KERJA PRAKTIK - EF234603

Pengembangan Aplikasi Mobile SWiDS

Klinik Utama Eagles HEAD *Medical Centre*
Jl. Seruni No.38, Ketabang, Kec. Genteng, Kota Surabaya, Jawa Timur 60275
Periode: 17 Juli 2024 - 30 November 2024

Oleh:

Thoriq Afif Habibi 5025211154

Pembimbing Departemen
Bagus Jati Santoso, S.Kom., Ph.D.
Pembimbing Lapangan
Dr. Shoffi Izza Sabilla, S.Kom.

DEPARTEMEN TEKNIK INFORMATIKA
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember
Surabaya 2024

KERJA PRAKTIK – EF234603

Pengembangan Aplikasi Mobile SWiDS

Klinik Utama Eagles HEAD *Medical Centre*
Jl. Seruni No.38, Ketabang, Kec. Genteng, Kota Surabaya, Jawa Timur 60275
Periode: 17 Juli - 30 November 2024

Oleh:

Thoriq Afif Habibi 5025211154

Pembimbing Departemen
Bagus Jati Santoso, S.Kom., Ph.D.

Pembimbing Lapangan
Dr. Shoffi Izza Sabilla, S.Kom.

DEPARTEMEN TEKNIK INFORMATIKA
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember
Surabaya 2024

[Halaman ini sengaja dikosongkan]

DAFTAR ISI

[Halaman ini sengaja dikosongkan]

DAFTAR GAMBAR

[Halaman ini sengaja dikosongkan]

DAFTAR TABEL

[Halaman ini sengaja dikosongkan]

LEMBAR PENGESAHAN
KERJA PRAKTIK

Pengembangan Aplikasi *Mobile* SWiDS

Oleh:

Thoriq Afif Habibi 5025211154

Disetujui oleh Pembimbing Kerja Praktik:

1. Bagus Jati Santoso, S.Kom., Ph.D.
   NIP. 198611252018031001

(Pembimbing Departemen)

2. Dr. Shoffi Izza Sabilla, S.Kom.
   NIP. 2022199412054

(Pembimbing Lapangan)

[Halaman ini sengaja dikosongkan]

Pengembangan Aplikasi *Mobile* SWiDS

Nama Mahasiswa	: Thoriq Afif Habibi
NRP	: 5025211154
Departemen	: Teknik Informatika FTEIC-ITS
Pembimbing Departemen	: Bagus Jati Santoso, S.Kom., Ph.D.
Pembimbing Lapangan	: Dr. Shoffi Izza Sabilla, S.Kom.

ABSTRAK

SWiDS merupakan aplikasi mobile yang dikembangkan untuk membantu tenaga kesehatan (nakes) dan dokter spesialis dalam mengelola data rekam medis pasien. Nakes dapat mengisi data pasien dan melakukan rekaman dengan stetoskop digital yang terintegrasi secara real-time. Di sisi lain, dokter spesialis dapat menerima data hasil rekaman dan memberikan diagnosis. Seluruh data yang berkaitan disimpan dalam Firebase storage dan firestore sehingga dapat dengan mudah diakses oleh pengguna.

Proyek ini dibangun dengan menggunakan framework Flutter dan arsitektur Model-View-Controller (MVC) untuk menangani bagian frontend dengan Firebase sebagai backend utama. Selain itu, dikembangkan pula service backend dengan Python dan SDK Firebase Admin untuk menangani fitur lainnya, yaitu pengonversi data hex menjadi file audio serta pengatur status rekaman berdasarkan ketersediaan dokter. Fitur utama aplikasi meliputi perekaman suara stetoskop, pengelolaan riwayat medis, manajemen daftar pasien, koneksi antara nakes dan dokter spesialis, serta integrasi dengan perangkat stetoskop.

Kata Kunci : Mobile, Flutter, Firebase, Stetoskop Digital, Kesehatan

[Halaman ini sengaja dikosongkan]

KATA PENGANTAR

Puji syukur penulis panjatkan kepada Allah SWT atas penyertaan dan karunia-Nya sehingga penulis dapat menyelesaikan salah satu kewajiban penulis sebagai mahasiswa Departemen Teknik Informatika ITS yaitu Kerja Praktik yang berjudul: Pengembangan Aplikasi *Mobile* SWiDS.

Penulis menyadari bahwa masih banyak kekurangan baik dalam melaksanakan kerja praktik maupun penyusunan buku laporan kerja praktik ini. Namun penulis berharap buku laporan ini dapat menambah wawasan pembaca dan dapat menjadi sumber referensi.

Melalui buku laporan ini penulis juga ingin menyampaikan rasa terima kasih kepada orang-orang yang telah membantu menyusun laporan kerja praktik baik secara langsung maupun tidak langsung antara lain:

1. Kedua orang tua penulis.
2. Bagus Jati Santoso, S.Kom., Ph.D. selaku dosen pembimbing kerja praktik sekaligus koordinator kerja praktik.
3. Dr. Shoffi Izza Sabilla, S.Kom. selaku pembimbing lapangan selama kerja praktik berlangsung.
4. Teman-teman penulis yang senantiasa memberikan semangat ketika penulis melaksanakan KP.

Surabaya, 5 Desember 2024
Thoriq Afif Habibi

[Halaman ini sengaja dikosongkan]

BAB I
PENDAHULUAN

1.1. Latar Belakang

Pada dunia kesehatan, sistem IT merupakan salah satu penunjang untuk membantu dokter dan tenaga kesehatan dalam memeriksa pasien. Dengan sistem IT yang baik, data pasien beserta rekam medisnya dapat dengan mudah diakses dan dikelola. Aplikasi untuk mengakses data rekam medis dapat berbentuk aplikasi web, aplikasi desktop, maupun aplikasi *mobile*. Bagi dokter dan tenaga kesahatan, platform *mobile* lebih disukai karena ponsel yang sering dibawa dan dapat diakses dengan lebih mudah dibandingkan desktop maupun laptop.

Salah satu permasalahan bagi nakes adalah kurangnya kompetensi dalam melakukan diagnosis penyakit. Nakes yang melakukan pemeriksaan ke pelosok, sering kali bukanlah dokter dengan spesialis penyakit dalam. Dengan begitu, diagnosis penyakit hasil pengecekan dengan stetoskop menjadi kurang mendalam. Dokter spesialis dengan keahlian yang sesuai biasanya berada di kota besar. Di lain sisi, masyarakat di pelosok juga membutuhkan akses kesehatan yang sama.

Oleh karena itu, aplikasi SWiDS dikembangkan sebagai solusi untuk membantu dokter dan tenaga kesehatan dalam mengelola data rekam medis. Aplikasi ini terintegrasi dengan stetoskop digital nirkabel yang dapat merekam jantung, paru-paru, maupun *abdomen*. Dengan perekaman digital, hasil rekaman dapat disimpan dan ditransmisikan sehingga dokter dapat melakukan diagnosis pasien dari jarak jauh. Aplikasi ini menghubungkan berbagai pihak, termasuk tenaga kesehatan, dokter, pasien, dan institusi kesehatan, untuk meningkatkan efektivitas dan efisiensi pelayanan kesehatan. Dengan fitur seperti perekaman suara stetoskop, pengelolaan riwayat

kesehatan, serta integrasi dengan perangkat medis lainnya, SWiDS bertujuan untuk meningkatkan kualitas diagnosis dan perawatan pasien secara menyeluruh.

1.2. Tujuan

Tujuan kerja praktik ini adalah menyelesaikan kewajiban nilai kerja praktik sebesar 4 sks dan membantu Klinik Utama Eagles HEAD *Medical Centre* untuk mengembangkan aplikasi SWiDS sesuai permintaan dari pada *stakeholder* aplikasi.

1.3. Manfaat

Manfaat aplikasi SWiDS adalah mempermudah proses pemeriksaan jantung, paru-paru, dan *abdomen* bagi para tenaga kesehatan. Dengan aplikasi ini, tenaga kesehatan dapat mengelola pasien, memasukkan keluhan pasien, serta mengatur perekaman stetoskop dengan sederhana. Aplikasi ini memiliki rekaman dari stetoskop yang terintegrasi, sehingga dokter dapat membantu tenaga kesehatan dalam memberikan diagnosis kepada pasien.

1.4. Rumusan Masalah

Rumusan masalah dari kerja praktik ini adalah sebagai berikut:

1. Bagaimana cara merancang aplikasi SWiDS sisi dokter yang mudah digunakan sesuai dengan kebutuhan dokter di lapangan?
2. Bagaimana mengintegrasikan stetoskop dengan aplikasi SWiDS agar dapat merekam dan menyimpan data medis secara akurat?

