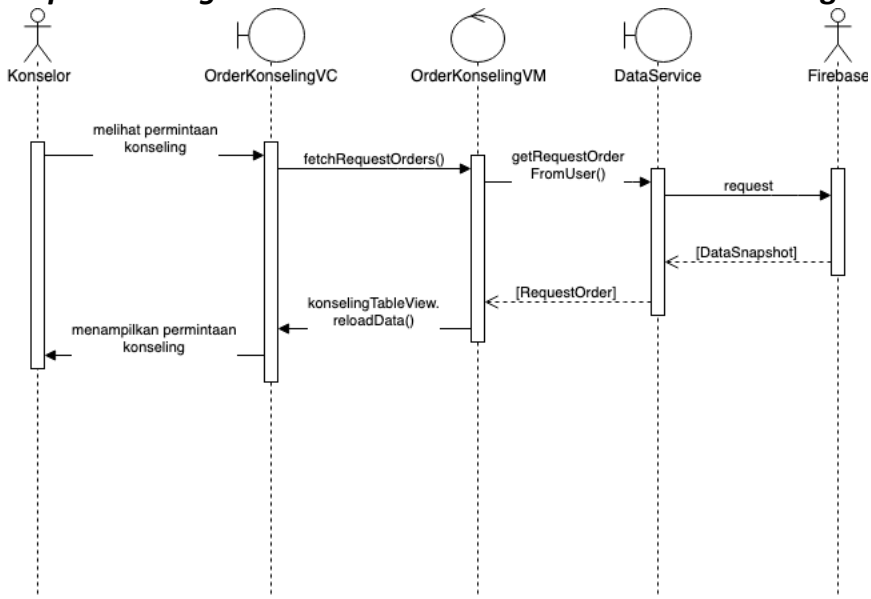
5.2.11 *Sequence Diagram* - Melihat Daftar Appointment Dengan Konselor



Gambar 5.12 Sequence Diagram - Melihat Daftar Appointment Dengan Konselor

Sequence diagram melihat daftar appointment dengan konselor digambarkan pada Gambar 5.12. Aktor pasien melihat daftar *appointment* pada halaman *AppointmentVC* yang kemudian memanggil fungsi `fetchAppointments()` dari objek *ListAppointmentVM*. Kemudian objek *ListAppointmentVM* memanggil fungsi `fetchAppointments()` pada objek *DataService*. Setelah itu objek *DataService* melakukan *request* ke Firebase dan mendapatkan respon berupa *array* bertipe data *Appointment*. Kemudian objek *ListAppointmentVM* melakukan *trigger* kepada *appointmentTableView* untuk memanggil fungsi `reloadData()` agar halaman *AppointmentVC* menampilkan data riwayat konseling kepada pasien.

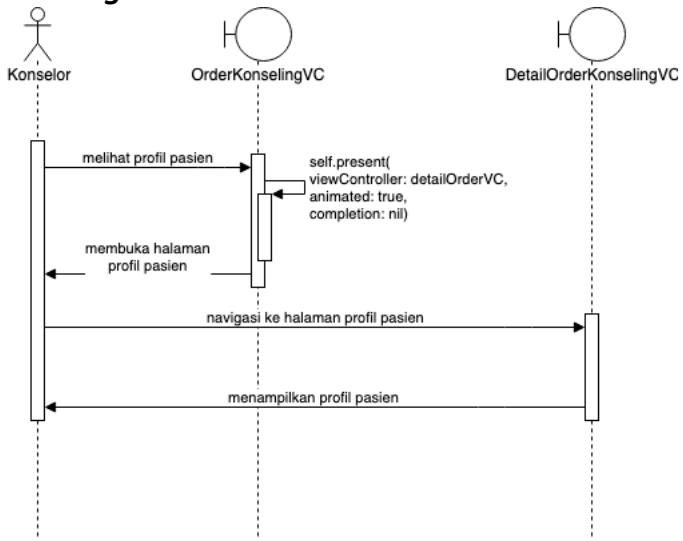
5.2.12 *Sequence Diagram* - Melihat Daftar Permintaan Konseling



Gambar 5.13 Sequence Diagram - Melihat Daftar Permintaan Konseling

Sequence diagram melihat daftar permintaan konseling oleh aktor konselor digambarkan pada Gambar 5.13. Aktor konselor melihat daftar permintaan konseling pada halaman *OrderKonselingVC* yang kemudian memanggil fungsi `fetchRequestOrders()` dari objek *OrderKonselingVM*. Kemudian objek *OrderKonselingVM* memanggil fungsi `getRequestOrderFromUser()` pada objek *DataService*. Setelah itu objek *DataService* melakukan *request* ke Firebase dan mendapatkan respon berupa *array* bertipe data *RequestOrder*. Kemudian objek *OrderKonselingVM* melakukan *trigger* kepada *konselingTableView* untuk memanggil fungsi `reloadData()` agar halaman *OrderKonselingVC* menampilkan data riwayat konseling kepada pasien.

5.2.13 *Sequence Diagram* - Melihat Detail Profil Pasien

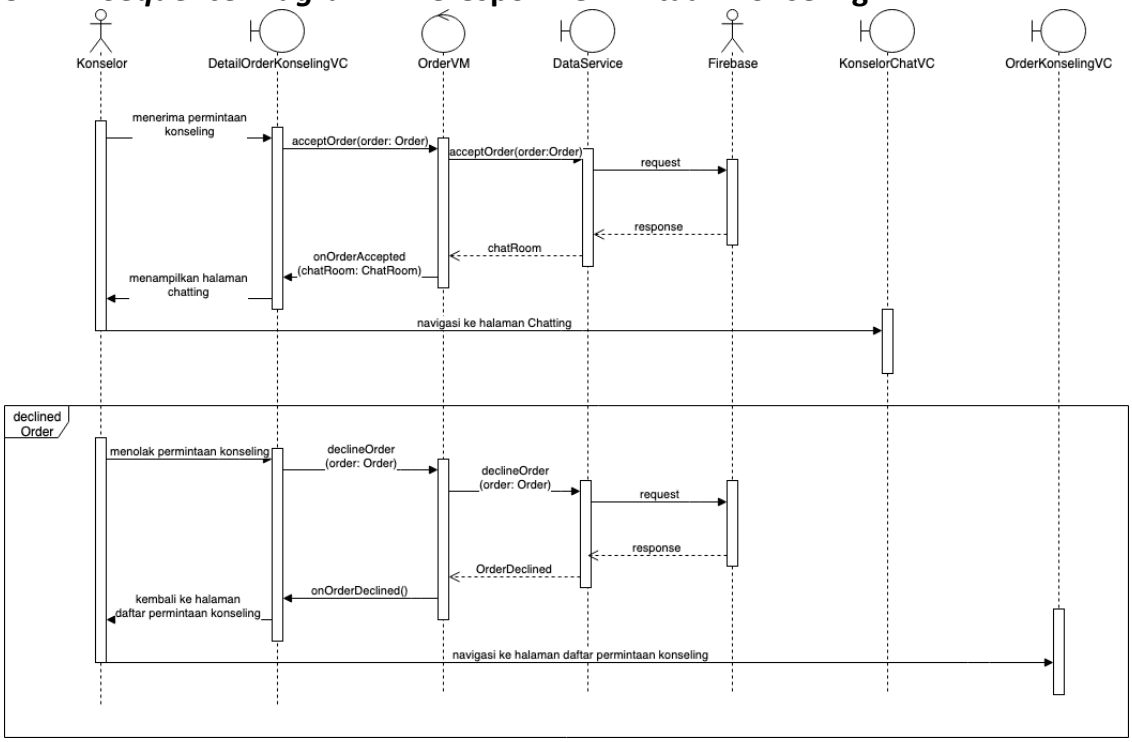


Gambar 5.14 Sequence Diagram - Melihat Profil Pasien

Sequence diagram melihat profil pasien oleh aktor konselor digambarkan pada Gambar 5.14. Pada halaman *OrderKonselingVC*, aktor konselor memilih

pasien yang tersedia untuk dapat melihat detail profilnya. Kemudian pada sistem membuka halaman profil pasien yang bernama *DetailOrderKonselingVC*.

