PENGEMBANGAN APLIKASI PENDETEKSI KANTUK PADA PENGENDARA KENDARAAN BERMOTOR DENGAN MENGGUNAKAN SENSOR DETAK JANTUNG PADA *SMARTWATCH*

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Imam Farouqi Faisal
NIM: 165150200111147

PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2020

PENGESAHAN

PENGEMBANGAN APLIKASI PENDETEKSI KANTUK PADA PENGENDARA KENDARAAN BERMOTOR DENGAN MENGGUNAKAN SENSOR DETAK JANTUNG PADA *SMARTWATCH*

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Imam Farouqi Faisal
NIM: 165150200111147

Skripsi ini telah diuji dan dinyatakan lulus pada
6 Januari 2020

Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
Agi Putra Kharisma, S.T., M.T.	Ir. Sutrisno, M.T.
NIK: 2013048604301001	NIP: 195703251987011001

Mengetahui
Ketua Jurusan Teknik Informatika

Tri Astoto Kurniawan, S.T., M.T., Ph.D.
NIP: 197105182003121001

PERNYATAAN ORISINALITAS

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah skripsi ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis disitasi dalam naskah ini dan disebutkan dalam daftar referensi.

Apabila ternyata didalam naskah skripsi ini dapat dibuktikan terdapat unsur-unsur plagiasi, saya bersedia skripsi ini digugurkan dan gelar akademik yang telah saya peroleh (sarjana) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, 19 Desember 2019

Imam Farouqi Faisal

NIM: 165150200111147

ABSTRAK

Imam Farouqi Faisal, Pengembangan Aplikasi Pendeteksi Kantuk Pada Pengendara Kendaraan Bermotor dengan Menggunakan Sensor Detak Jantung Pada *Smartwatch*

Pembimbing: Agi Putra Kharisma, S.T., M.T. dan Ir. Sutrisno, M.T.

Mayoritas kecelakaan lalu lintas yang terjadi disebabkan oleh *human error*, salah satunya ialah mengantuk. Ketika mengantuk, konsentrasi pengendara cenderung menurun, sehingga dapat menyebabkan pengendara kehilangan kontrol atas kendaraannya, hal ini dapat berujung terjadinya kecelakaan lalu lintas. Oleh karena itu, perlu adanya solusi yang dapat mencegah terjadinya kecelakaan lalu lintas yang disebabkan oleh pengendara yang mengantuk. Berdasarkan permasalahan tersebut, penulis menawarkan suatu sistem yang dapat mendeteksi kondisi kantuk pengendara, sehingga jika pengendara terdeteksi mengantuk dapat segera diberikan peringatan. Sistem yang ditawarkan berbentuk aplikasi berbasis Android yang menggunakan data detak jantung untuk mendeteksi kondisi kantuk pengendara. Untuk memperoleh data detak jantung, aplikasi ini memanfaatkan sensor detak jantung pada *smartwatch* yang harus dikenakan pengendara.

Dalam pengembangannya, aplikasi ini dirancang dengan pendekatan *Object Oriented Design* (OOD) dan diimplementasikan menggunakan metode *Object Oriented Programming* (OOP). Pada aplikasi ini dilakukan pengujian *black box,* akurasi, dan *usability.* Hasil pengujian *black box* menunjukkan bahwa aplikasi ini telah memiliki tingkat validitas sebesar 100%, pengujian akurasi dilakukan terhadap lima responden dan menghasilkan tingkat akurasi sebesar 86,3%, yang menunjukkan bahwa aplikasi dapat mendeteksi kantuk kelima responden dengan cukup akurat, sedangkan berdasarkan pengujian *usability,* aplikasi memperoleh skor 83,5 yang termasuk dalam skala B *(Excellent).*

Kata kunci: android, *smartwatch, smartphone, usability,* kantuk

ABSTRACT

Imam Farouqi Faisal, Development of Motor Vehicle Driver Drowsiness Detection Application Using Heart Rate Monitor Sensor on Smartwatch

Supervisors: Agi Putra Kharisma, S.T., M.T. dan Ir. Sutrisno, M.T.

The majority of traffic accidents that occur are caused by human error, one of which is drowsy. When drowsy, the driver's concentration tends to decrease, so it can cause loss of control over the vehicle, this can lead to traffic accidents. Therefore, there's a need for a solution that can prevent traffic accidents caused by drowsiness. Based on these problems, the author offers a system that can detect driver drowsiness, so if the driver is drowsy, he can be given a warning immediately. This system is an Android based application that uses heart rate data to detect drowsiness. To obtain heart rate, this application is using the heart rate monitor sensor on the smartwatch which the driver must wear.

In its development, this application is designed using Object Oriented Design (OOD) approach dan implemented using Object Oriented Programming (OOP) method. Black box, accuracy, dan usability are tested. Black box testing result shows that this application has 100% validity rate, based on accuracy testing conducted on five respondents, this application has 86,3% accuracy rate, showing that this application can detect the respondents' drowsiness accurately, whereas based on usability testing, this application has 83,5 score which means it got B (Excellent) rate.

Keywords : android, smartwatch, smartphone, usability, drowsy

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR LAMPIRAN

BAB 1 PENDAHULUAN

Bab pendahuluan terbagi ke dalam enam subbab, diantaranya Latar Belakang, Rumusan Masalah, Tujuan, Manfaat, Batasan Masalah, dan Sistematika Pembahasan.

1.1 Latar Belakang

Kecelakaan lalu lintas merupakan penyebab kematian tertinggi ke-8 di dunia (WHO, 2018). Masih menurut data WHO (2018), kecelakaan lalu lintas diperkirakan merenggut nyawa sebanyak 1,35 juta jiwa tiap tahunnya. Di Indonesia, berdasarkan data yang dirilis Korlantas Polri jumlah kasus kecelakaan lalu lintas sepanjang tahun 2018 adalah sebanyak 215.492 kasus, dengan jumlah korban meninggal dunia sebanyak 50.416. Faktor utama penyebab kecelakaan lalu lintas adalah *human error,* salah satunya ialah mengantuk. Studi yang dilakukan *American Automobile Association* (AAA) (2018) menunjukkan bahwa sekitar 10% kecelakaan lalu lintas yang terjadi diakibatkan oleh pengemudi yang mengantuk. Fakta-fakta di atas menunjukkan bahwa mengantuk merupakan salah satu penyebab utama kecelakaan lalu lintas.