1.5. Lokasi dan Waktu Kerja Praktik

Kerja praktik ini dilakukan secara *remote* dengan waktu yang *fleksibel*. Untuk rapat pada saat kerja praktik,

dilaksanakan secara *online* di Zoom dan *offline* di Fakultas Kedokteran dan Kesehatan.

Adapun kerja praktik dimulai pada tanggal 17 Juli 2024 hingga 30 November 2024.

1.6. Metodologi Kerja Praktik

Metodologi dalam pembuatan buku kerja praktik meliputi :

1.6.1. Perumusan Masalah

Perumusan masalah kami lakukan dengan mengikuti rapat rutin bersama tim dari Fakultas Kedokteran dan Kesehatan ITS, yang terdiri dari para dokter, tim *software*, dan tim *hardware*. Pada rapat-rapat tersebut, para dokter menyampaikan kebutuhannya terkait sistem aplikasi yang dapat mengelola data rekam medis dan terintegrasi dengan stetoskop. Selanjutnya, kita juga membahas mengenai alur aplikasi beserta detail fitur yang dibutuhkan. Bersama tim *software*, kami selanjutnya membuat desain antarmuka aplikasi dan desain *database* sebagai dasar aplikasi yang akan kami kembangkan. Setelah desain disetujui oleh tim, kami kemudian memulai tahap pengembangan aplikasi.

1.6.2. Studi Literatur

Setelah memahami bagaimana aplikasi akan bekerja, kami diberikan tinjauan mengenai teknologi, pustaka, dan kerangka kerja yang diperlukan dalam pengembangan aplikasi. Tinjauan tersebut meliputi Firebase, Flutter, beserta fitur-fitur di dalamnya. Kami juga mengeksplorasi berbagai pustaka dari Flutter yang membantu dalam pengembangan aplikasi. Pada Firebase, kami menggunakan fitur *Authentication*, Firestore *Database*, *Realtime Database*, dan *Storage*. Kami juga menggunakan GeoFire untuk mengimplementasikan fitur pencarian dokter berdasarkan lokasi terdekat.

1.6.3. Analisis dan Perancangan Sistem

Langkah selanjutnya adalah merancang sistem dengan desain arsitektur yang baik. Aplikasi kami menggunakan pendekatan arsitektur MVC (*Model-View-Controller*) pada *framework* Flutter. Flutter menangani semua bagian *frontend* (*mobile*) dan sebagian besar fungsi *backend*, memanfaatkan Firebase sebagai *backend* utama karena telah menyediakan SDK yang mempermudah integrasi. Layanan *backend* untuk mengubah data hex menjadi data audio serta pengubahan status rekaman berdasarkan ketersediaan dokter dikembangkan menggunakan Python dan Firebase Admin.

1.6.4. Implementasi Sistem

Implementasi merupakan realisasi dari tahap perancangan. Pada tahap ini, kami melakukan pembuatan aplikasi sesuai dengan fitur-fitur dan alur yang diinginkan. Setelah tahap *development* selesai, kami melakukan *deployment* terhadap *service* yang digunakan.

1.6.5. Pengujian dan Evaluasi

Setelah aplikasi sudah jadi, dilakukan evaluasi pengujian apakah aplikasi sesuai dengan harapan *client*. Jika masih belum sesuai atau menambah fitur, maka akan dilakukan penambahan atau perbaikan fitur.

1.6.6. Kesimpulan dan Saran

Pengujian yang dilakukan ini telah memenuhi syarat yang diinginkan, dan berjalan dengan baik dan lancar.

1.7. Sistematika Laporan

1.7.1. Bab I Pendahuluan

Bab ini berisi latar belakang, tujuan, manfaat, rumusan masalah, lokasi dan waktu kerja praktik, metodologi, dan sistematika laporan.

1.7.2. Bab II Profil Perusahaan

Bab ini berisi gambaran umum Klinik Utama Eagles HEAD *Medical Centre* mulai dari profil, lokasi klinik.

1.7.3. Bab III Tinjauan Pustaka

Bab ini berisi dasar teori dari teknologi yang digunakan dalam menyelesaikan proyek kerja praktik.

1.7.4. Bab IV Infrastruktur Sistem

Bab ini berisi mengenai tahap analisis sistem aplikasi dalam menyelesaikan proyek kerja praktik.

1.7.5. Bab V Implementasi Sistem

Bab ini berisi uraian tahap - tahap yang dilakukan untuk proses pengembangan aplikasi.

1.7.6. Bab VI Pengujian dan Evaluasi

Bab ini berisi hasil uji coba dan evaluasi dari aplikasi yang telah dikembangkan selama pelaksanaan kerja praktik.

1.7.7. Bab VII Kesimpulan dan Saran

Bab ini berisi kesimpulan dan saran yang didapat dari proses pelaksanaan kerja praktik.

[Halaman ini sengaja dikosongkan]

BAB II
PROFIL PERUSAHAAN

2.1. Profil Klinik Utama Eagles HEAD *Medical Centre*

Klinik Utama Eagles HEAD *Medical Centre* di Surabaya adalah fasilitas kesehatan yang menitikberatkan pada pelayanan berorientasi kekeluargaan dan persahabatan dengan pasien. Klinik ini dilengkapi dengan fasilitas laboratorium klinik, radiologi, jantung, dan elektromedis yang modern, memberikan hasil yang cepat dan berkualitas dengan harga terjangkau. Beroperasi mulai pukul 6 pagi hingga 21 malam, klinik ini didukung oleh tenaga medis dan paramedis berpengalaman serta peralatan medis berkualitas, yang terletak strategis di pusat kota untuk kemudahan akses pasien.

2.2. Lokasi

Jl. Seruni No.38, Ketabang, Kec. Genteng, Kota Surabaya, Jawa Timur 60272

[Halaman ini sengaja dikosongkan]

BAB III
TINJAUAN PUSTAKA

3.1. Figma

Figma adalah alat desain antarmuka pengguna berbasis *cloud* yang populer di kalangan pengembang dan desainer UI/UX. Figma memungkinkan kolaborasi secara *realtime*, yang memudahkan tim untuk merancang dan memodifikasi desain aplikasi secara bersama-sama. Alat ini mendukung pembuatan prototipe, *wireframe*, dan desain antarmuka yang interaktif, serta mempermudah pengembang dalam menerapkan desain ke dalam kode aplikasi (Figma, 2024).

3.2. Pemrograman *Mobile*

Pemrograman *mobile* merupakan proses pengembangan aplikasi yang dapat dijalankan di perangkat *mobile* seperti *smartphone* maupun tablet. Aplikasi yang dikembangkan biasanya ditargetkan untuk sistem operasi tertentu, seperti Android dan iOS. Pengembangan aplikasi Android biasanya menggunakan bahasa Java atau Kotlin, sedangkan iOS menggunakan Swift atau Objective-c. Bahasa lain seperti JavaScript, Dart, dan Python, juga dapat digunakan untuk mengembangkan aplikasi lintas platform dengan kerangka kerja React Native, Flutter, dan Kivy. Setiap bahasa dan kerangka kerja memiliki kelebihan serta kekurangan. Pemilihan teknologi bergantung pada preferensi pengembang dan kebutuhan aplikasi. Dengan pemahaman yang baik tentang *mobile* programming, programmer dapat menciptakan aplikasi *mobile* yang inovatif dan efisien (MDN, 2024).

3.3. Flutter

Flutter merupakan salah satu kerangka kerja yang cukup populer untuk mengembangkan aplikasi lintas platform, baik *mobile*, desktop, maupun web. Kerangka kerja yang dikembangkan oleh Google ini menggunakan bahasa pemrograman Dart. Dengan *widget* yang dapat disesuaikan, pembuatan antarmuka aplikasi yang responsif dan dinamis lebih mudah dilakukan. Flutter menawarkan kinerja yang tinggi dengan kemampuan *rendering native* dan akses langsung ke API platform, memungkinkan pengembang untuk menciptakan aplikasi yang lancar dan interaktif (Flutter, 2024).

3.4. Firebase

Firebase adalah platform pengembangan aplikasi berbasis *cloud* yang disediakan oleh Google. Firebase menyediakan berbagai layanan seperti autentikasi pengguna, *database real-time*, analitik, penyimpanan *file*, dan layanan notifikasi *push* yang memudahkan pengembang dalam membangun dan mengelola aplikasi. Firebase mendukung integrasi dengan berbagai platform seperti Android, iOS, dan web, menjadikannya solusi yang fleksibel dan *scalable* untuk aplikasi modern.