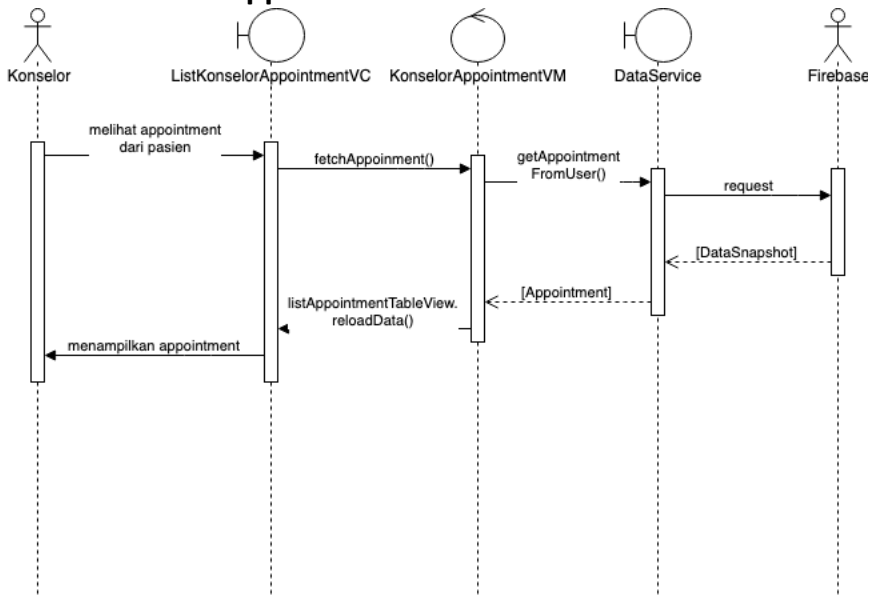
5.2.14 *Sequence Diagram* - Merespon Permintaan Konseling



Gambar 5.15 Sequence Diagram - Merespon Permintaan Konseling

Sequence diagram merespon permintaan konseling dari aktor konselor digambarkan pada Gambar 5.15. Konselor menekan tombol terima konseling yang disediakan oleh *DetailOrderKonselingVC* dan memanggil fungsi `acceptOrder(order: Order)` dari objek *OrderVM*. Kemudian objek *OrderVM* memanggil fungsi `acceptOrder(order: Order)` dari objek *DataService*. Setelah itu objek *DataService* melakukan *request* ke Firebase dan mendapatkan respon berupa objek *chatRoom*. Kemudian objek *OrderVM* memanggil fungsi *onOrderAccepted(chatRoom: ChatRoom)* dan menampilkan halaman *KonselorChatVC*.

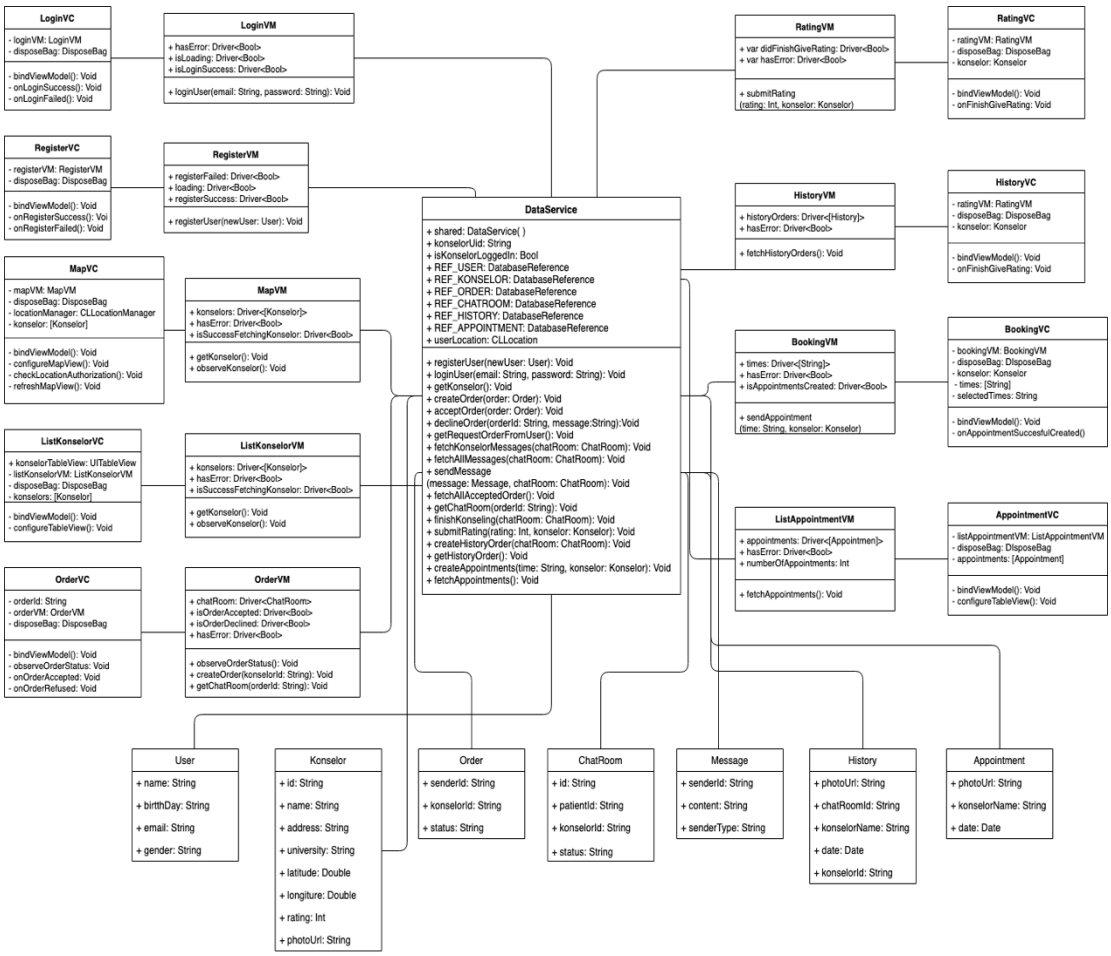
5.2.15 Melihat Daftar Appointment Dari Pasien



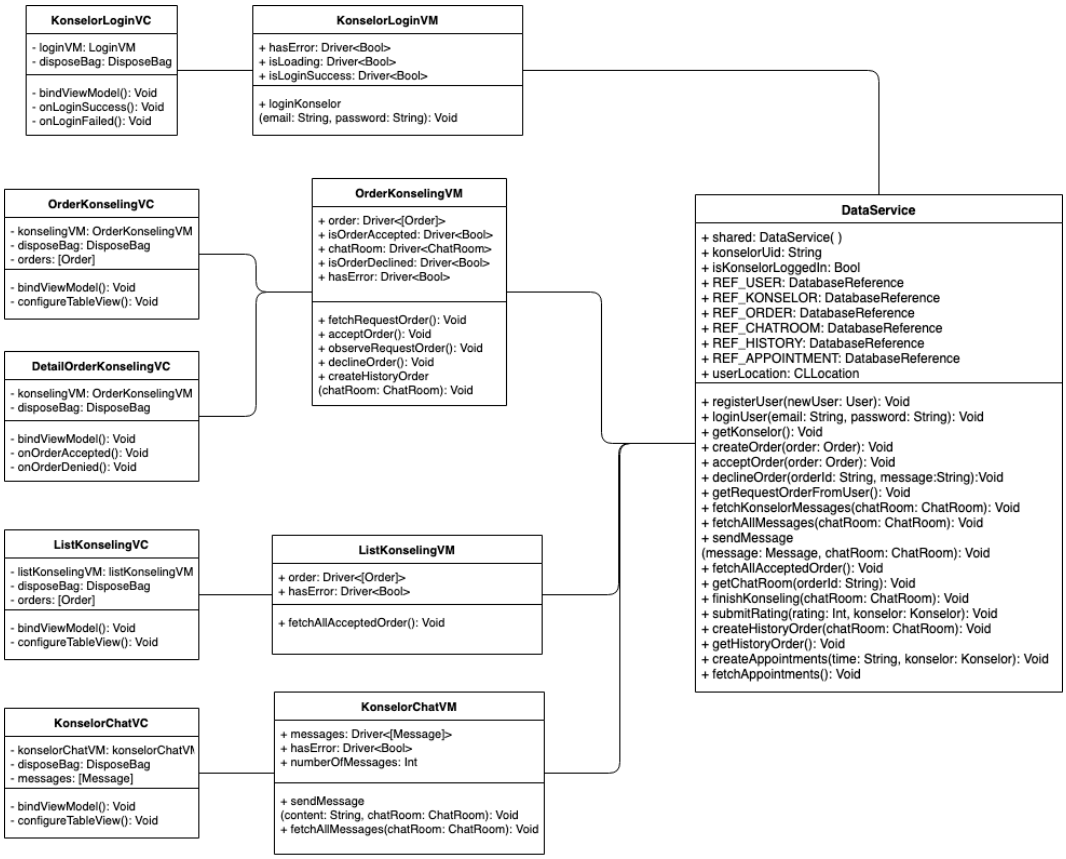
Gambar 5.16 Sequence Diagram - Melihat Daftar Appointment Dari Pasien