Berkendara merupakan aktivitas yang memerlukan fokus tinggi, sehingga berkendara dalam durasi yang lama akan menyebabkan kelelahan dan mengakibatkan pengendara mengantuk. Berkendara dalam kondisi mengantuk dapat menyebabkan pengendara tertidur secara tidak sadar, sehingga kehilangan kontrol kendaraannya, hal ini sangatlah berbahaya karena dapat menyebabkan kecelakaan lalu lintas. Untuk dapat mengurangi jumlah kasus kecelakaan lalu lintas yang disebabkan oleh mengantuk, diperlukan adanya solusi, salah satunya adalah dengan membuat sistem yang dapat mendeteksi kantuk pada pengendara, tujuannya adalah agar pengendara yang terdeteksi mengantuk segera diberikan peringatan.

Terdapat beberapa metode yang dapat digunakan untuk mendeteksi kantuk pada pengendara. Yang pertama adalah *vehicle-based measures,* yaitu dengan memonitor kendaraan, contohnya dengan mengukur kecepatan, pergerakan roda kemudi, atau tekanan pada pedal gas (Liu, Hosking dan Lenné, 2009). Metode berikutnya adalah *behavioral measures*, yaitu dengan memonitor pergerakan pengendara, seperti menguap, kedipan mata, atau posisi kepala (Fan, Yin dan Sun, 2007). Yang terakhir adalah *physiological measures,* yaitu dengan memonitor psikologis pengendara, misalnya detak jantung, pergerakan otot, atau pergerakan retina (Sahayadhas, Sundaraj dan Murugappan, 2012). Salah satu dari metode di atas akan diimplementasikan ke dalam sebuah sistem pendeteksi kantuk dalam bentuk aplikasi *mobile* berbasis Android.

Dalam penelitian ini, metode yang digunakan adalah *physiological measures*, yaitu dengan cara mengukur detak jantung. Terdapat dua alasan mengapa metode

ini dipilih. Alasan pertama adalah karena metode *physiological measures* lebih akurat dibandingkan metode-metode lainnya, hal ini dikarenakan psikologi manusia memberikan sinyal yang mengindikasikan kondisi mengantuk paling cepat, sehingga peringatan bisa diberikan sesegera mungkin (Sahayadhas, Sundaraj dan Murugappan, 2012). Alasan kedua ialah pada penelitian ini akan memanfaatkan perangkat *wearable* untuk memonitor kondisi pengendara, yaitu *smartwatch*, yang mana memiliki sensor *Heart Rate Monitor* (HRM) yang berfungsi untuk memonitor detak jantung penggunanya. *Smartwatch* akan memonitor detak jantung pengendara secara *real-time* dan terhubung dengan aplikasi pada Android, dari data detak jantung yang didapatkan, maka kondisi pengemudi akan dapat dideteksi, jika terdeteksi mengantuk, maka aplikasi pada Android akan mengeluarkan suara alarm sekaligus menampilkan pesan peringatan pada pengendara, selain itu aplikasi juga akan menandai lokasi di mana pengendara tersebut terdeteksi mengantuk dan secara otomatis akan mengirimkan SMS notifikasi yang menginformasikan kondisi pengendara sedang mengantuk dan lokasi pengendara saat terdeteksi mengantuk kepada keluarga atau rekan pengendara yang nomor teleponnya telah disimpan terlebih dahulu dalam aplikasi.

1.2 Rumusan Masalah

1. Bagaimana kebutuhan fungsional dan non-fungsional dari aplikasi pendeteksi kantuk pada pengendara?
2. Bagaimana hasil rancangan aplikasi pendeteksi kantuk pada pengendara?
3. Bagaimana hasil implementasi aplikasi pendeteksi kantuk pada pengendara?
4. Bagaimana kinerja aplikasi pendeteksi kantuk pada pengendara?

1.3 Tujuan

1.3.1 Tujuan Umum

Mengembangkan aplikasi berbasis Android untuk mendeteksi kantuk pada pengendara dengan memanfaatkan sensor detak jantung pada *smartwatch*.

1.3.2 Tujuan Khusus

1. Mengidentifikasi persyaratan fungsional dan non fungsional aplikasi pendeteksi kantuk pada pengendara.
2. Merancang aplikasi pendeteksi kantuk dengan pemodelan berorientesi objek.
3. Mengimplementasikan aplikasi pendeteksi kantuk dengan teknologi berorientasi objek.
4. Menguji aplikasi tersebut sesuai dengan persyaratan fungsional dan non fungsionalnya.

1.4 Manfaat

Manfaat dari aplikasi pendeteksi kantuk pada pengendara yang dikembangkan dalam skripsi ini adalah dapat mencegah terjadinya kecelakaan lalu lintas, terutama yang disebabkan oleh pengendara yang mengantuk, sehingga dapat mengurangi jumlah kasus kecelakaan lalu lintas di Indonesia yang tentunya akan mengurangi angka korban yang diakibatkan oleh kecelakaan lalu lintas.

1.5 Batasan Masalah

Batasan-batasan dari penelitian ini adalah sebagai berikut :

1. Aplikasi pendeteksi kantuk pada pengendara hanya berjalan pada platform Android.
2. Aplikasi untuk memonitor detak jantung hanya berjalan pada *smartwatch* berbasis Tizen.

1.6 Sistematika Pembahasan

- **Bab 1 : Pendahuluan**
  Pada bab ini mendeskripsikan tentang apa yang dilakukan pada penelitian serta alasan mengapa melakukan penelitian tersebut.

- **Bab 2 : Landasan Kepustakaan**
  Pada bab ini menjelaskan apa saja konteks masalah serta ilmu yang mendukung penelitian ini.

- **Bab 3 : Metodologi Penelitian**
  Pada bab ini menjelaskan metodologi yang digunakan dalam penelitian ini.

- **Bab 4 : Analisis Kebutuhan**
  Pada bab ini menjelaskan proses analisis kebutuhan dalam penelitian ini.

- **Bab 5 : Perancangan**
  Pada bab ini menjelaskan proses perancangan sistem dalam penelitian ini.

- **Bab 6 : Implementasi**
  Pada bab ini menjelaskan proses implementasi sistem dalam penelitian ini.

- **Bab 7 : Pengujian**
  Pada bab ini menjelaskan hasil pengujian yang dilakukan dalam penelitian ini.