Firebase menyediakan SDK yang lengkap dan terintegrasi dengan Flutter, yang memungkinkan pengembang untuk memanfaatkan berbagai layanan Firebase secara langsung dalam aplikasi Flutter. SDK ini mencakup autentikasi pengguna, integrasi dengan Firestore dan *Realtime Database*, penyimpanan *file* menggunakan Firebase *Storage*, serta fitur analitik dan notifikasi *push*. Integrasi dengan Flutter memastikan bahwa layanan-layanan tersebut dapat digunakan dengan efisien dan memberikan performa optimal pada aplikasi lintas platform (Firebase, 2024).

3.5. WAV (*Waveform Audio File Format*)

Audio merupakan *file* yang sering kali dibutuhkan pada aplikasi *mobile* untuk memenuhi fitur tertentu. Untuk memenuhi kebutuhan aplikasi, *file* ini dapat disimpan secara lokal di perangkat atau disimpan pada penyimpanan *cloud*. Penyimpanan *file* audio juga dapat menggunakan berbagai macam format, seperti CAF, MP3, maupun WAV. Setiap format memiliki spesifikasi dan metode kompresinya masing-masing.

WAV merupakan format penyimpanan audio yang merupakan bagian dari spesifikasi RIFF (*Resource Interchange File Format*) Microsoft. Format penyimpanan ini tidak menerapkan kompresi apapun pada aliran bit. Selain itu, format ini juga menyimpan rekaman suara dengan *sampling rate* dan *bit rate* yang berbeda. Format WAV terdiri dari berkas *header* dengan ukuran 44 bit diikuti rangkaian data *chunk* (FileFormat, 2024).

3.6. Python

Python merupakan bahasa pemrograman tingkat tinggi dengan *syntax* yang sederhana dan mudah dipahami. Salah satu penggunaan Python adalah pada pengembangan layanan *backend,* baik berjalan sendiri maupun terintegrasi dengan Firebase. Integrasi dengan Firebase dapat mudah dilakukan karena SDK Firebase Admin secara resmi telah tersedia untuk bahasa Python. Selain itu, Python juga memiliki fitur asinkronus dengan *asyncio* untuk mengembangkan layanan yang cepat (Kramer, 2024).

3.7. Systemd

Systemd adalah *init system* dan *service manager* untuk sistem operasi berbasis Linux. Alat ini dirancang untuk meningkatkan performa *booting* sistem, manajemen layanan, dan monitoring layanan. Systemd juga memiliki

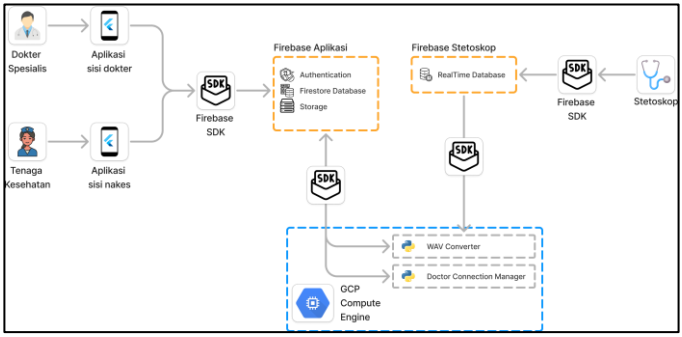
kemampuan *failure recovery*. Layanan yang mengalami *crash* dapat secara otomatis dijalankan kembali oleh systemd. Untuk monitoring layanan, *log* aktivitas dapat diakses melalui journalctl. Oleh karenanya, systemd cukup baik untuk digunakan pada *deployment* layanan *backend* (Systemd, 2024).

[Halaman ini sengaja dikosongkan]

BAB IV
INFRASTRUKTUR SISTEM

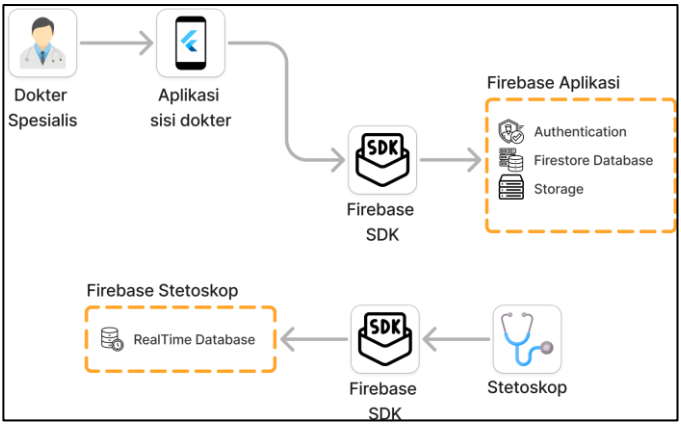
4.1. Desain Sistem

Desain arsitektur pada aplikasi SWiDS ini terbagi menjadi beberapa komponen utama, sebagaimana dijelaskan berikut.



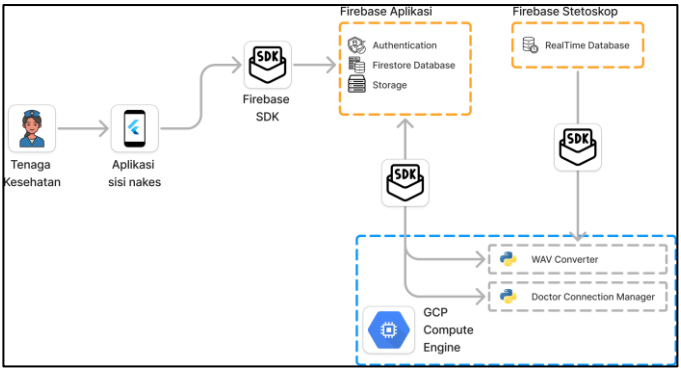
Gambar 4.1 Desain Arsitektur Aplikasi SWiDS

Aplikasi SWiDS memiliki 2 bagian yang masing-masing akan digunakan oleh nakes dan dokter spesialis. Kerangka kerja yang digunakan untuk mengembangkan antarmuka pengguna adalah Flutter. Aplikasi Flutter ini terhubung dengan Firebase melalui SDK yang tersedia untuk memenuhi berbagai kebutuhan *backend*, seperti fitur autentikasi, penyimpanan audio, dan *database*. Untuk mendukung aplikasi, terdapat *microservice* WAV *Converter* dan *Doctor Connection Manager* yang dikembangkan menggunakan Python. Hasil rekaman nantinya akan dikirim oleh stetoskop berbasis IoT ke Firebase dalam bentuk hex. Namun, perangkat IoT ini berada di luar cakupan KP.



Gambar 4.2 Komponen Arsitektur Diluar Pengerjaan KP

Aplikasi SWiDS merupakan sistem yang cukup besar dan kompleks sehingga tidak semua tercakup pada KP ini. Bagian sistem yang tidak tercakup adalah aplikasi bagian nakes dan stetoskop IoT seperti terlihat pada gambar 4.2. Aplikasi bagian nakes dikembangkan dengan bahasa flutter yang memiliki fitur berupa menambahkan/ mengelola pasien, menghubungkan/mengaktifkan stetoskop, melihat *history*, dan mengelola profil. Fitur-fitur ini tentunya menggunakan Firebase untuk menyimpan, menambah, dan menghapus data. Di sisi lain, perangkat IoT stetoskop berfungsi untuk mengirimkan data bit secara *realtime* ke *database* Firebase.



Gambar 4.3 Komponen Arsitektur Pengerjaan KP

Gambar 4.3 menunjukkan cakupan pengerjaan KP, yaitu aplikasi bagian dokter spesialis dan berserta layanan penunjang aplikasi. Aplikasi dokter dikembangkan dengan Flutter, kerangka kerja pengembangan aplikasi lintas platform. Pemilihan flutter didasarkan pada kemudahannya dalam mengembangkan antarmuka yang interaktif dan responsif. Selain itu, komunitas dan dukungan SDK yang luas juga menjadi pertimbangan tim dalam memilih kerangka kerja ini. Flutter juga memiliki fitur *hot reload* yang dapat menampilkan perubahan aplikasi tanpa memulai ulang sehingga meningkatkan efisiensi pengembangan aplikasi.