Sequence diagram melihat daftar *appointment* dari pasien digambarkan pada Gambar 5.16. Aktor konselor melihat daftar *appointment* pada halaman *ListKonselorAppointmentVC* yang kemudian memanggil fungsi `fetchAppointments()` dari objek *KonselorAppointmentVM*. Kemudian objek *KonselorAppointmentVM* memanggil fungsi `getAppointmentFromUser()` pada objek *DataService*. Setelah itu objek *DataService* melakukan *request* ke Firebase dan mendapatkan respon berupa *array* bertipe data *Appointment*. Kemudian objek *KonselorAppointmentVM* melakukan *trigger* kepada *listAppointmentTableView* untuk memanggil fungsi `reloadData()` agar halaman *ListKonselorAppointmentVC* menampilkan data riwayat konseling kepada pasien.

5.3 Class Diagram



Gambar 5.17 Class Diagram Aplikasi Pasien



Gambar 5.18 Class Diagram - Aplikasi Sisi Konselor

Class Diagram adalah diagram yang menggambarkan arsitektur objek dan class pada sistem beserta hubungan keterkaitan antar satu dengan yang lain dan menjelaskan bagaimana tiap modul dikelompokkan. *Class Diagram* yang dibuat penelitian ini dikelompokkan menjadi 2, yaitu *Class Diagram* untuk aplikasi sisi pasien dan *Class Diagram* untuk aplikasi sisi konselor. *Class Diagram* untuk pasien digambarkan pada Gambar 5.17 dan *Class Diagram* untuk konselor digambarkan pada Gambar 5.18. Penjelasan tiap *class* pada *Class Diagram* aktor pasien dijelaskan pada Tabel 5.1 dan penjelasan tiap *class* pada *Class Diagram* aktor konselor dijelaskan pada Tabel 5.2.

Tabel 5.1 Penjelasan Class Diagram Pasien

No	*Class*	*Responsibility*
1.	LoginVC	Menyediakan tempat bagi pasien untuk login ke sistem dan memonitor validasi login pada fungsi `onLoginSuccesfull` dan `onLoginFailed`
2.	LoginVM	Mengatur *business-logic* untuk login pada *class LoginVC* dengan fungsi `loginUser`
3.	RegisterVC	Menyediakan tempat bagi pasien untuk register ke sistem dan memonitor validasi register pada fungsi `onRegisterSuccesfull` dan `onRegisterFailed`

4.	RegisterVM	Mengatur *business-logic* untuk register pada class *RegisterVC* dengan fungsi `registerUser`
5.	MapVC	Menampilkan lokasi konselor yang terdekat dengan lokasi pasien pada fungsi `configureMapView`
6.	MapVM	Mengatur *business-logic* untuk menampilkan lokasi konselor pada class *MapVC* dengan fungsi `observeKonselor`
7.	ListKonselorVC	Menampilkan daftar konselor terdekat dengan pasien pada fungsi `configureTableView`
8.	ListKonselorVM	Mengatur *business-logic* untuk menampilkan daftar konselor terdekat dengan pasien pada class *ListKonselorVM* dengan fungsi `observeKonselor`
9.	OrderVC	Menampilkan halaman menunggu konfirmasi permintaan konseling dari pasien ke konselor pada fungsi `observeOrderStatus`
10.	OrderVM	Mengatur *business-logic* memonitor status permintaan konseling pada *class OrderVC* dengan fungsi `observeOrderStatus`
11.	RatingVC	Menampilkan halaman rating setelah pasien selesai melakukan konseling. Pada *class* ini menyediakan tempat bagi pasien untuk memberi rating pada konselor pada fungsi `onFinishGiveRating`
12.	RatingVM	Mengatur *business-logic* untuk pasien dapat memberi rating pada *class RatingVC* dengan fungsi `submitRating`
13.	HistoryVC	Menampilkan halaman riwayat konseling antara pasien dengan konselor pada fungsi `fetchHistoryOrders`
14.	HistoryVM	Mengatur *business-logic* untuk menampilkan riwayat konseling antara pasien dengan konselor pada *class HistoryVC* dengan fungsi `fetchHistoryOrders`
15.	BookingVC	Menyediakan tempat bagi pasien untuk membuat *appointment* dengan konselor pada fungsi `onAppointmentSuccesfulCreated`
16.	BookingVM	Mengatur business-logic untuk pasien dapat membuat *appointment* dengan konselor pada *class BookingVC* dengan fungsi `sendAppointment`
17.	AppointmentVC	Menampilkan daftar *appointment* pasien dengan konselor pada fungsi `configureTableView`
18.	AppointmentVM	Mengatur *business-logic* untuk pasien menampilkan daftar *appointment* pada *class AppointmentVC* dengan fungsi `fetchAppointment`
19.	User	Merupakan representasi objek dari pasien
20.	Konselor	Merupakan representasi objek dari konselor
21.	Order	Merupakan representasi objek order yaitu transaksi

		konseling antara pasien dengan konselor.
22.	ChatRoom	Merupakan representasi objek untuk ruang percakapan untuk konseling antara pasien dengan konselor.
23.	Message	Merupakan representasi objek dari pesan.
24.	History	Merupakan representasi objek dari riwayat konseling pada pasien.
25.	Appointment	Merupakan representasi objek dari *appointment* yang dibuat pasien.

Tabel 5.2 Penjelasan Class Diagram Konselor

No	*Class*	*Responsibility*
1.	KonselorLoginVC	Menyediakan tempat bagi konselor untuk login ke sistem dan memonitor validasi login pada fungsi `onLoginSuccesfull` dan `onLoginFailed`
2.	KonselorLoginVM	Mengatur *business-logic* untuk login pada *class LoginVC* dengan fungsi `loginKonselor`
3.	OrderKonselingVC	Menampilkan daftar permintaan konseling dari pasien pada fungsi `configureTableView`
4.	DetailOrderKonselingVC	Menampilkan detail profil pasien yang mengirim permintaan konseling dan memonitor apakah konselor menerima atau menolak konseling pada fungsi `onOrderAccepted` dan `onOrderDenied.`
5.	OrderKonselingVM	Mengatur *business-logic* pada *class OrderKonselingVC* dan *DetailOrderKonselingVC* untuk memonitor status dari order pada `observeOrderStatus.`
6.	ListKonselingVC	Menampilkan daftar konseling dengan pasien pada fungsi `configureTableView`
7.	ListKonselingVM	Mengatur *business-logic* pada *class ListKonselingVC* untuk memperoleh data konseling pada fungsi `fetchAllAcceptedOrder`
8.	KonselorChatVC	Menampilkan halaman chat antara konselor dengan pasien pada fungsi `configureTableView`
9.	KonselorChatVM	Mengatur *business-logic* pada *class KonselorChatVM* untuk memperoleh data pesan antara konselor dan pasien pada fungsi `fetchAllMessages`

5.4 Perancangan Tabel Data

Perancangan tabel data dibuat untuk merepresentasikan entitas objek yang digunakan dalam pengembangan aplikasi yang dilakukan. Proses penyimpanan data yang dilakukan pada aplikasi yang dikembangkan menggunakan layanan Firebase *Realtime Database.* Entitas objek yang didefinisikan terdiri dari *Patient*, *Konselor*, *Order*, *Appointment*, dan *ChatRoom*. Berikut merupakan hasil perancangan dari tiap entitas yang telah didefinisikan.