- **Bab 8 : Penutup**
  Pada bab ini berisi kesimpulan dan saran terhadap penelitian yang dilaksanakan dalam skripsi ini.

BAB 2 LANDASAN KEPUSTAKAAN

Pada bab ini dijabarkan pembahasan mengenai teori dan konsep yang digunakan dalam penelitian ini. Bab Landasan Kepustakaan terbagi ke dalam dua bagian yaitu Tinjauan Pustaka dan Dasar Teori.

2.1 Tinjauan Pustaka

Bahasan mengenai pendeteksian kantuk pada pengendara telah cukup umum diteliti sebelumnya, setelah melakukan analisis terhadap beberapa penelitian yang ada, penulis menemukan tiga penelitian yang memiliki kemiripan topik bahasan dengan penelitian ini.

Penelitian pertama dilakukan oleh Ray Toban dan Alif Finandhita pada tahun 2017 dengan judul "Pembangunan Aplikasi Pendeteksi Kantuk Berbasis Android". Hasil penelitian ini ialah sebuah aplikasi berbasis Android yang dapat mengingatkan pengendara jika terdeteksi mengantuk. Aplikasi ini memanfaatkan *smartband* untuk mengukur detak jantung secara *real-time*, kemudian jika detak jantung terdeteksi kurang dari 60 BPM, maka pengguna dianggap mengantuk sehingga aplikasi akan memberikan peringatan. Kesamaan penelitian tersebut dengan penelitian ini yang pertama adalah pada tujuannya, yaitu untuk mendeteksi kantuk dengan memanfaatkan detak jantung, yang kedua adalah pada platform yang digunakan, yaitu Android. Sedangkan perbedaannya terdapat pada metode analisis detak jantung yang digunakan serta fitur-fitur yang terdapat pada aplikasi .

Penelitian kedua adalah penelitian yang dilakukan oleh Azka Zakiyyatuddin dkk. pada tahun 2018 yang berjudul "*Wristband* Inovatif Penghilang Kantuk Saat Belajar dengan Sensor Detak Jantung Berbasis IOT". Hasil penelitian ini ialah sebuah perangkat berbentuk *wristband* yang memanfaatkan sensor detak jantung dan *microcontroller* yang bernama "KANMURU" yang dapat bergetar jika mendeteksi pengguna dalam keadaan mengantuk. Alat tersebut dapat tersambung ke aplikasi IOT bernama "Blynk" pada Android, yang akan menampilkan detak jantung pengguna. Pada aplikasi tersebut pengguna diminta memasukkan ambang batas *(threshold)* detak jantung, sehingga jika detak jantung pengguna kurang dari *threshold*, maka dianggap mengantuk dan *wristband* akan memberikan efek kejut ke pengguna. Kesamaan penelitian tersebut dengan penelitian ini ialah pada tujuannya, yaitu untuk mendeteksi kantuk dengan memanfaatkan detak jantung. Sedangkan perbedaannya ialah pada metode analisis detak jantung dan perangkat yang digunakan.

Penelitian ketiga ialah penelitian yang dilakukan oleh Prima Dewi Purnamasari dan Aziz Zul Hazmi pada tahun 2018 yang berjudul "*Heart Beat Based Drowsiness Detection System for Driver*". Hasil dari penelitian ini adalah sebuah perangkat pendeteksi kantuk yang memanfaatkan *microcontroller* Arduino Nano, *microcomputer* Odroid XU4, LCD, dan sensor detak jantung. Cara kerjanya adalah

pertama-tama alat ini akan menyimpan rata-rata detak jantung pengguna dalam 5 detik pertama, yang akan dijadikan sebagai *threshold*, kemudian jika detak jantung pengguna terdeteksi kurang dari nilai *threshold* maka pengguna dianggap mengantuk sehingga alat akan menampilkan pesan peringatan pada layar LCD. Kesamaan penelitian tersebut dengan penelitian ini ialah pada tujuannya, yaitu untuk mendeteksi kantuk dengan memanfaatkan detak jantung, selain itu juga pada metode analisis detak jantung yang digunakan. Sedangkan perbedaannya ialah pada perangkat untuk mengukur detak jantung pengguna.

2.2 Dasar Teori

2.2.1 Kantuk

Kantuk ialah suatu keadaan di mana seseorang cenderung merasa ingin tidur (Hall, 2016). Tidur dapat dikategorikan dalam 3 tahap, yaitu *awake*, *Non-Rapid Eye Movement Sleep* (NREM), dan *Rapid Eye Movement Sleep* (REM) (Sahayadhas, Sundaraj dan Murugappan, 2012). Fase NREM terbagi lagi dalam 3 fase, di antaranya mengantuk, tidur ringan, dan tidur berat (Brodbeck et al., 2012). Rasa kantuk yang timbul saat malam hari setelah selesai beraktivitas adalah normal, namun jika rasa kantuk datang saat siang hari di mana kita sedang beraktivitas, maka hal ini menjadi masalah, karena akan sangat mengganggu aktivitas sehari-hari.

Menurut *American Sleep Association* (ASA), terdapat 9 faktor yang dapat menyebabkan kantuk pada siang hari, diantaranya :

1. Lingkungan tempat tidur tidak nyaman.
2. Durasi tidur tidak cukup.
3. Jadwal tidur yang tidak tentu.
4. Tidak ada rutinitas sebelum tidur.
5. Minum kafein atau makan berat tepat sebelum tidur.
6. Olahraga tepat sebelum tidur.
7. Tidur karena minuman beralkohol.
8. Efek obat atau penyakit kronis.
9. Kelainan tidur.

2.2.2 Detak Jantung

Detak jantung *(heart rate)* adalah jumlah banyaknya jantung berdetak dalam satu waktu (MedicineNet, 2019). Jantung berdetak bertujuan untuk memompa darah yang telah bersih dari ventrikel kiri ke seluruh pembuluh darah tubuh melalui aorta. Pada umumnya, detak jantung diukur dalam kurun waktu satu menit, sehingga detak jantung memiliki satuan *beats per minute* (BPM). Detak jantung dapat berubah-ubah tergantung dari aktivitas manusia. Terdapat beberapa faktor yang dapat memengaruhi detak jantung, yaitu temperatur udara, posisi tubuh, emosi, ukuran tubuh, dan penggunaan obat. Detak jantung juga berbeda-beda pada tiap rentang usia, Tabel 2.1 menunjukkan detak jantung pada

saat aktivitas normal berdasarkan rentang usia menurut *National Institute of Health* (NIH) (MacGill, 2017).