Aplikasi ini terhubung dengan Firebase sebagai *backend*. Fitur Firebase yang digunakan antara lain *authentication*, firestore *database*, *realtime database*, dan *storage*. Firebase *authentication* digunakan untuk mengautentikasi dokter berdasarkan email dan password yang terdaftar. Selanjutnya, data-data yang berkaitan dengan aplikasi dokter, seperti profil, riwayat diagnosis, hingga data klinik, disimpan pada Firestore *database*. Sedangkan data hex hasil rekaman stetoskop disimpan pada *realtime database* sehingga memungkinkan data audio diproses dan disimpan secara *realtime*. Terakhir,

Firebase *storage* digunakan untuk menyimpan *file* WAV hasil rekaman.

Sistem aplikasi ini juga didukung oleh microservice yang di-*deploy* pada GCP *compute engine*. *Microservice* ini berguna untuk mengonversi data hex dari stetoskop menjadi *file* WAV. Layananan pengonversi WAV akan melakukan *stream* pada *realtime database*. Selanjutnya, proses konversi dan penyimpanan ke Firebase storage akan dilakukan jika data pada *realtime database* telah terkirim seluruhnya oleh stetoskop. *Microservice* juga berguna untuk mengatur koneksi dokter spesialis. Hal ini ditujukan agar nakes tidak harus menunggu diagnosis dokter yang tidak memberikan respon.

Dengan arsitektur ini, aplikasi SWiDS dapat menangani berbagai fitur dengan lancar, mulai dari *streaming* data medis, penambahan data, hingga proses koneksi nakes dengan dokter.

4.2. Fungsionalitas Aplikasi SWiDS

Aplikasi SWiDS dirancang untuk membantu tenaga medis dalam merekam dan mengatur data rekam stetoskop secara efektif. Hal ini dapat mendukung diagnosis medis berbasis suara. Secara garis besar, nakes melalui aplikasi dapat memulai perekaman dengan mengaktifkan stetoskop, memilih pasien, dan memasukkan keluhan. Selanjutnya, rekaman dapat dilakukan menggunakan stetoskop yang hasilnya akan dikirim ke Firebase *realtime database*. Hasil rekaman berupa data hex akan dikonversi ke dalam *file* WAV dan disimpan ke Firebase *storage* oleh *microservice*. Aplikasi kemudian dapat mengakses *file* WAV tersebut dan menjalankannya untuk didengar oleh nakes maupun dokter.

Hasil rekaman akan digunakan oleh dokter spesialis untuk melakukan diagnosis sehingga SWiDS

menyediakan fitur pada aplikasi sisi nakes untuk mencari dan memilih dokter spesialis terdekat. Fitur ini telah dilengkapi dengan *filter* sesuai spesialisasi dokter, spesialis jantung dan pembuluh darah (sp.jp) dapat mendiagnosis rekaman jantung, spesialis paru (sp.p) dapat mendiagnosis rekaman paru-paru, sedangkan spesialis penyakit dalam (sp.pd) dapat mendiagnosis seluruh jenis rekaman. Setelah memilih dokter, aplikasi sisi nakes akan menunggu dokter spesialis menerima permintaan diagnosis.

Aplikasi sisi dokter terdiri dari 3 menu utama, yaitu *home*, *history*, dan profil. Pada menu *home*, dokter dapat melihat daftar permintaan diagnosis yang diterima dari nakes. Selanjutnya, dokter dapat menerima permintaan dan memulai sesi konsultasi dengan menekan "mulai" pada salah satu permintaan. Dengan memulai sesi konsultasi, dokter akan mendapatkan akses ke data pasien, keluhan, dan hasil rekamannya. Jika rekaman kurang jelas, terdapat fitur chat yang dapat digunakan untuk meminta nakes melakukan rekaman ulang. Selanjutnya, dokter dapat mengisi diagnosis dan resep obat jika rekaman telah jelas.

Pada menu *home*, terdapat pula tombol untuk mengaktifkan/menonaktifkan status ketersediaan konsultasi. Dokter dengan status ketersediaan nonaktif tidak akan bisa menerima permintaan konsultasi dari nakes, nama dokter tidak akan muncul pada fitur pencarian dokter di aplikasi nakes. Jika permintaan konsultasi telah dilakukan sebelum status nonaktif, permintaan tersebut akan dibatalkan oleh *microservice*. *Microservice* ini juga dapat membatalkan permintaan konsultasi jika dokter tidak menerima setelah 30 menit. Hal ini ditujukan agar nakes tidak perlu menunggu 1 dokter terlalu lama.

Sesi konsultasi yang telah selesai akan masuk ke riwayat konsultasi pada menu *history*. Pada menu tersebut,

dokter dapat meninjau kembali data rekam medis pasien yang pernah ditangani. Untuk memudahkan pencarian konsultasi, terdapat pula fitur pencarian yang mem-*filter* data konsultasi berdasarkan kata kunci yang diberikan.

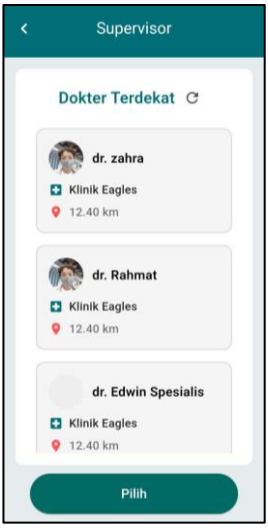
[Halaman ini sengaja dikosongkan]

BAB V
IMPLEMENTASI SISTEM

Bab ini membahas implementasi sistem yang telah kami buat, mencakup pengembangan aplikasi SWiDS menggunakan Flutter dan Firebase beserta pengembangan *microservice* wav *converter* dan *doctor connection manager*. Implementasi fitur aplikasi SWiDS yang tercakup pada KP ini dibagi menjadi fitur pemilihan dokter spesialis, fitur diagnosis, fitur chat dokter dengan nakes, fitur riwayat konsultasi, layanan pengatur koneksi dokter, dan layanan wav *converter*. Selain itu, dijelaskan pula implementasi *deployment* layanan *backend* pada *server*.

5.1 Fitur Pemilihan Dokter Spesialis

Setelah melakukan rekaman, nakes dapat memilih dokter spesialis terdekat untuk meminta diagnosis. Fitur pemilihan dokter ini terlebih dahulu melakukan pencarian klinik/rumah sakit berdasarkan lokasinya. Selanjutnya, daftar dokter spesialis yang terafiliasi dengan klinik tersebut akan ditampilkan pada nakes. Tampilan pemilihan dokter spesialis dapat dilihat pada Gambar 1.



Gambar 1. Tampilan Fitur Pencarian Dokter Terdekat

Pencarian berbasis lokasi ini dilakukan dengan meng-*encoding* nilai koordinat posisi klinik ke dalam bentuk *geohash*. Selanjutnya, nilai *geohash* tersebut digunakanan untuk mengkalkulasi jarak dengan lebih mudah dan cepat. *Geohash* diimplementasikan menggunakan Firebase *geo query* dan *library* geofirefultter. Pertama, pada Firebase dibuat *field "location"* pada klinik berupa map dengan 2 nilai, yaitu 'geohash' dan 'geopoint' seperti pada Gambar 2. Nilai keduanya disesuaikan dengan koordinat asli klinik.

```
location
   geohash: "w2081040"
   geopoint: [0° N, 102° E]
```

Gambar 2 Struktur Penyimpanan Lokasi Klinik pada Firebase

Selanjutnya, fitur ini juga perlu koordinat langsung nakes yang sedang mencari dokter spesialis. Untuk mendapat data lokasi secara langsung, aplikasi flutter perlu mengakses GPS yang ada pada gawai. Pengaksesan GPS menggunakan *pacakage* geolocator dengan terlebih dahulu meminta izin dari pengguna. Posisi kemudian didapat dengan menggunakan fungsi "getCurrentPosition" seperti pada Kode Sumber 1.

```
Future<bool> getPermission() async {
  LocationPermission permission = await
Geolocator.checkPermission();

  // Request permissions if not granted
  if (permission == LocationPermission.denied ||
      permission == LocationPermission.deniedForever) {
    permission = await Geolocator.requestPermission();
  }

  // Return whether permission is granted
  return permission == LocationPermission.always ||
      permission == LocationPermission.whileInUse;
}

Future<void> getPosition() async {
  if (!permission) {
    if (await getPermission() == false) {
      return;
    }
    permission = true;
  }

  state = await Geolocator.getCurrentPosition(
      desiredAccuracy: LocationAccuracy.medium);
}
```