Tabel 5.3 Perancangan Data Patient

No	Nama Field	Deskripsi
1.	name	Merupakan nama lengkap pasien
2.	email	Merupakan alamat email dari pasien
3.	gender	Merupakan jenis kelamin dari pasien
4.	birthday	Merupakan tanggal lahir dari pasien
5.	latitude	Merupakan satuan posisi latitude dari pasien
6.	longitude	Merupakan satuan posisi longitude dari pasien

Tabel 5.4 Perancangan Data Konselor

No	Nama Field	Deskripsi
1.	name	Merupakan nama lengkap konselor
2.	email	Merupakan alamat email dari konselor
3.	address	Merupakan alamat dari pasien
4.	university	Merupakan nama universitas tempat konselor menempuh studinya
5.	photoUrl	Merupakan link url foto profil dari konselor
6.	patient_count	Merupakan jumlah total pasien yang pernah konselor layani dalam aplikasi
7.	rating	Merupakan rating dari konselor
8.	schedule	Merupakan objek berisi jadwal tersedia dari konselor

Tabel 5.5 Perancangan Data Order

No	Nama Field	Deskripsi
1.	senderId	Merupakan id pasien dalam permintaan konseling
2.	konselorId	Merupakan id konselor dalam permintaan konseling
3.	status	Merupakan keterangan order

Tabel 5.6 Perancangan Data Appointment

No	Nama Field	Deskripsi
1.	date	Merupakan keterangan waktu dari *appointment*
2.	konselorName	Merupakan nama konselor dalam *appointment*
3.	patientId	Merupakan id pasien dalam *appointment*
4.	photoUrl	Merupakan link url dari foto profil konselor

Tabel 5.7 Perancangan Data ChatRoom

No	Nama Field	Deskripsi
1.	patientId	Merupakan id pasien dalam permintaan konseling
2.	konselorId	Merupakan id konselor dalam permintaan konseling
3.	status	Merupakan keterangan kondisi dari sesi konseling
4.	messages	Merupakan pesan antara pasien dengan konselor dalam sesi konseling

5.5 Perancangan Algoritme

Tabel 5.8 Perancangan Algoritme - Memperoleh Data Konselor Terdekat

```
1   func getKonselor(completion: (_ konselor: [Konselor]-> Void) {
2        input: -
3        output: [Konselor]
4         var konselorArray: [Konselor]()
5   REF_KONSELOR.observeSingleEvent() { (dataSnapshot) in
6       var rawKonselor = dataSnapshot as [DataSnapshot]
7       for konselor in rawKonselor {
8           let newKonselor = Konselor(konselor: konselor)
9           let latitude = newKonselor.latitude
10          let longitude = newKonselor.longitude
11          newKonselor.distance      =      calculateDistance(latitude,
12  longitude)
13          konselorArray.append(newKonselor)
14      }
15      konselorArray.sorted(by: $0.distance < $1.distance)
16           completion(konselorArray)
17  }
18
19  func  calculateDistance(latitude:  Double,  longitude:  Double)  ->
20  Double {
21      var userCoordinate = CLLocation()
22      var   konselorCoordinate   =   CLLocation(latitude:   latitude,
23  longitude: longitude)
24      return userCoordinate.distance(from: konselorCoordinate) / 1000
25  }
```

Tabel 5.9 Perancangan Algoritme - Mengirim Permintaan Konseling

```
1   func createOrder(order: Order) {
2      input: Order
3      ouput: Bool
4
5      let newOrder = Dictionary<String,String>()
6      let newOrderId = REF_ORDER.childByAutoId().key
7      REF_ORDER.child(newOrderId).updateChild(newOrder)
8      completion(true)
9   }
```

Tabel 5.10 Perancangan Algoritme - Membuat Appointment

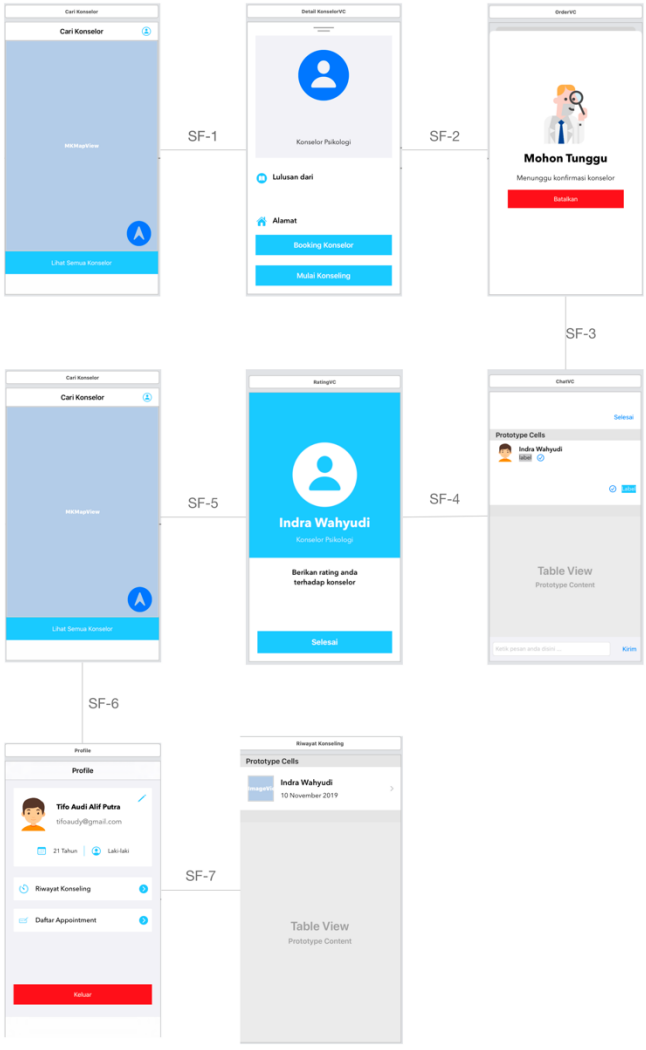
```
1   func createAppointment(time: String, konselor: Konselor) {
2       input: time, konselor
3       output: Bool
4
5       let uid = Auth.auth().currentUser.uid
6       let newAppointment = Dictionary<String,String>()
7       REF_APPOINTMENT.childByAutoId.updateChild(newAppointment)
8       completion(true)
```

9	}

Perancangan algoritme dilakukan untuk membuat prosedur yang berisi langkah-langkah yang diperlukan dalam menyelesaikan sebuah permasalahan. Dalam aplikasi yang dikembangkan pada penelitian ini, terdapat 3 algoritme fitur utama pada aplikasi yaitu memperoleh konselor terdekat dengan pasien yang dijelaskan pada Tabel 5.8, mengirim permintaan konseling yang dijelaskan pada Tabel 5.9, dan membuat *appointment* dengan konselor yang dijelaskan pada Tabel 5.10.