Tabel 2.1 Detak Jantung Berdasarkan Usia

Usia	Detak jantung (bpm)
0-1 bulan	70-190
1-11 bulan	80-160
1-2 tahun	80-130
3-4 tahun	80-120
5-6 tahun	75-115
7-9 tahun	70-110
>10 tahun	60-100

Bagian tubuh yang dapat digunakan untuk mengukur detak jantung diantaranya pergelangan tangan, dibawah alis, sisi leher, dan diatas telapak kaki, namun pada umumnya pergelangan tangan menunjukkan hasil yang lebih akurat (Harvard Health Publishing, 2019). Durasi yang cukup untuk mendapatkan hasil pengukuran detak jantung yang akurat menurut *Consensus Panel of the European Society of Hypertension* (dalam Palatini, 2010) adalah selama 30 detik.

Detak jantung dapat dijadikan indikator kondisi kantuk seseorang. Pada saat dalam kondisi mengantuk, terjadi akumulasi hormon adenosin dalam otak (Peters, 2019). Akumulasi hormon adenosin inilah yang memicu saraf parasimpatik untuk menurunkan detak jantung (Holst dan Landolt, 2015). Detak jantung seseorang dalam keadaan mengantuk akan berkurang sekitar 8 BPM dibandingkan dengan saat keadaan normal sehari-hari (Waldeck dan Lambert, 2003).

2.2.3 Sensor *Heart Rate Monitor*

Sensor *Heart Rate Monitor* (HRM) adalah sensor elektroda yang berfungsi untuk mengukur detak jantung dan menampilkan hasil pengukuran tersebut secara kontinyu (Bumgardner, 2019). Pada prinsipnya, cara kerja sensor ini adalah dengan mengukur perubahan volume darah yang melewati suatu organ tubuh, dimana perubahan volume darah akan menyebabkan perubahan pula pada intensitas cahaya yang melewati organ tersebut (Elprocus, 2013).

Sensor HRM menerapkan teknik yang disebut *photoplethysmography* (PPG), yaitu dengan memancarkan cahaya ke kulit sehingga akan ada cahaya yang terpantul akibat aliran darah, jumlah pantulan cahaya tersebut yang diukur oleh sensor (Valencell, 2015). Untuk dapat mengukur ddetak jantung, sensor HRM memadukan beberapa komponen yaitu (Valencell, 2015) :

1. *Optical emitter*, merupakan gabungan dari setidaknya dua buah LED yang berfungsi memancarkan cahaya ke kulit.
2. *Digital Signal Processor* (DSP), berfungsi untuk menangkap pantulan cahaya yang kemudian dikonversi kedalam bilangan biner.
3. *Accelerometer*, digunakan untuk mengukur pergerakan pengguna.
4. Algoritma, dari sinyal yang didapatkan DSP dan dikombinasikan dengan data pergerakan pengguna dari sensor *accelerometer*, dibuat algoritma yang dapat menentukan BPM detak jantung pengguna.

2.2.4 *Smartwatch*

Smartwatch atau jam tangan pintar secara bahasa berarti sebuah perangkat *mobile* yang dirancang untuk dapat digunakan di pergelangan tangan (Oxford Press, 2019). *Smartwatch* merupakan sebuah perangkat komputasi *wearable* yang bentuknya menyerupai arloji (Rouse, 2019). Disamping fungsi utamanya yaitu untuk menunjukkan waktu, sama seperti *smartphone, smartwatch* juga memiliki fungsi yang beragam, hal ini dikarenakan pada *smartwatch* modern juga terdapat sistem operasi mobile, sehingga dapat dipasang berbagai aplikasi didalamnya.

Pada umumnya, *smartwatch* tidak dirancang sebagai perangkat *standalone*, oleh karenanya setiap *smartwatch* pasti memiliki koneksi *Bluetooth* ataupun *Wi-Fi* agar dapat terhubung dengan *smartphone*. Pengguna *smartwatch* juga dapat memanfaatkan perangkatnya untuk memonitor kesehatan, karena *smartwatch* juga memiliki fitur-fitur kesehatan seperti monitor detak jantung, *fitness tracker,* dan *sleep tracker*. Saat ini telah banyak *brand* yang memproduksi *smartwatch*, seperti Apple, Samsung, Xiaomi, dan Motorola.

2.2.5 Android

Android merupakan sebuah sistem operasi milik Google yang dirancang untuk berjalan pada perangkat *mobile*, seperti *smartphone* atau tablet. Sistem operasi Android dikembangkan oleh Google bekerjasama dengan sejumlah manufaktur besar seperti HTC, Samsung, Intel, dan Qualcomm membentuk aliansi yang diberi nama *Open Handset Alliance* (OHA) (Callaham, 2018).

Android merupakan sistem operasi yang berbasis Linux 2.6 yang dimodifikasi, sehingga *source code* dari sistem operasi ini bersifat *open source*. Android terdiri dari sejumlah aplikasi dan *library* Java yang berjalan diatas *framework object-oriented* Java dan Dalvik Virtual Machine (DVM), sehingga memungkinkan Android dijalankan pada perangkat *mobile* yang memiliki spesifikasi terbatas (Techopedia, 2019).

2.2.6 Tizen

Tizen adalah sistem operasi yang dikembangkan oleh Linux Foundation. Tizen adalah sistem operasi yang difokuskan untuk perangkat *mobile* dan ekosistemnya yang dapat disesuaikan dengan kebutuhan masing-masing manufaktur (Tizen, 2019). Tizen merupakan salah satu sistem operasi *open source* yang dikembangkan dengan Kernel Linux.

Saat ini, manufaktur yang mengimplementasikan sistem operasi ini pada produknya hanya Samsung. Tizen digunakan sebagai sistem operasi produk-produk elektronik Samsung diantaranya *smartwatch, smartphone,* printer, kamera, dan *smart TV*. Aplikasi *native* berbasis Tizen dapat dikembangkan dengan menggunakan teknologi web (HTML, CSS, JS) ataupun dengan bahasa C.

2.2.7 Samsung Accessory SDK

Samsung Accessory adalah sebuah *Software Development Kit* (SDK) yang berfungsi untuk menghubungkan perangkat aksesori Samsung berbasis Tizen dengan *smartphone* Android (Samsung Developers, 2018). Samsung Accessory SDK memungkinkan komunikasi antara perangkat aksesori dengan *smartphone* Android, sehingga selain dapat saling bertukar data, kedua perangkat juga dapat saling memanfaatkan fitur satu sama lain.