Kode Sumber 1. Implementasi Akses Lokasi Pengguna pada Flutter

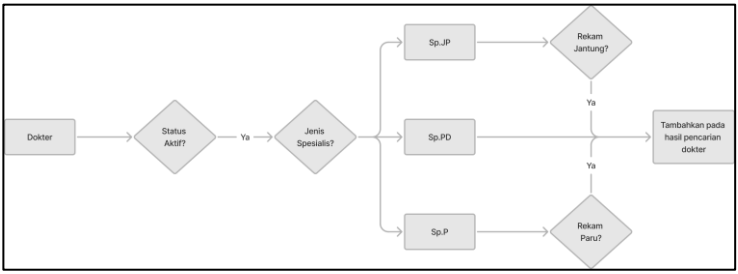
Setelah mendapatkan lokasi pengguna, dilakukan pencarian klinik terdekat menggunakan fungsi

"fetchWithin()" seperti pada Kode Sumber 2. Fungsi tersebut mengembalikan kumpulan "DocumentSnapshot" klinik terkait. Pada Firebase, dokumen klinik menyimpan *reference* ke dokter pada suatu *array*. Kumpulan dokter tersebut menjadi *output* dari pencarian. Namun, kumpulan dokter tersebut masih perlu disaring berdasarkan tipe rekaman dan status ketersediaan dengan aturan seperti pada Gambar 3.

```
final GeoCollectionReference<Map<String, dynamic>> geoCollection
=
    GeoCollectionReference<Map<String,
dynamic>>(_clinicReference);
GeoPoint geopointFrom(Map<String, dynamic> data) =>
    (data['location'] as Map<String, dynamic>)['geopoint'] as
GeoPoint;

// get clinic in radius
final List<DocumentSnapshot<Map<String, dynamic>>> clinicsDocs =
    await geoCollection.fetchWithin(
  center: GeoFirePoint(currLoc),
  radiusInKm: radius,
  field: 'location',
  geopointFrom: geopointFrom,
);
List<DocumentReference> doctorRefs = [];
List<int> doctorRefEndIndexes = [];
List<Clinic> clinics = clinicsDocs.map((doc) {
  final GeoPoint clinicPoint =
```

```
doc.data()!['location']['geopoint'];
  List<dynamic> clinicDoctorRefs = doc.data()!['doctors'];
  clinicDoctorRefs.forEach((ref) {
    doctorRefs.add(ref);
  });
  doctorRefEndIndexes.add(doctorRefs.length);

  print(doc.data()!['doctors'][0].get());
  double dist = Geolocator.distanceBetween(clinicPoint.latitude,
      clinicPoint.longitude, center.latitude, center.longitude);
  // List<UserData> userData =
  return Clinic.fromMap(doc.data() as Map<String, dynamic>,
doc.id,
      dist: dist);
}).toList();

// get doctors data
List<UserData> doctors = [];
List<Future<DocumentSnapshot>> futures =
    doctorRefs.map((ref) => ref.get()).toList();
List<DocumentSnapshot> doctorsSnapshots = await
Future.wait(futures);
// Process the documents once they're fetched
doctorsSnapshots.forEach((snapshot) {
  doctors.add(UserData.fromMap(
      snapshot.data() as Map<String, dynamic>, snapshot.id));
});
```

Kode Sumber 2. Implementasi Pencarian Klinik terdekat dari Lokasi Saat Ini



Gambar 3. Diagram Alur Penyaringan Dokter Spesialis

5.2 Fitur *Inbox*

Dokter spesialis dapat menerima permintaaan diagnosis dengan terlebih dahulu menekan *toggle* yang terdapat pada kanan atas kotak *inbox* di Gambar 4. Jika sudah aktif, *toggle* akan berwarna hijau seperti pada tampilan Gambar 5. Jika terdapat nakes yang memilih dokter terkait pada fitur 5.1, id dokter akan tercatat pada *database*. Dalam hal ini, aplikasi yang melakukan *stream* pada *database* akan menambahkan permintaan diagnosis tersebut ke kotak *inbox* seperti pada Gambar 6. Implementasi fitur ini terdapat pada

Gambar 4. Tampilan Inbox Dokter Spesialis dengan Status Nonaktif

Gambar 5. Tampilan Inbox Dokter Spesialis dengan Status Aktif

Gambar 6. Tampilan Inbox Dokter Spesialis dengan Permintaan Konsultasi

```
final userData = ref.watch(userProvider);

return Scaffold(
  body: Container(
    width: double.infinity,
    decoration: const BoxDecoration(
      gradient: LinearGradient(
        colors: [
          Color.fromRGBO(1, 109, 119, 1),
          Color.fromARGB(255, 131, 196, 190),
          Color.fromARGB(207, 1, 109, 119),
        ],
        begin: Alignment.topCenter,
        end: Alignment.bottomCenter,
      ),
    ),
    child: Stack(
      children: [
```

```
        const BackgroundImage(
          imageUrl: 'assets/image/bg-homescreen-1.svg',
          top: 0,
          right: 0,
          widthMultiplier: 0.7,
        ),
        const BackgroundImage(
          imageUrl: 'assets/image/bg-homescreen-2.svg',
          top: 10,
          right: 0,
          widthMultiplier: 0.3,
        ),
        Column(
          children: [
            InboxHeader(user: userData),
            Expanded(
              child: InboxContainer(
                user: userData,
              ), // Use the new widget
            ),
          ],
        ),
      ],
    ),
  ),
);
```

Kode Sumber 3. Implementasi Inbox Permintaan Konsultasi

```
class ReviewReportNotifier extends StateNotifier<List<Report>> {
  ReviewReportNotifier() : super([]) {
    _listenToReports();
  }

  StreamSubscription<QuerySnapshot>? _reportSubscription;

  void _listenToReports() async {
    final User? currentUser = FirebaseAuth.instance.currentUser;

    if (currentUser == null) {
      state = [];
```

```
      return;
    }

    final userId = currentUser.uid;
    print(userId);
    _reportSubscription = FirebaseFirestore.instance
        .collection('reports')
        .where('connectionStatus', isEqualTo: 'Menunggu')
        .where('supervisorId', isEqualTo: userId) //
        .snapshots()
        .listen((QuerySnapshot snapshot) {
      final reports = snapshot.docs.map((doc) {
        return Report.fromMap(doc.data() as Map<String,
dynamic>, doc.id);
      }).toList();

      // Filter the reports by userId
      final filteredReports =
          reports.where((report) => report.supervisorId ==
userId).toList();

      state = filteredReports;
    }, onError: (error) {
      state = [];
    });
  }

  @override
  void dispose() {
    _reportSubscription?.cancel();
    super.dispose();
  }
}
```

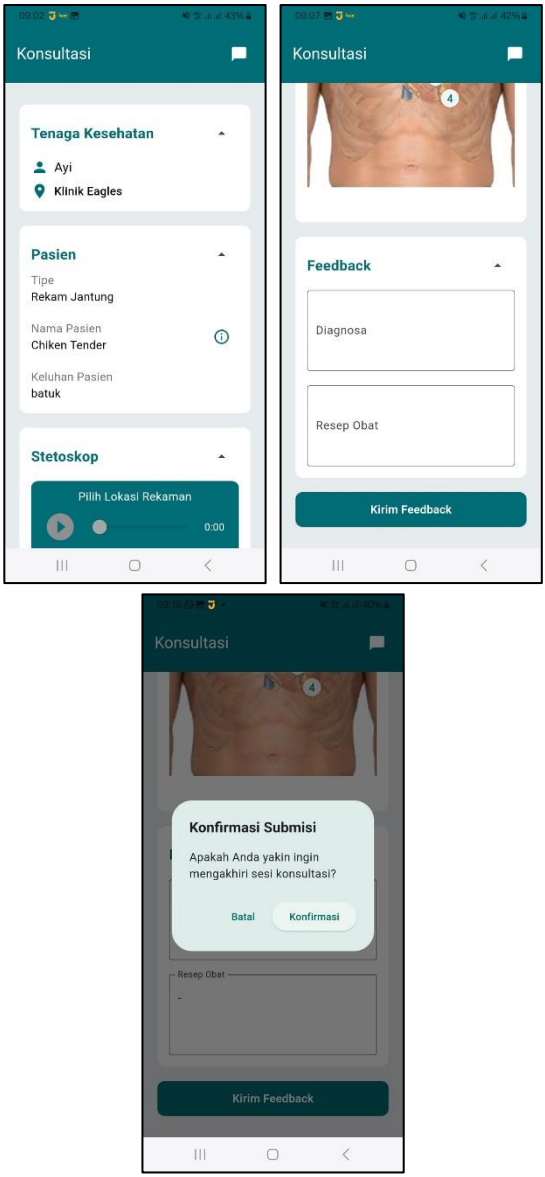
Kode Sumber 5.11. Implementasi Provider untuk Stream Data Konsultasi ke Firebase

5.3 Fitur Konsultasi

Untuk menerima permintaan konsultasi, dokter dapat menekan *card* pada *inbox*, lalu menekan “mulai”

pada pesan konfirmasi seperti di Gambar 7. Selanjutnya, sesi konsultasi akan dimulai dan aplikasi masuk ke halaman konsultasi seperti pada Gambar 8. Pada halaman tersebut, tersedia data rekaman, mulai dari nakes yang menangani, pasien, hingga audio rekaman. Form untuk mengisi diagnosis dan resep obat terdapat pada bagian bawah halaman. Terdapat pula tombol "Kirim *Feedback*" untuk mengirim diagnosis yang telah diisi. Implementasi fitur sesi konsultasi tertulis pada Kode Sumber 4 dan Kode Sumber 5.