5.6 Perancangan Antarmuka

5.6.1 Screenflow - Aplikasi Sisi Pasien



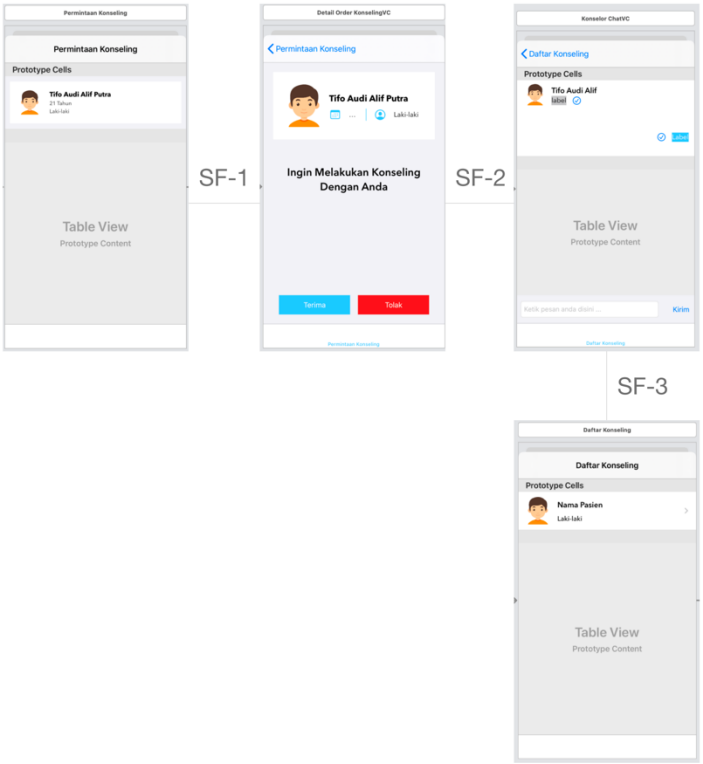
Gambar 5.19 Screenflow Aplikasi Pasien

Tabel 5.11 Penjelasan Screenflow Pasien

No.	Flow Identifier	Penjelasan

1.	SF-1	Pasien memilih konselor dan menampilkan profil konselor.
2.	SF-2	Pasien mengirim permintaan konseling dan menampilkan halaman menunggu konfirmasi konselor.
3.	SF-3	Konselor menerima permintaan dari pasien dan menampilkan halaman *chat.*
4.	SF-4	Pasien telah selesai melakukan konseling dan menampilkan halaman *rating.*
5.	SF-5	Pasien telah selesai memberi rating ke konselor dan kembali ke halaman peta.
6.	SF-6	Dari halaman peta, pasien menampilkan halaman profil
7.	SF-7	Pasien menampilkan halaman riwayat konseling.

5.6.2 Screenflow - Aplikasi Sisi Konselor



Gambar 5.20 Screenflow Aplikasi Konselor

Tabel 5.12 Penjelasan Screenflow Konselor

No	Flow Identifier	Penjelasan
1	SF-1	Konselor melihat daftar permintaan konseling dari pasien dan menampilkan profil pasien
2	SF-2	Konselor menerima permintaan konseling dari pasien dan menampilkan halaman *chat*
3	SF-3	Sesi konseling telah selesai dan sistem menampilkan halaman awal pada sistem

Screenflow dibuat agar memudahkan dalam memahami alur sistem dalam membangun *user interface.* Pada aplikasi sisi pasien, *screenflow* yang dirancang dapat dilihat pada Gambar 5.19 dan *screenflow* aplikasi sisi konselor yang dirancang dapat dilihat pada Gambar 5.20.

5.7 Implementasi Perangkat Lunak

Pada tahap ini membahas tentang implementasi untuk membangun aplikasi dari hasil perancangan yang telah didefinisikan pada tahap sebelumnya. Pembahasan mengenai tahap implementasi meliputi spesifikasi lingkungan sistem dalam tahap implementasi, batasan pada tahap implementasi, implementasi perancangan basis data, implementasi kode program, dan implementasi antarmuka.

5.7.1 Spesifikasi Lingkungan Sistem

5.7.1.1 Spesifikasi Perangkat Keras

Spesifikasi perangkat keras yang digunakan pada penelitian ini dijelaskan pada Tabel 5.13.

Tabel 5.13 Spesifikasi Perangkat Keras

Processor	3,1 GHz Dual-Core Intel Core i5
Ram	8 GB 2133 MHz LPDDR3
GPU	Intel Iris Plus Graphics 650 1536 MB
Storage	128GB PCI-r based flash storage
OS	macOS Catalina 10.15

5.7.1.2 Spesifikasi Perangkat Lunak

Spesifikasi perangkat lunak yang digunakan pada penelitian ini dijelaskan pada Tabel 5.14.

Tabel 5.14 Spesifikasi Perangkat Lunak

IDE	Xcode 11.1
Design Tools	*Draw.io* dan *Storyboard*
Document	Microsoft Office Word for Mac Ver. 16.16
Programming Language	Swift 5.1
Device Testing	iPhone 7

5.7.2 Batasan Implementasi

Adapun batasan implementasi pada penelitian yang dilakukan adalah sebagai berikut:

1. Aplikasi dibangun pada platform iOS dengan spesifikasi OS minimal iOS 13.
2. Sistem basis data yang digunakan menggunakan layanan Realtime Database milik Firebase.

5.7.3 Implementasi Tabel Data

Gambar 5.21 Implementasi Tabel Data

Implementasi tabel data dilakukan dengan menggunakan *realtime database* milik Firebase. Hasil implementasi tabel data pada aplikasi yang dibangun digambarkan pada Gambar 5.21.

5.7.4 Implementasi Kode Program

Implementasi kode program pada penelitian ini menggunakan bahasa *Swift* yang merupakan bahasa resmi untuk pemrograman aplikasi perangkat bergerak pada sistem operasi iOS. Implementasi dilakukan atas dasar hasil perancangan algoritme pada sub bab sebelumnya. Implementasi kode program dijelaskan pada Tabel 5.15 hingga Tabel 5.17.

Tabel 5.15 Implementasi Kode Program - Mendapatkan Konselor Terdekat

```
1   func getKonselor(success: @escaping (_ konselor:
2   [Konselor]) -> (), failure: @escaping ()-> ()) {
3
4         var konselorArray = [Konselor]()
5         REF_KONSELOR.observeSingleEvent(of: .value) { [unowned
6   self] (dataSnapshot) in
7           guard let rawKonselor =
8   dataSnapshot.children.allObjects as? [DataSnapshot] else {
9             failure()
10            return
11          }
12
13          for konselor in rawKonselor {
14            let name = konselor.childSnapshot(forPath:
15  "name").value as! String
16            let address = konselor.childSnapshot(forPath:
17  "address").value as! String
18            let university = konselor.childSnapshot(forPath:
19  "university").value as! String
20            let latitude = konselor.childSnapshot(forPath:
21  "location").childSnapshot(forPath: "latitude").value as!
22  Double
23            let longitude = konselor.childSnapshot(forPath:
24  "location").childSnapshot(forPath: "longitude").value as!
25  Double
26            let isOnline = konselor.childSnapshot(forPath:
27  "isOnline").value as! Bool
28            let distance = self.calculateDistance(konselor:
29  latitude, konselor: longitude)
30            let rating = konselor.childSnapshot(forPath:
31  "rating").value as! Double
32            let patientCount = konselor.childSnapshot(forPath:
33  "patient_count").value as! Int
34            let photoUrl = konselor.childSnapshot(forPath:
35  "photoUrl").value as! String
36            let mondaySchedule =
37  konselor.childSnapshot(forPath:
38  "schedule").childSnapshot(forPath: "Monday").value as! String
39            let tuesdaySchedule =
40  konselor.childSnapshot(forPath:
41  "schedule").childSnapshot(forPath: "Tuesday").value as!
42  String
43            let wednesdaySchedule =
44  konselor.childSnapshot(forPath:
45  "schedule").childSnapshot(forPath: "Wednesday").value as!
46  String
47            let thursdaySchedule =
48  konselor.childSnapshot(forPath:
```

```
"schedule").childSnapshot(forPath:    "Thursday").value    as!
String
           let                fridaySchedule                =
konselor.childSnapshot(forPath:
"schedule").childSnapshot(forPath: "Friday").value as! String
           let  konselorSchedule  =  KonselorSchedule(monday:
mondaySchedule,    tuesday:    tuesdaySchedule,    wednesday:
wednesdaySchedule,    thursday:    thursdaySchedule,    friday:
fridaySchedule)
           let newKonselor = Konselor(id: konselor.key, name:
name,  address:  address,  university:  university,  latitude:
latitude, longitude: longitude, isOnline: isOnline, distance:
distance,   patientCount:   patientCount,   rating:   rating,
photoUrl: photoUrl, schedule: konselorSchedule)

           if newKonselor.isOnline {
             konselorArray.append(newKonselor)
           }
         }
         konselorArray    =    konselorArray.sorted(by:    {
$0.distance < $1.distance })
         success(konselorArray)
       }
  }

   func calculateDistance(konselor latitude: Double, konselor
longitude: Double) -> Double {
       guard let userCoordinate = self.userLocation else {
         return 0
       }
       let    konselorCoordinate    =    CLLocation(latitude:
latitude, longitude: longitude)
       let          distanceInKm:          Double          =
userCoordinate.distance(from: konselorCoordinate) / 1000
       return round(100 * distanceInKm)/100
  }
```