Samsung Accessory SDK menyediakan protokol khusus yang bernama Samsung Accessory Protocol (SAP). SAP mendukung koneksi Wi-Fi dan Bluetooth, sehingga antar perangkat dapat saling terhubung dengan memanfaatkan kedua koneksi tersebut.

2.2.8 Pengujian *Black Box*

Pengujian *black box* ialah metode pengujian perangkat lunak menginisiasi kasus uji berdasarkan spesifikasi kebutuhan. Pengujian *black box* tidak memperhatikan struktur internal dari PL, oleh karenanya penguji tidak harus memiliki akses ke kode program. Fokus dari metode pengujian ini ialah keluaran program yang dihasilkan oleh masukan yang telah ditentukan sebelumnya oleh penguji (Nidhra dan Dondeti, 2012).

2.2.9 Pengujian *Usability*

Pengujian *usability* adalah metode pengujian suatu produk yang bertujuan untuk mengetahui seberapa mudah pengguna dapat mengoperasikan suatu produk (Experience UX, 2019). Pengujian *usability* sangat menekankan pada pengguna, karena untuk dapat menciptakan produk yang *usable*, maka kita harus mengerti pengguna dari produk kita (Dumas, Dumas dan Redish, 1999).

2.2.10 *Karolinska Sleepiness Scale*

Karolinska Sleepiness Scale (KSS) ialah suatu skala yang digunakan untuk mengukur tingkat kantuk seseorang secara subjektif. KSS sudah umum digunakan dalam berbagai penelitian yang berkaitan dengan kantuk (Miley, Kecklund dan Åkerstedt, 2016). Tabel 2.2 menunjukkan tingkatan skala pada KSS (Åkerstedt dan Gillberg, 1990).

Tabel 2.2 *Karolinska Sleepiness Scale* (KSS)

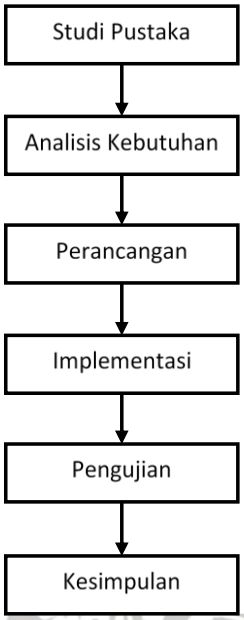
Deskripsi	*Karolinska Sleepiness Scale* (KSS)
Extremely alert	1
Very alert	2
Alert	3
Rather alert	4
Neither alert nor sleepy	5
Some signs of sleepiness	6
Sleepy, but no effort to keep awake	7
Sleepy, but some effort to keep awake	8
Very sleepy, great effort to keep awake, fighting sleep	9

Pada penelitian ini, KSS akan digunakan pada tahap pengujian akurasi, yaitu untuk mengukur tingkat kantuk responden.

BAB 3 METODOLOGI PENELITIAN

Pada bab ini akan dijelaskan metodologi yang digunakan dalam penelitian ini. Metode yang diterapkan terdiri dari beberapa tahapan, diantaranya studi pustaka, analisis kebutuhan, perancangan, implementasi, pengujian, dan kesimpulan.

Gambar 3.3.1 Metodologi Penelitian



3.1 Studi Pustaka

Studi pustaka merupakan tahap awal dimana penulis melakukan pengumpulan literatur penelitian yang memiliki keterkaitan dengan topik bahasan pada penelitian ini, selanjutnya penelitian-penelitian tersebut dipelajari dan dianalisis sebagai acuan dasar teori. Teori-teori yang dipelajari dari kumpulan literatur yang didapat antara lain :

1. Kantuk
2. Detak Jantung
3. Sensor *Heart Rate Monitor*
4. *Smartwatch*
5. Android
6. Tizen
7. Samsung Accessory SDK
8. Pengujian *Black Box*
9. Pengujian *Usability*

10. *Karolinska Sleepiness Scale* (KSS)

3.2 Analisis Kebutuhan

Tahap analisis kebutuhan merupakan tahap di mana kebutuhan-kebutuhan yang ada dianalisis lebih lanjut, tujuannya adalah untuk menentukan fitur-fitur apa saja yang akan tersedia serta sebagai acuan pada tahap perancangan. Metode yang digunakan adalah dengan pemodelan berorientasi objek. Hasil dari tahap ini ialah berupa kebutuhan fungsional, *use case diagram* dan *use case scenario*.

3.3 Perancangan

Tahap perancangan bertujuan untuk menghasilkan rancangan PL yang memenuhi kebutuhan-kebutuhan yang telah didapat pada tahap sebelumnya. Rancangan yang dihasilkan akan dijadikan rujukan pada tahap-tahap berikutnya. Metode yang digunakan pada tahap ini adalah *Object Oriented Design* (OOD) dengan pola *Model View Viewmodel* (MVVM). Hasil dari tahap ini terbagi menjadi dua yaitu perancangan arsitektur berupa *sequence diagram* dan *class diagram,* serta perancangan antarmuka berupa *wireframe* tampilan aplikasi.

3.4 Implementasi

Tahap implementasi ialah tahap yang bertujuan untuk merealisasikan rancangan yang telah dibuat menjadi kode program. Metode yang digunakan yaitu *Object Oriented Programming* (OOP). Bahasa pemrograman yang digunakan adalah Kotlin.

3.5 Pengujian

Pengujian merupakan tahap dimana aplikasi yang telah selesai diimplementasikan diuji fungsionalnya. Tujuan dari tahap ini ialah menemukan kesalahan-kesalahan yang terdapat pada kode program, sehingga kesalahan tersebut dapat diperbaiki sebelum aplikasi digunakan oleh pengguna. Metode yang digunakan ialah pengujian *black box,* pengujian *usability*, dan pengujian akurasi.