Gambar 7. Tampilan Pesan Konfirmasi Mulai Sesi Konsultasi.



Gambar 8. Tampilan Halaman Konsultasi

```
Scaffold(
  appBar: ConsultationHeader(
    hasNewMessage: _hasNewMessage,
    animationController: _animationController,
    report: reportProgress,
    resetNewMessage: () {
      setState(() {
        _hasNewMessage = false;
      });
    },
  ),
  body: SingleChildScrollView(
    child: Container(
      width: double.infinity,
      padding: const EdgeInsets.symmetric(horizontal: 20,
vertical: 30),
      color: const Color(0xFFE6ECED),
      child: Form(
        key: _formKey,
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.start,
          children: [
            ReportNakesCard(),
            const SizedBox(height: 20),
            DetailReportDataBox(reportProgress),
            const SizedBox(height: 20),
            ConsultationAudioPlayBox(
              recordPath: reportProgress.recordPath,
              type: reportProgress.type,
              report: reportProgress,
            ),
            const SizedBox(height: 20),
            ConsultationInputBox(
              diagnosisController: _diagnosisController,
              prescriptionController: _prescriptionController,
            ),
            const SizedBox(height: 20),
            SubmitConsultationButton(
              diagnosis: _diagnosisController.text,
              prescription: _prescriptionController.text,
              onSubmit: () => _submitForm(reportProgress),
```

```
            ),
          ],
        ),
      ),
    ),
  ),
)
```

Kode Sumber 4. Implementasi Halaman Konsultasi

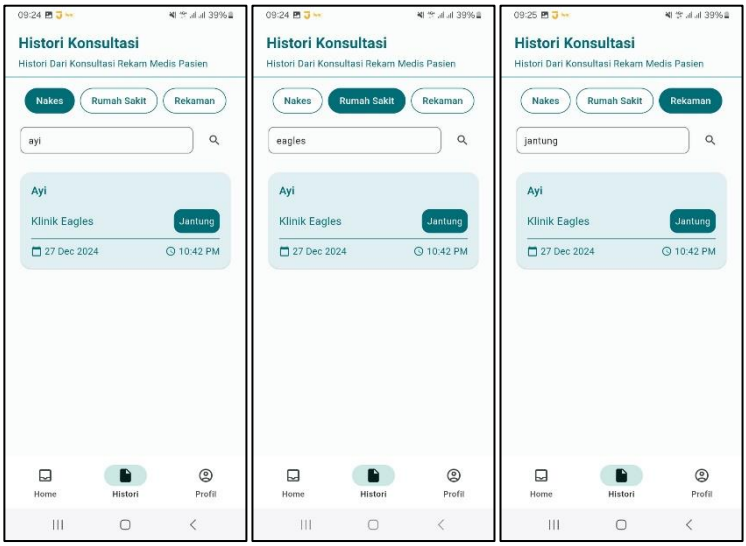
```
Future<void> _updateReportInFirestore(Report report) async {
  try {
    await FirebaseFirestore.instance
        .collection('reports')
        .doc(report.id)
        .update({
      'connectionStatus': 'Selesai',
      'diagnose': _diagnosisController.text,
      'medicine': _prescriptionController.text,
    });
    ScaffoldMessenger.of(context).showSnackBar(
      const SnackBar(content: Text('Data berhasil dikirim!')),
    );
    Navigator.popUntil(context, (route) => route.isFirst);
  } catch (e) {
    ScaffoldMessenger.of(context).showSnackBar(
      const SnackBar(
          content: Text('Gagal mengirim data. Silakan coba
lagi.')),
    );
  }
}
```

Kode Sumber 5. Implementasi Pengiriman Diagnosis ke Firebase

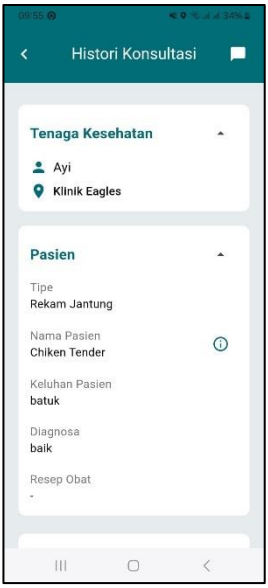
5.4 Fitur Riwayat Konsultasi

Sesi konsultasi yang telah selesai dapat ditinjau kembali dengan fitur ini. Fitur ini memperlihatkan daftar konsultasi yang dapat di-*filter* berdasarkan nama nakes, rumah sakit, maupun tipe rekaman ditampilkan pada

Gambar 9. Dengan menekan salah satu rekaman, aplikasi akan mengarahkan ke halaman detail konsultasi berisi data rekaman beserta diagnosisnya seperti pada Gambar 10. Implementasi daftar riwayat konsultasi terdapat pada Kode Sumber 6, sedangkan detail konsulatisi pada Kode Sumber 7.



Gambar 9. Tampilan Halaman Riwayat Konsultasi



Gambar 10. Tampilan Rangkuman Hasil Konsultasi

```
final reports = ref.watch(consultedReportProvider);

return Scaffold(
  appBar: const PreferredSize(
    preferredSize: Size.fromHeight(80.0),
    child: HistoryConsultationHeader(),
  ),
  body: Padding(
    padding: const EdgeInsets.symmetric(horizontal: 23.0,
vertical: 16),
    child: Column(
      crossAxisAlignment: CrossAxisAlignment.start,
      children: [
        HistoryFilter(
          selectedFilter: selectedFilter,
          onFilterChanged: (value) {
            setState(() {
              selectedFilter = value;
```

```
            });
          },
        ),
        const SizedBox(height: 16),
        // Search Box
        HistorySearch(
          selectedFilter: selectedFilter,
          onSearchChanged: (value) {
            setState(() {
              searchQuery = value;
            });
          },
        ),
        const SizedBox(height: 16),
        Expanded(
          child: Center(
            child: _buildContent(reports),
          ),
        ),
      ],
    ),
  ),
);
```

Kode Sumber 6. Implementasi Tampilan Daftar Riwayat Konsultasi

```
Scaffold(
  appBar: AppBar(
      toolbarHeight: 80,
      flexibleSpace: Container(
        color: const Color(0xFF016D77),
      ),
      title: const Text(
        "Histori Konsultasi",
        style: TextStyle(
          color: Colors.white,
          fontSize: 22,
          fontWeight: FontWeight.w400,
        ),
      ),
      leading: IconButton(
```

```
        icon: const Icon(Icons.chevron_left_rounded),
        onPressed: () => Navigator.pop(context),
        color: Colors.white,
        iconSize: 30,
      ),
      centerTitle: true,
      actions: [
        Container(
          margin: const EdgeInsets.only(right: 12, top: 10,
bottom: 10),
          decoration: BoxDecoration(
            borderRadius: BorderRadius.circular(8),
          ),
          child: IconButton(
            icon: const Icon(Icons.chat_bubble),
            color: Colors.white,
            iconSize: 24,
            onPressed: () {
              Navigator.push(
                context,
                MaterialPageRoute(
                  builder: (context) => ChatScreen(report:
widget.report),
                ),
              );
            },
          ),
        ),
      ]),
  body: SingleChildScrollView(
    child: Container(
      width: double.infinity,
      padding: const EdgeInsets.symmetric(horizontal: 20,
vertical: 30),
      color: const Color(0xFFE6ECED),
      child: Column(
        crossAxisAlignment: CrossAxisAlignment.start,
        children: [
          HistoryReportNakesCard(report: widget.report),
          const SizedBox(height: 20),
          HistoryConsultationReportBox(widget.report),
```

```
          const SizedBox(height: 25),
          DetailRecordBox(
            recordPath: widget.report.recordPath,
            type: widget.report.type,
          ),
        ],
      ),
    ),
  ),
);
```

Kode Sumber 7. Implementasi Tampilan Detail Konsultasi

5.5 Layanan Pengatur Koneksi Dokter

Seperti dijelaskan pada fitur 5.1, dokter dapat menonaktifkan konsultasi. Layanan ini berfungsi untuk menghapus daftar permintaan konsultasi kepada dokter yang nonaktif. Hal ini diperlukan agar nakes tidak perlu menunggu dokter yang sedang tidak aktif. Selain itu, terdapat pula kasus ketika dokter sedang tidak mengakses aplikasi, tetapi lupa menonaktifkan konsultasi. Dalam hal ini, layanan akan menerapkan *timeout* permintaan konsultasi jika telah melebihi 30 menit sehingga nakes dapat memillih dokter lainnya.