Tabel 5.16 Implementasi Kode Program - Mengirim Permintaan Konseling

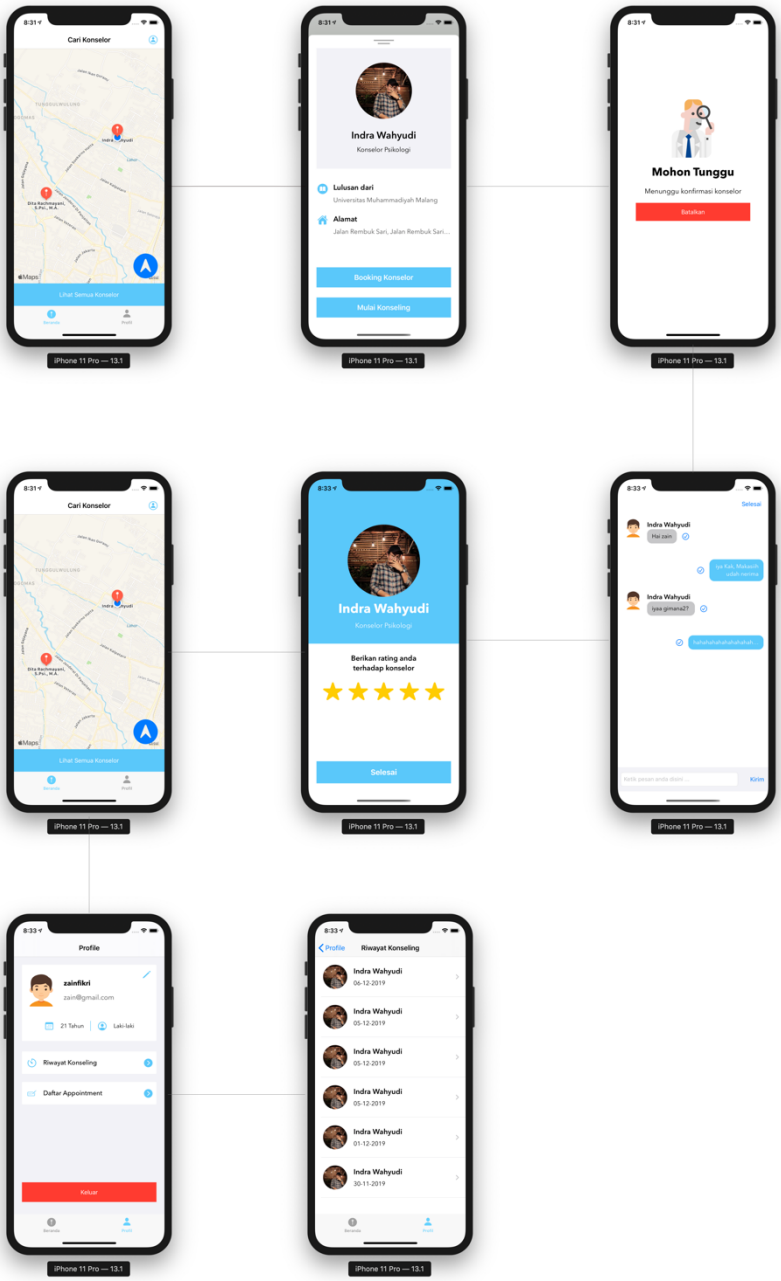
```
   func  createOrder(order:  Order,  completion:  @escaping  (_
success: Bool, _ orderId: String)->()) {

       let newOrder = [
         "senderId": order.senderId,
         "konselorId": order.konselorId,
         "status": order.status.rawValue
       ]

       guard  let  newOrderId  =  REF_ORDER.childByAutoId().key
else {
         return
       }


REF_ORDER.child(newOrderId).updateChildValues(newOrder)
       completion(true, newOrderId)
  }
```

Tabel 5.17 Membuat Appointment

```
1    func createAppointment(time: String, konselor: Konselor,
2   completion: @escaping(_ success: Bool)->()) {
3          guard let uid = Auth.auth().currentUser?.uid else {
4   return }
5
6          let newAppointment = [
7            "photoUrl": konselor.photoUrl,
8            "date": time,
9            "patientId": uid,
10           "konselorName": konselor.name
11         ]
12
13
14  REF_APPOINTMENT.childByAutoId().updateChildValues(newAppointm
15  ent)
16         completion(true)
17    }
```

5.7.5 Implementasi Antarmuka

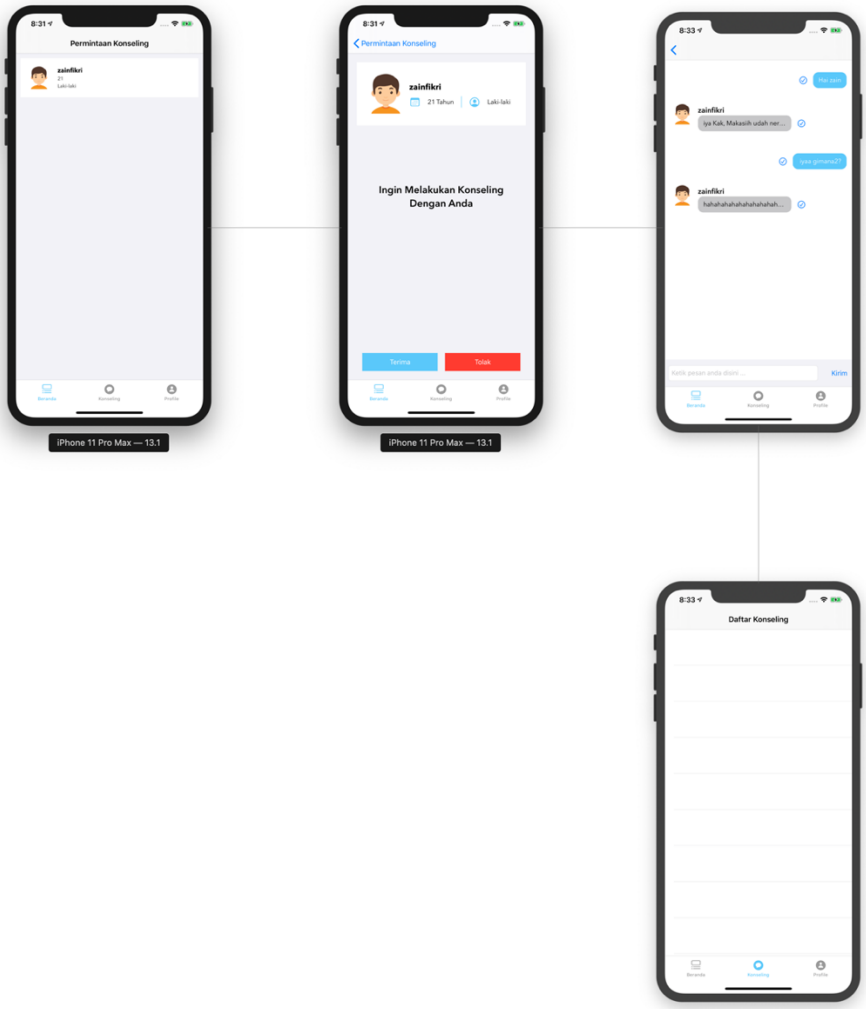
Implementasi antarmuka merupakan proses mengimplementasikan perancangan antarmuka yang dilakukan dengan menggunakan *storyboard* pada Xcode sebagai *Integrated Development Environment* untuk pengembangan aplikasi perangkat bergerak pada sistem operasi iOS. Implementasi antarmuka dibagi menjadi dua bagian, yaitu implementasi pada aplikasi pasien dan implementasi pada aplikasi konselor. Implementasi antarmuka pada aplikasi pasien digambarkan pada dan implementasi antarmuka pada Gambar 5.22 dan aplikasi konselor digambarkan pada Gambar 5.23.