3.6 Kesimpulan

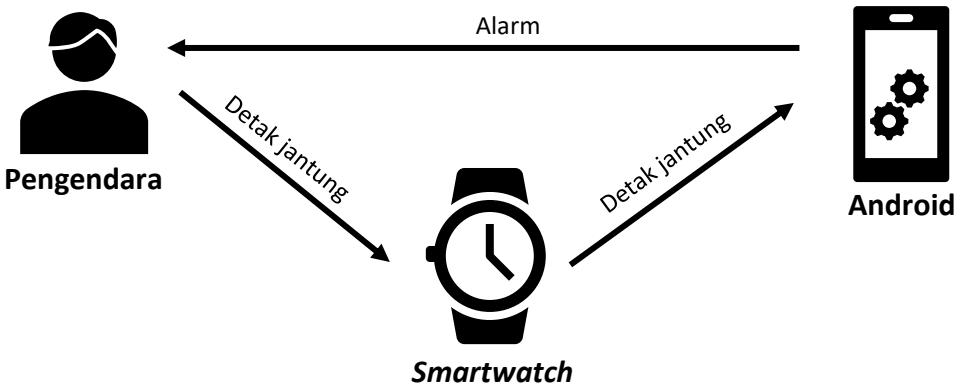
Tahap terakhir adalah kesimpulan, pada tahap ini peneliti melakukan penarikan kesimpulan terhadap hasil penelitian. Penarikan kesimpulan dimaksudkan untuk merangkum hasil yang didapat dari keseluruhan penelitian yang dilakukan. Untuk dapat menarik kesimpulan, peneliti akan meninjau kembali keseluruhan hasil penelitian yang didapatkan, kemudian mengkorelasikannya dengan hasil penelitian-penelitian sebelumnya dan dasar teori yang telah dikumpulkan. Selain kesimpulan, pada tahap ini penulis juga melakukan penarikan saran, yang bertujuan untuk menjadi bahan evaluasi bagi penelitian-penelitian selanjutnya yang hendak dilaksanakan.

BAB 4 ANALISIS KEBUTUHAN

Pada bab ini akan membahas proses analisis kebutuhan dari aplikasi pendeteksi kantuk yang terbagi ke dalam tujuh subbab diantaranya Gambaran Umum Aplikasi, Identifikasi Aktor, Aturan Penomoran, Kebutuhan Fungsional, Kebutuhan Non Fungsional, *Use Case Diagram,* dan *Use Case Scenario*.

4.1 Gambaran Umum Aplikasi

Aplikasi pendeteksi kantuk yang akan dibangun merupakan aplikasi berbasis Android. Fitur utama dari aplikasi ini tentunya adalah mendeteksi kantuk pengendara berdasarkan detak jantungnya. Alur proses aplikasi ini digambarkan pada Gambar 4.1.



Gambar 4.1 Gambaran Umum Aplikasi

Sebelum memulai pendeteksian kantuk, tentunya *smartwatch* dan Android harus terkoneksi terlebih dahulu menggunakan Bluetooth. Selanjutnya *smartwatch* akan mengukur detak jantung pengendara dan mengirimkannya ke aplikasi pada Android secara *real-time*. Setelah aplikasi menerima data detak jantung, pengendara harus menunggu terlebih dahulu selama 30 detik untuk mendapatkan nilai *threshold*, nilai *threshold* didapatkan dengan cara mengurangi rata-rata detak jantung pengendara dengan 8, sehingga nilai BPM *threshold* akan berbeda-beda pada setiap pendeteksian sesuai dengan rata-rata detak jantung pengendara pada 30 detik pertama. Durasi 30 detik digunakan karena menurut dasar teori detak jantung yang telah dijelaskan pada Bab 2, 30 detik merupakan durasi yang cukup untuk mengukur detak jantung secara akurat. Nilai *threshold* hanya ditentukan sekali pada setiap pendeteksian kantuk. Selanjutnya jika detak jantung berada di bawah atau sama dengan nilai *threshold*, maka aplikasi akan menyalakan alarm sebagai peringatan kepada pengendara.

4.2 Identifikasi Aktor

Aktor adalah pengguna dari sistem yang akan dibangun. Aktor merupakan entitas yang berada di luar sistem namun berinteraksi secara langsung dengan

sistem. Daftar aktor yang terlibat dengan aplikasi pendeteksi kantuk tertera pada Tabel 4.1.

Tabel 4.1 Identifikasi Aktor

No	Aktor	Karakteristik
1	Pengendara	Seluruh pengendara kendaraan bermotor.

4.3 Aturan Penomoran

Penomoran dilakukan untuk mengidentifikasi setiap kebutuhan yang ada sekaligus untuk membedakan antara kebutuhan fungsional dengan non fungsional. Aturan penomoran dari aplikasi pendeteksi kantuk dijelaskan pada

No.	Kode	Deskripsi
1	APK	Singkatan dari Aplikasi Pendeteksi Kantuk. Diletakkan di awal penomoran.
2	F	Singkatan dari Fungsional. Menunjukkan bahwa kebutuhan merupakan kebutuhan fungsional.
3	NF	Singkatan dari Non Fungsional. Menunjukkan bahwa kebutuhan merupakan kebutuhan non fungsional.
4	(Nomor Kebutuhan)	Merupakan nomor urut kebutuhan, dimulai dari 001.

Berikut adalah contoh penomoran kebutuhan.

APK-F-001 : menunjukkan kebutuhan fungsional dengan nomor urut 001.

4.4 Elisitasi Kebutuhan

Untuk menentukan kebutuhan dari sistem, metode elisitasi yang digunakan ialah pengamatan dokumen. Penulis melakukan observasi terhadap sejumlah artikel jurnal penelitian terkait yang telah dijelaskan pada bagian Tinjauan Pustaka pada Bab 2. Dari tiga penelitian yang diamati, masing-masing menghasilkan sistem yang berbeda namun memiliki kesamaan pada kebutuhannya secara umum, sehingga penulis mengembangkan kebutuhan tersebut untuk diterapkan menjadi kebutuhan fungsional dan non-fungsional pada sistem yang akan dibuat.

4.5 Kebutuhan Fungsional

Kebutuhan fungsional ialah fungsi-fungsi yang tersedia pada sistem yang akan dibangun. Daftar kebutuhan fungsional dari aplikasi pendeteksi kantuk tertera pada Tabel 4.2.

Tabel 4.2 Kebutuhan Fungsional Aplikasi Pendeteksi Kantuk

No.	Kode Fungsi	Nama Fungsi	Deskripsi
1	APK-F-001	Mendapatkan alarm peringatan	Fungsi untuk membunyikan alarm jika pengendara terdeteksi mengantuk.
2	APK-F-002	Mengirimkan pesan darurat	Fungsi untuk mengirimkan SMS notifikasi ke nomor darurat jika pengendara terdeteksi mengantuk.
3	APK-F-003	Menyimpan nomor telepon darurat	Fungsi untuk menyimpan nomor telepon darurat untuk dikirimi notifikasi jika pengendara terdeteksi mengantuk.
4	APK-F-004	Melihat histori	Fungsi untuk melihat histori pendeteksian kantuk pengendara.