Layanan ini dikembangkan dengan Firebase SDK yang ada di Python. Pertama, perlu dilakukan konfigurasi Firebase ke layanan Python dengan kredensial yang didapat dari Firebase. Konfigurasi diimplementasikan pada dengan kredensial tersimpan pada *file* ".env". Selanjutnya, layanan ini melakukan *stream* pada data dokter dan me-*reset* dokter jika terjadi penonaktifan status konsultasi. Selain itu, *stream* kepada data rekaman juga dilakukan untuk melihat apakah terdapat kegiatan permintaan konsultasi ke dokter. Jika terdapat permintaan

konsultasi baru, layanan akan menjadwalkan *task* yang akan dilakukan 30 menit lagi. *Task* ini akan melihat kembali apakah permintaan konsultasi sudah diterima setelah 30 menit. Jika belum, maka *reset* dokter akan dilakukan. Implementasi konfigurasi Firebase terdapat pada Kode Sumber 8, *reset* dokter yang nonaktif pada Kode Sumber 9, dan *reset* ketika *timeout* pada Kode Sumber 10. Contoh *log* layanan ini dapat dilihat pada Gambar 11.

```
load_dotenv(override=True)
softwareFirebaseCertificate = os.getenv("FIREBASE_CONFIG2")
storageUrl = os.getenv("STORAGE_URL2")

cred = credentials.Certificate(softwareFirebaseCertificate)
app = initialize_app(
    cred,
    {
        "storageBucket": storageUrl
    },
    name="softwareApp"
)
```

Kode Sumber 8. Implementasi Konfigurasi Firebase pada Layanan Python

```
Future<void> _updateUserProfile() async {
    final user = FirebaseAuth.instance.currentUser;
    if (user != null) {
      String? imageUrl;

      if (_selectedImage != null) {
        setState(() {
          _isUploading = true;
        });

        final storageRef = FirebaseStorage.instance
            .ref()
```

```
            .child('profile_pictures')
            .child('${user.uid}.jpg');

        final uploadTask = storageRef.putFile(_selectedImage!);
        final snapshot = await uploadTask.whenComplete(() {});
        imageUrl = await snapshot.ref.getDownloadURL();

        setState(() {
          _isUploading = false;
        });
      }

      final userDoc =

FirebaseFirestore.instance.collection('users').doc(user.uid);
      await userDoc.update({
        'name': nameController.text,
        // 'email': emailController.text,
        'phoneNumber': phoneController.text,
        if (imageUrl != null) 'profilePicture': imageUrl,
      });
    }
  }
```

Kode Sumber 9. Implementasi Fitur Reset Permintaan Konsultasi Dokter yang Nonaktif

```
def on_waiting_report_timeout(col_snapshot, changes, read_time):
    for change in changes:
        if change.type.name == "MODIFIED":
            doc = change.document
            data = doc.to_dict()
            timeout = 30*60
            if data.get("supervisorId") != "" and
data.get("connectionStatus") == "Menunggu":
                print(f"{doc.id} connecting to supervisor")

asyncio.run(run_after_delay(unset_report_timeout, timeout,
doc.id))

    callback_done.set()
```

```
def unset_report_timeout(reportId: str):
    print("timeout")
    # get report
    doc_ref = db.collection("reports").document(reportId)
    doc = doc_ref.get()

    if doc.exists:
        data = doc.to_dict()
        if data.get("connectionStatus") == "Menunggu":
            print(f"Timeout: {doc.id} not accept the request")
            # update report
            doc_ref.update({
                "supervisorId": ""
            })

async def run_after_delay(task_function, delay, args):
    await asyncio.sleep(delay)  # Wait for the specified delay
    task_function(args)       # Run the provided task

report_query =
db.collection("reports").where(filter=FieldFilter("connectionSta
tus", "==", "Menunggu"))
report_query_watch =
report_query.on_snapshot(on_waiting_report_timeout)
```

Kode Sumber 10. Implementasi Reset Permintaan Konsultasi Dokter ketika Melebihi Timeout

```
fZ3OUktNTbYS9ThUk0rrA7Ib9As1 inactive, unset waited report
fZ3OUktNTbYS9ThUk0rrA7Ib9As1 inactive, unset waited report
timeout
Timeout: ycZpeuEmIeMBFtTVAioa not accept the request
```

Gambar 11. Contoh Log Layanan Pengatur Koneksi Dokter

5.6 Layanan WAV *Converter*

Layanan ini mengkonversi data hex dari stetoskop yang tersimpan di *database* menjadi *file* wav yang disimpan pada Firebase *Storage*. Implementasi menggunakan Python dan Firebase SDK seperti pada

layanan 5.5. Karena Firebase stetoskop berbeda dengan Firebase aplikasi, konfigurasi dilakukan untuk 2 Firebase dengan kode implementasi pada Kode Sumber 11.

Setelah konfigurasi Firebase berhasil, layanan akan melakukan *stream* pada *realtime database* tempat data rekaman disimpan. Jika data rekaman terkirim sepenuhnya, ditandai dengan "#end#", proses konversi akan dilakukan. Implementasi pengonversi terdapat pada Kode Sumber 12. Setelah dijalankan, contoh *log* dari layanan ini dapat dilihat pada Gambar 12.

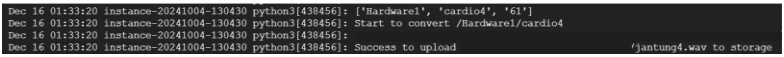
```
def initHardwareApp():
    load_dotenv(override=True)
    hardwareFirebaseCertificate = os.getenv("FIREBASE_CONFIG1")
    dbUrl = os.getenv("DB_URL1")

    cred = credentials.Certificate(hardwareFirebaseCertificate)
    app = initialize_app(
        cred,
        {
            "databaseURL": dbUrl,
        },
        name="HardwareApp"
    )

    return app

def initSoftwareApp():
    load_dotenv(override=True)
    softwareFirebaseCertificate = os.getenv("FIREBASE_CONFIG2")
    storageUrl = os.getenv("STORAGE_URL2")

    cred = credentials.Certificate(softwareFirebaseCertificate)
    app = initialize_app(
        cred,
        {
            "storageBucket": storageUrl
```

```
        },
        name="softwareApp"
    )

    return app
```

Kode Sumber 11. Implementasi Konfigurasi Firebase pada Layanan WAV Converter

```
async def __convertToWav(self, hex: str) -> bytes:
    result: bytearray = bytearray()

    # change each hex to byte
    for h in hex.split():
        b = int(h, 16)
        result.append(b)

    # change overall file size
    fileSize: int = len(result) - 8
    fileSizeHex: bytearray = bytearray(fileSize.to_bytes(4,
"little"))
    for i in range(4):
        result[i + 4] = fileSizeHex[i]

    # change data size
    dataSize: int = len(result) - 44
    dataSizeHex: bytearray = bytearray(dataSize.to_bytes(4,
"little"))
    for i in range(4):
        result[i + 40] = dataSizeHex[i]

    return bytes(result)
```

Kode Sumber 12. Implementasi WAV Converter



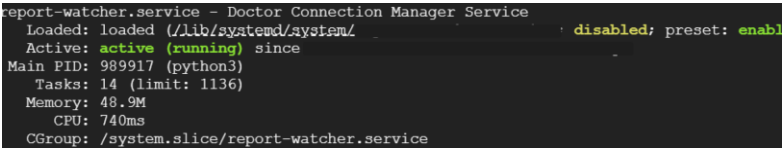
Gambar 12. Contoh Log Layanan Wav Converter