Gambar 5.22 Implementasi Antarmuka Pasien

Alur pada aplikasi sisi pasien dimulai ketika pasien memilih konselor yang tersedia pada halaman peta. Kemudian aplikasi menampilkan detail profil dari konselor yang dipilih oleh pasien. Pasien mengirim permintaan konseling kepada konselor dengan menekan tombol yang telah disediakan. Kemudian aplikasi menampilkan halaman untuk pasien menunggu konfirmasi konselor. Apabila konselor menerima permintaan tersebut maka aplikasi menampilkan halaman *chatting.* Proses konseling secara online telah terjadi antara pasien dan konselor. Kemudian pasien mengakhiri konseling dengan menekan tombol selesai yang disediakan. Aplikasi akan menampilkan halaman rating untuk pasien memberi rating kepada konselor. Kemudian aplikasi kembali menampilkan halaman peta, pasien dapat melihat riwayat konselingnya dengan membuka *tab profile* terlebih

dahulu. Kemudian dari halaman profil, pasien dapat melihat riwayat konselingnya dengan konselor dengan menekan menu yang disediakan.



Gambar 5.23 Implementasi Antarmuka Konselor

Alur pada aplikasi sisi konselor dimulai pada halaman untuk menampilkan permintaan konseling. Konselor dapat melihat profil pasien dengan memilih permintaan konseling yang ditampilkan. Apabila konselor menerima permintaan tersebut maka aplikasi menampilkan halaman *chatting.* Jika sesi konseling telah selesai maka aplikasi kembali menampilkan halaman awal.

BAB 6 PENGUJIAN

Pada bab pengujian membahas tentang pengujian validasi menggunakan *blackbox testing* dan pengujian usabilitas pada aplikasi. Pengujian validasi dilakukan di setiap fitur untuk memastikan aplikasi berjalan dengan sesuai dan memastikan tidak ada *bug.* Pengujian validasi dilakukan dengan cara menyediakan input dan memastikan output sesuai dengan harapan. Pengujian usabilitas dilakukan untuk mengukur tingkat kemudahan dalam penggunaan aplikasi.

6.1 Pengujian Validasi

6.1.1 Login

Pengujian validasi yang dilakukan pada *usecase* login dijelaskan pada Tabel 6.1.

Tabel 6.1 Pengujian Validasi - Login

Usecase	*Target*	*Test Step*	*Expected Result*	*Actual Result*
Login	Proses login berhasil.	Mengisi *email* dan *password* pada *field* yang telah disediakan.	Login berhasil dan aplikasi menampilkan halaman utama berupa peta.	Valid
	Proses login gagal.	Mengisi *email* dan *password* yang tidak valid.	Aplikasi menampilkan pesan bahwa login tidak berhasil.	Valid

6.1.2 Register

Pengujian validasi yang dilakukan pada *usecase* register dijelaskan pada Tabel 6.2.

Tabel 6.2 Pengujian Validasi - Register

Usecase	*Target*	*Test Step*	*Expected Result*	*Actual Result*
Register	Proses register berhasil.	Mengisi data diri pada *field* yang telah disediakan.	Register berhasil dan aplikasi menampilkan halaman utama berupa peta.	Valid
	Proses register gagal.	Mengisi data diri yang tidak valid.	Aplikasi menampilkan pesan bahwa register tidak berhasil.	Valid

6.1.3 Melakukan Konseling

Pengujian validasi yang dilakukan pada *usecase* melakukan konseling dijelaskan pada Tabel 6.3.

Tabel 6.3 Pengujian Validasi - Melakukan Konseling

Usecase	*Target*	*Test Step*	*Expected Result*	*Actual Result*
Melakukan Konseling	Pasien dapat melakukan konseling secara online	Pasien memilih konselor dan mengirim permintaan konseling.	Aplikasi menampilkan halaman *chatting.*	Valid
	Konselor menolak permintaan konseling dari pasien	Pasien memilih konselor dan mengirim permintaan konseling.	Aplikasi menampilkan pesan bahwa permintaan konseling ditolak.	Valid

6.1.4 Membuat Appointment

Pengujian validasi yang dilakukan pada *usecase* membuat appoinment dijelaskan pada Tabel 6.4.

Tabel 6.4 Pengujian Validasi - Membuat Appointment

Usecase	*Target*	*Test Step*	*Expected Result*	*Actual Result*
Membuat Appointment	Proses membuat *appointment* berhasil	Pasien memilih waktu dari jadwal tersedia konselor dan menekan tombol kirim	Aplikasi menampilkan pesan appointment berhasil dibuat	Valid
	Proses membuat *appointment* tidak berhasil	Pasien menekan tombol kirim tanpa memilih jadwal yang tersedia	Aplikasi menampilkan pesan untuk memilih waktu yang tersedia dari jadwal konselor.	Valid

6.2 Pengujian Usabilitas

Pengujian usabilitas merupakan pengujian yang dilakukan untuk mengukur tingkat kemudahan dalam penggunaan sistem. Jumlah responden yang ditentukan pada penelitian ini adalah 5 orang untuk aplikasi sisi pasien dan 5 orang untuk aplikasi sisi konselor. Langkah pengujian usabilitas dimulai ketika responden melakukan semua skenario pengujian yang berisi tugas-tugas yang berbeda dalam mengoperasikan aplikasi. Setelah melakukan semua skenario pengujian, responden diminta untuk mengisi kuisoner untuk memberikan

penilaian terhadap aplikasi yang dikembangkan. Instrumen pertanyaan yang digunakan pada pengisian kuisoner menggunakan SUPER-Qm dimana tiap pertanyaan menggunakan skala likert sebagai skala penilaian. Keterangan skala nilai pada skala likert dijelaskan pada Tabel 6.5.

Tabel 6.5 Skala Likert

Nilai	Keterangan
1	Sangat tidak setuju
2	Tidak setuju
3	Netral
4	Setuju
5	Sangat Setuju

Setelah selesai dalam mengisi kuisoner, responden dapat memberikan kritik dan saran terkait aplikasi yang dikembangkan. Kritik dan saran tersebut sangat dibutuhkan untuk menjadi umpan balik pengembang dalam mengembangkan aplikasi lebih baik lagi sehingga dapat dilanjutkan implementasinya pada penelitian lain. Daftar pertanyaan SUPER-Qm dijelaskan pada Tabel 6.6.

Tabel 6.6 Daftar Pertanyaan SUPER-Qm

No	Pertanyaan
1	Aplikasi ini penting untuk saya
2	Aplikasi ini merupakan aplikasi pencari konselor psikologi terbaik yang pernah saya gunakan
3	Saya tidak tahu apakah ada aplikasi pencari konselor psikologi yang lebih baik dari aplikasi ini
4	Saya tidak akan menghapus aplikasi ini dari smartphone saya
5	Saya akan menyarankan aplikasi ini ke teman saya
6	Saya suka mengeksplorasi fitur-fitur yang terdapat dalam fitur ini
7	Aplikasi ini memiliki seluruh fitur dan fungsi yang saya butuhkan dalam mencari konselor psikologi terdekat
8	Saya akan sering membuka aplikasi ini setiap kali saya ingin konseling
9	Aplikasi ini menyenangkan
10	Aplikasi ini bekerja dengan baik dengan fitur-fitur lain yang ada pada smartphone saya (contohnya gps, peta)
11	Saya akan menggunakan aplikasi ini lagi ketika saya ingin mencari konselor untuk konseling
12	Desain dari aplikasi ini memudahkan saya dalam menemukan informasi yang saya inginkan
13	Menurut saya aplikasi ini menarik
14	Kemampuan aplikasi ini sesuai dengan yang saya butuhkan
15	Aplikasi ini sangat mudah dalam hal navigasi
16	Aplikasi ini mudah digunakan