4.6 Kebutuhan Non Fungsional

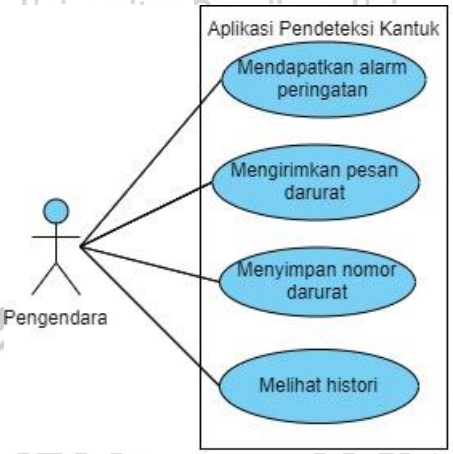
Kebutuhan non fungsional merupakan kebutuhan yang berhubungan dengan perikalu sistem yang akan dibangun. Daftar kebutuhan non fungsional dari aplikasi pendeteksi kantuk terdapat pada Tabel 4.3.

Tabel 4.3 Kebutuhan Non Fungsional Aplikasi Pendeteksi Kantuk

No.	Kode Fungsi	Nama Kebutuhan	Deskripsi
1	APK-NF-001	*Reliability*	Aplikasi harus dapat mendeteksi kantuk dengan tingkat akurasi minimal 80%.
2	APK-NF-002	*Usability*	Aplikasi harus memiliki skor *usability* dengan skala B *(Excellent).*

4.7 Use Case Diagram

Use Case Diagram merupakan diagram yang merepresentasikan interaksi antara aktor dengan sistem. Gambar 4.2 menunjukkan *Use Case Diagram* dari aplikasi pendeteksi kantuk.



Gambar 4.2 *Use Case Diagram* Aplikasi Pendeteksi Kantuk

4.8 Use Case Scenario

Use Case Scenario mendeskripsikan alur yang dijalankan oleh sistem untuk setiap *use case* atau fungsi yang tersedia. *Use Case Scenario* dari aplikasi pendeteksi kantuk terdapat pada Tabel 4.4 sampai dengan Tabel 4.7.

Tabel 4.4 *Use Case Scenario* Mendapatkan Alarm Peringatan

APK-F-001	Mendapatkan alarm peringatan
Objektif	Aplikasi menyalakan alarm sebagai peringatan.
Aktor	Pengendara
Pre Condition	Aplikasi telah terhubung dengan *smartwatch* dan telah memulai pendeteksian.
Main Flow	1. Aplikasi memerintahkan *smartwatch* untuk memonitor detak jantung dengan sensor HRM. 2. Aplikasi menampilkan nilai detak jantung secara *real-time*. 3. Aplikasi menghitung dan menyimpan nilai *threshold* detak jantung. 4. Jika detak jantung berada di bawah atau sama dengan *threshold*, maka aplikasi mendeteksi aktor dalam kondisi mengantuk. 5. Aplikasi menyalakan alarm peringatan.

Alternative Flow	-
Post Condition	Alarm berhasil dinyalakan dan pesan peringatan berhasil ditampilkan.

Tabel 4.5 *Use Case Scenario* Mengirimkan Pesan Darurat

APK-F-002	Mengirimkan pesan darurat
Objektif	Aplikasi mengirimkan pesan darurat ke nomor darurat.
Aktor	Pengendara
Pre Condition	Aplikasi telah terhubung dengan *smartwatch* dan telah memulai pendeteksian.
Main Flow	1. Aplikasi memerintahkan *smartwatch* untuk memonitor detak jantung dengan sensor HRM. 2. Aplikasi menampilkan nilai detak jantung secara *real-time*. 3. Aplikasi menghitung dan menyimpan nilai *threshold* detak jantung. 4. Jika detak jantung berada di bawah *threshold*, maka aplikasi mendeteksi aktor dalam kondisi mengantuk. 5. Aplikasi mendapatkan lokasi aktor. 6. Aplikasi mengirimkan pesan darurat ke nomor darurat.
Alternative Flow	-
Post Condition	Pesan darurat berhasil dikirimkan ke nomor darurat.

Tabel 4.6 *Use Case Scenario* Menyimpan Nomor Telepon Darurat

APK-F-003	Menyimpan nomor telepon darurat
Objektif	Aktor dapat menyimpan nomor telepon darurat.
Aktor	Pengendara
Pre Condition	Aktor berada pada halaman Nomor Darurat
Main Flow	1. Aktor menekan tombol “Tambah Kontak” 2. Aplikasi menampilkan *form* kontak darurat. 3. Aktor mengisi *form* kontak darurat. 4. Aktor menekan tombol “Simpan”.

	5. Aplikasi menyimpan nomor darurat di penyimpanan lokal.
Alternative Flow	1. Jika nomor sudah terdaftar, maka aplikasi akan menampilkan pesan *error.*
Post Condition	Nomor darurat berhasil disimpan di penyimpanan lokal.

Tabel 4.7 *Use Case Scenario* Melihat Histori

APK-F-004	Menampilkan histori
Objektif	Aktor dapat melihat histori pendeteksian kantuk.
Aktor	Pengendara
Pre Condition	Aktor berada pada halaman utama.
Main Flow	1. Aktor menekan teks “Lihat Semua” pada bagian Histori. 2. Aplikasi menampilkan daftar histori pendeteksian kantuk.
Alternative Flow	-
Post Condition	Daftar histori pendeteksian detak jantung berhasil ditampilkan.

BAB 5 PERANCANGAN

Pada bab ini menjelaskan tahapan dari perancangan aplikasi pendeteksi kantuk yang terbagi ke dalam dua bagian, yaitu Perancangan Arsitektur dan Perancangan Antarmuka.