5.7 *Deployment* Layanan *Backend*

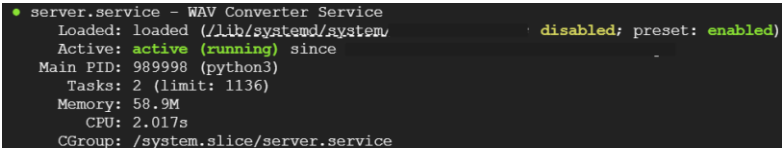
Layanan pada 5.5 dan 5.6 di-*deploy* pada *server* GCP *Compute Engine*. Systemd yang digunakan sebagai *service manager* akan menjalankan kode Python yang telah dikembangkan. Konfigurasi systemd mengikuti format pada Kode Sumber 13. Status layanan yang telah dideploy terdapat pada Gambar 13 dan Gambar 14.

```
[Unit]
Description=<Service Description>
After=

[Service]
Type=
User=
WorkingDirectory=<path to project>
ExecStart=<path to python env> <path to project's main script>
Restart=on-failure

[Install]
WantedBy=multi-user.target
```

Kode Sumber 13. Konfigurasi Systemd



Gambar 13. Status Layanan Pengatur Koneksi Dokter



Gambar 14. Status Layanan Wav Converter

[Halaman ini sengaja dikosongkan]

BAB VI
PENGUJIAN DAN EVALUASI

Bab ini menjelaskan tahap uji coba terhadap Aplikasi SWiDS sisi dokter besera layanan penunjangnya. Pengujian dilakukan untuk memastikan fungsionalitas dan kesesuaian hasil implementasi arsitektur dengan analisis dan perancangan arsitektur.

6.1. Tujuan Pengujian

Pengujian dilakukan terhadap Aplikasi SWiDS dan layanan Python guna menguji kemampuan arsitektur dalam melayani permintaan sistem aplikasi.

6.2. Kriteria Pengujian

Penilaian atas pencapaian tujuan pengujian didapatkan dengan memperhatikan beberapa hasil yang diharapkan berikut :

a. Kemampuan aplikasi untuk menampilkan antar muka yang sesuai.
b. Kemampuan aplikasi untuk mengambil data dari *database*.
c. Kemampuan aplikasi untuk menyimpan data *input* pengguna ke *database*.
d. Kemampuan aplikasi untuk menyimpan *file input* ke *storage*.
e. Kemampuan WAV *converter* dalam mengkonversi data Audio menjadi *file* WAV yang dapat diakses aplikasi.

6.3. Skenario Pengujian

Skenario pengujian dilakukan dengan melakukan peran sebagai dokter yang akan menjalankan fitur-fitur. Langkah-langkah untuk setiap kebutuhan fungsionalitas yaitu sebagai berikut :

1. *Login* sebagai Dokter dan Mengakses halaman *Home* Aplikasi
    - Pengguna dapat *login* sebagai Dokter di aplikasi SWiDS melalui perangkat *mobile*.
    - Aplikasi dapat menampilkan *Home* berisi Nama Dokter, *Inbox* konsultasi, dan Status Konsultasi.

2. Mengaktifkan Koneksi
    - Status konsultasi berubah menjadi berwarna hijau.
    - Muncul permintaan konsultasi baru pada *inbox* (setelah dilakukan pemilihan dokter di aplikasi sisi nakes).

3. Memulai Sesi Konsultasi
    - Dokter dapat melihat nama nakes, data pasien, dan hasil rekaman.

4. Mengirim Diagnosis dan Resep Obat
    - Dokter dapat mengisi diagnosis dan resep obat.
    - Dokter dapat mengirim diagnosis dan resep obat yang telah diisi.
    - Diagnosis dan resep obat tersimpan pada *database*.

5. Melihat Riwayat Konsultasi
    - Aplikasi menampilkan daftar riwayat konsultasi yang telah dilakukan.
    - Dokter dapat mencari riwayat konsultasi berdasarkan nama nakes, klinik, atau jenis rekaman (jantung/paru-paru/*abdomen*).

- Dokter dapat melihat detail riwayat konsultasi yang berisi nama nakes, data pasien, audio hasil rekaman, diagnosis, dan resep obat.

6. Menonaktifkan Koneksi
   - Status pada aplikasi berubah menjadi nonaktif.
   - Sistem menghapus permintaan konsultasi pada kotak *inbox*.

Selain itu, dilakukan pula pengujian layanan WAV *converter* dengan mengakses Aplikasi sisi Nakes yang telah terhubung dengan stetoskop. Selanjutnya, pengguna akan memulai perekaman dengan mengoperasikan stetoskop. Didapatkan hasil pengujian WAV *converter* sebagai berikut:

- Pengguna dapat mendengarkan audio dengan jelas.
- Sistem dapat mengonversi *file* audio segera setelah proses perekaman selesai.

Pengujian lain dilakukan bersama para *stakeholder* aplikasi dalam rapat bulanan untuk mengevaluasi implementasi fitur dan kemajuan pengembangan aplikasi SWiDS.

6.4. Evaluasi Pengujian

Hasil pengujian dilakukan terhadap pengamatan mengenai perilaku aplikasi SWiDS terhadap kasus skenario uji coba. Tabel 1 di bawah ini menjelaskan hasil uji coba terhadap aplikasi yang telah dibuat.

Kriteria Pengujian	**Hasil Pengujian**

Aplikasi dapat menampilkan antar muka yang sesuai	Terpenuhi
Aplikasi dapat mengambil data dari *database*	Terpenuhi
Aplikasi dapat menyimpan data *input* pengguna ke *database*	Terpenuhi
Aplikasi dapat menyimpan *file input* ke *storage*	Terpenuhi
Layanan WAV *converter* dapat mengkonversi data Audio menjadi *file* WAV yang dapat diakses aplikasi	Terpenuhi

Tabel 1. Hasil Evaluasi Pengujian

[Halaman ini sengaja dikosongkan]

BAB VII
KESIMPULAN DAN SARAN

7.1. Kesimpulan

Kesimpulan yang didapat setelah melakukan pembangunan aplikasi SWiDS adalah sebagai berikut :

a. Aplikasi SWiDS sisi dokter dengan fitur utama aplikasi, termasuk integrasi dengan stetoskop, *inbox*, sesi konsultasi, dan riwayat konsultasi, telah berhasil dikembangkan sesuai dengan permintaan *stakeholder*.
b. Aplikasi SWiDS telah memudahkan dokter dalam memberikan diagnosis hasil rekam medis, mendukung pemeriksaan berbasis suara, dan menyimpan catatan riwayat konsultasi secara terstruktur, sehingga meningkatkan efisiensi layanan kesehatan.

7.2. Saran

Saran untuk pengembangan aplikasi SWiDS selanjutnya adalah sebagai berikut :

a. Menambahkan fitur pengolahan audio yang dapat membantu dokter untuk memperbaiki kualitas audio jika dirasa terdapat *noise*.
b. Menambah keamanan data medis dengan menerapakan sistem *enkripsi* yang lebih layak dan teruji.

[Halaman ini sengaja dikosongkan]

DAFTAR PUSTAKA

Figma (2024) *Figma: The Collaborative Interface Design Tool*. Available at: https://www.figma.com/.

FileFormat (2024) *What is a WAV file?* Available at: https://docs.fileformat.com/audio/wav/.

Firebase (2024) *Dokumentasi Firebase*. Available at: https://firebase.google.com/docs?hl=id.

Flutter (2024) *Flutter—Build apps for any screen*. Available at: https://flutter.dev/.

Kramer, N. (2024) *https://daily.dev/blog/get-to-know-asynchio-multithreaded-python-using-asyncawait*. Available at: https://daily.dev/blog/get-to-know-asynchio-multithreaded-python-using-asyncawait.

MDN (2024) *Mobile accessibility—Learn web development*. Available at: https://developer.mozilla.org/en-US/docs/Learn/Accessibility/Mobile.

Systemd (2024) *System and Service Manager*. Available at: https://systemd.io/.

[Halaman ini sengaja dikosongkan]

BIODATA PENULIS I

Nama	: Thoriq Afif Habibi
Tempat, Tanggal Lahir	: Surabaya, 25 April 2003
Jenis Kelamin	: Laki-laki
Telepon	: +6282131322005
Email	: thoriq.afif.habibi@gmail.com

AKADEMIS

Kuliah	: Departemen Teknik Informatika – FTEIC , ITS
Angkatan	: 2021
Semester	: 7 (Tujuh)