6.2.1 Skenario Pengujian

Skenario pengujian merupakan daftar yang berisi tugas-tugas yang perlu dijalankan ketika mengoperasi aplikasi untuk melakukan pengujian usabilitas. Daftar skenario pengujian pada aplikasi sisi pasien pada penelitian ini dijelaskan pada Tabel 6.7 dan daftar skenario pengujian pada aplikasi sisi konselor dijelaskan pada

Tabel 6.7 Skenario Pengujian - Aplikasi Sisi Pasien

No	Skenario	Penjelasan
1	Daftar ke dalam aplikasi	Responden diminta untuk membuat akun baru pada aplikasi
2	Login ke aplikasi	Responden diminta login dengan akun yang sudah terdaftar
3	Melihat daftar konselor terdekat	Responden diminta menampilkan halaman daftar konselor terdekat
4	Melihat detail profil konselor	Responden diminta menampilkan halaman detail profil konselor
5	Mengirim permintaan konseling	Responden diminta mengirim permintaan konseling pada konselor
6	Melakukan konseling online dengan konselor	Responden diminta melakukan konseling secara online dengan konselor
7	Memberi rating ke konselor	Responden diminta memberi rating kepada konselor setelah melakukan konseling
8	Membuat appointment dengan konselor	Responden diminta membuat appointment dengan konselor
9	Melihat daftar riwayat konseling	Responden diminta membuka halaman riwayat konseling

Tabel 6.8 Skenario Pengujian - Aplikasi Sisi Konselor

No	Skenario	Penjelasan
1	Login ke aplikasi	Responden diminta login dengan akun yang sudah terdaftar
2	Melihat daftar permintaan konseling	Responden diminta menampilkan permintaan konseling
4	Melihat detail profil pasien	Responden diminta menampilkan halaman detail profil pasien
5	Menerima permintaan konseling	Responden diminta menerima permintaan konseling
6	Menolak permintaan konseling	Responden diminta menolak permintaan konseling
7	Melakukan konseling online dengan pasien	Responden diminta melakukan konseling secara online dengan

		pasien
8	Melihat daftar appointment dengan pasien	Responden diminta membuka halaman daftar appointment

6.2.2 Analisis Hasil Pengujian

DAFTAR REFERENSI

Apple. (n.d.-a). MapKit. Retrieved October 6, 2019, from https://developer.apple.com/documentation/mapkit

Apple. (n.d.-b). Swift The powerful programming language that is also easy to learn. Retrieved October 5, 2019, from https://developer.apple.com/swift/

Apple. (2018). Managing Your App's Life Cycle. Retrieved August 24, 2019, from https://developer.apple.com/documentation/uikit/app_and_environment/managing_your_app_s_life_cycle

Arjadi, R., Nauta, M. H., & Bockting, C. L. H. (2018). Acceptability of internet-based interventions for depression in Indonesia. *Internet Interventions*, *13*(April), 8–15. https://doi.org/10.1016/j.invent.2018.04.004

Arjadi, R., Nauta, M. H., Scholte, W. F., Hollon, S. D., Chowdhary, N., Suryani, A. O., & Bockting, C. L. H. (2016). Guided Act and Feel Indonesia (GAF-ID) - Internet-based behavioral activation intervention for depression in Indonesia: Study protocol for a randomized controlled trial. *Trials*, *17*(1), 1–10. https://doi.org/10.1186/s13063-016-1577-9

Avram, A. (2013). iOS vs. Android Development. Retrieved October 5, 2019, from InfoQ website: http://www.infoq.com/news/2013/08/ios-vs-android-development

Bajpai, V., & Gorthi, R. P. (2012). On non-functional requirements: A survey. *2012 IEEE Students' Conference on Electrical, Electronics and Computer Science: Innovation for Humanity, SCEECS 2012*, (April). https://doi.org/10.1109/SCEECS.2012.6184810

Cao, J. (2015). *The Guide to Usability Testing*.

Chen, M., & Jiang, S. (2019). Analysis and research on mental health of college students based on cognitive computing. *Cognitive Systems Research*, *56*, 151–158. https://doi.org/10.1016/j.cogsys.2019.03.003

Erna Kumalasari Nurnawati, J. M. (2014). Aplikasi mobile berbasis lokasi untuk penyedia lokasi layanan kesehatan di Yogyakarta. *Prosiding Seminar Nasional Aplikasi Sains & Teknologi (SNAST)*, 293–300.

Faisandier, A. (2012). *SYSTEMS OPPORTUNITIES AND REQUIREMENTS*.

Farouqi, M. I., Aknuranda, I., & Herlambang, A. D. (2018). Evaluasi Usability pada Aplikasi Go-Jek Dengan Menggunakan Metode Pengujian Usability. *Jurnal Pengembangan Teknologi Informasi Dan Ilmu Komputer*, *2*(10), 3150–3156. Retrieved from http://j-ptiik.ub.ac.id/index.php/j-ptiik/article/download/2396/947/

Hidalgo, E. S. (2019). Adapting the scrum framework for agile project management in science: case study of a distributed research initiative. *Heliyon*, *5*(3), e01447. https://doi.org/10.1016/j.heliyon.2019.e01447

Islam, R., & Mazumder, T. (2010). Mobile application and its global impact. *International Journal of Engineering & …*, (06), 72–78. Retrieved from http://ijens.org/107506-0909 IJET-IJENS.pdf

Junglas, B. I. A., & Watson, R. T. (2008). *LOCATION-BASED SERVICES*. *51*(3), 65–70.

Larrea, M. (2017). Black-Box Testing Technique for Information Visualization. Sequencing Constraints with Low-Level Interactions. *Journal of Computer Science and Technology*, *17*(1), 37–48.

Mahoney, J., Le Moignan, E., Long, K., Wilson, M., Barnett, J., Vines, J., & Lawson, S. (2019). Feeling alone among 317 million others: Disclosures of loneliness on Twitter. *Computers in Human Behavior*, *98*(February), 20–30. https://doi.org/10.1016/j.chb.2019.03.024

Mevada, D. (2018). The Benefits of Having Firebase for Mobile App Development. Retrieved September 6, 2019, from https://www.mindinventory.com/blog/benefits-of-firebase-in-mobile-app-development/

Microsoft. (2017). The Model-View-ViewModel Pattern. Retrieved October 5, 2019, from https://docs.microsoft.com/en-us/xamarin/xamarin-forms/enterprise-application-patterns/mvvm

Nielson, J., & Landauer, J. (1993). A mathematical model of finding the usability problem. Proceedings of the CHI 93 proceedings of the Interact conference on human factors in computing systems. *Espacio de Trabajo Matemático. Quinto Simposio Internacional ET*, 206–213. https://doi.org/10.1145/169059.169166

Nugroho, S. C., Nurhayati, O. D., & Widianto, E. D. (2017). Aplikasi Pencarian Rute Perguruan Tinggi Berbasis Android Menggunakan Location Based Service (LBS) di Kota Semarang. *Jurnal Teknologi Dan Sistem Komputer*, *3*(2), 311. https://doi.org/10.14710/jtsiskom.3.2.2015.311-319

Oinas-Kukkonen, H., & Kurkela, V. (2003). Developing successful mobile applications. *Proceedings of the IASTED International Conference on Computer Science and Technology*, (January 2003), 50–54.

Rahman. (2015). Android Kuasai Asia Tenggara, di Indonesia Paling Juara.

Rather, M. A., & Bhatnagar, V. (2016). *A comprative study of sdlc model*. (October 2015).

Sauro, J., & Zarolia, P. (2017). SUPR-Qm: A Questionnaire to Measure the Mobile App User Experience. *Journal of Usability Studies*, *13*(1), 17–37.

Shivaram, A. M., & Handigund, S. M. (2015). An ameliorated methodology for the abstraction of object oriented features from software requirements specification. *Procedia Computer Science*, *62*(Scse), 274–281. https://doi.org/10.1016/j.procs.2015.08.450

Tristiana, R. D., Yusuf, A., Fitryasari, R., Wahyuni, S. D., & Nihayati, H. E. (2018). Perceived barriers on mental health services by the family of patients with mental illness. *International Journal of Nursing Sciences*, *5*(1), 63–67. https://doi.org/10.1016/j.ijnss.2017.12.003

WHO. (2012). *Suicide mortality rate (per 100,000 population)*.