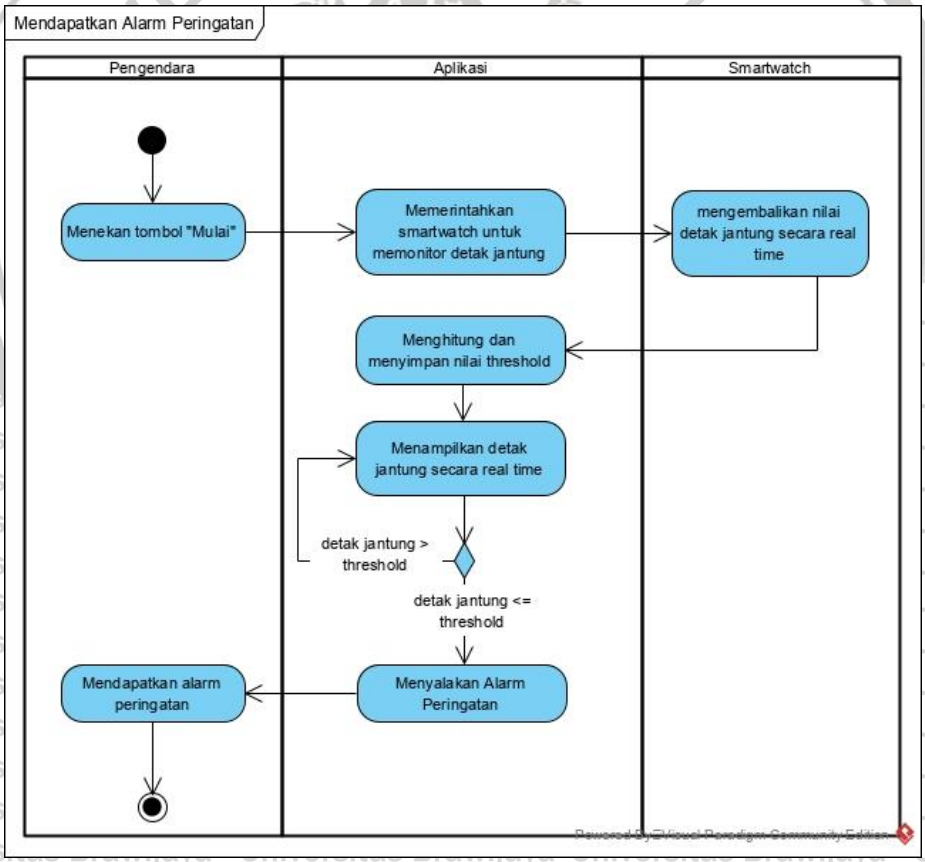
5.1 Perancangan Arsitektur

Perancangan arsitektur dari aplikasi pendeteksi kantuk terbagi dalam tiga bagian, yaitu *Activity Diagram, Sequence Diagram,* dan *Class Diagram*.

5.1.1 *Activity Diagram*

Activity diagram ialah diagram yang menggambarkan perilaku dari sistem yang dibangun. *Activity diagram* menjelaskan alur kontrol dari tiap aktivitas mulai dari awal hingga akhir. *Activity diagram* dari aplikasi pendeteksi kantuk dijabarkan pada subbab 5.1.1.1 sampai dengan 5.1.1.4.

5.1.1.1 Mendapatkan Alarm Peringatan (APK-F-001)

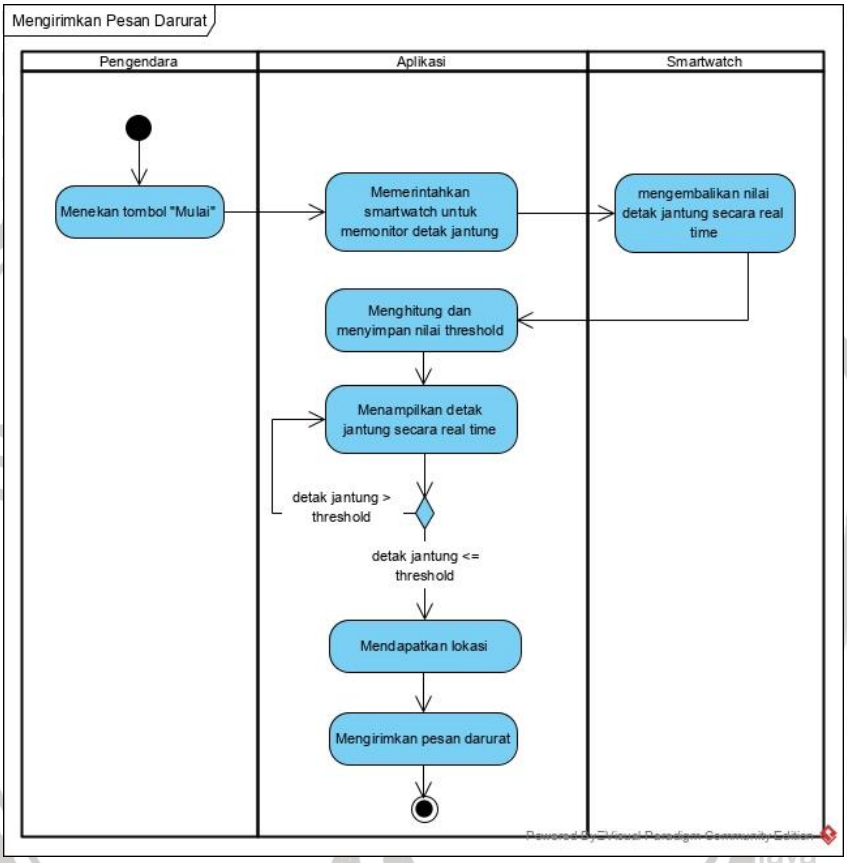


Gambar 5.1 *Activity Diagram* Mendapatkan Alarm Peringatan

Alur dari aktivitas mendapatkan alarm peringatan yang ditunjukkan pada Gambar 5.1 diawali ketika pengendara menekan tombol "Mulai" pada aplikasi, kemudian aplikasi akan memerintahkan *smartwatch* untuk memonitor detak

jantung pengendara sehingga *smartwatch* mengembalikan data detak jantung ke aplikasi secara *real-time*. Selanjutnya aplikasi akan menghitung nilai *threshold* detak jantung pengendara dan kemudian disimpan, lalu nilai detak jantung yang diterima ditampilkan secara *real-time* pula oleh aplikasi, kemudian aplikasi akan selalu memeriksa apakah nilai detak jantung yang diterima kurang dari atau sama dengan nilai *threshold*, jika tidak maka aplikasi akan terus menampilkan nilai detak jantung, namun jika ya, maka aplikasi akan secara otomatis menyalakan alarm, sehingga pengendara mendapatkan alarm peringatan.

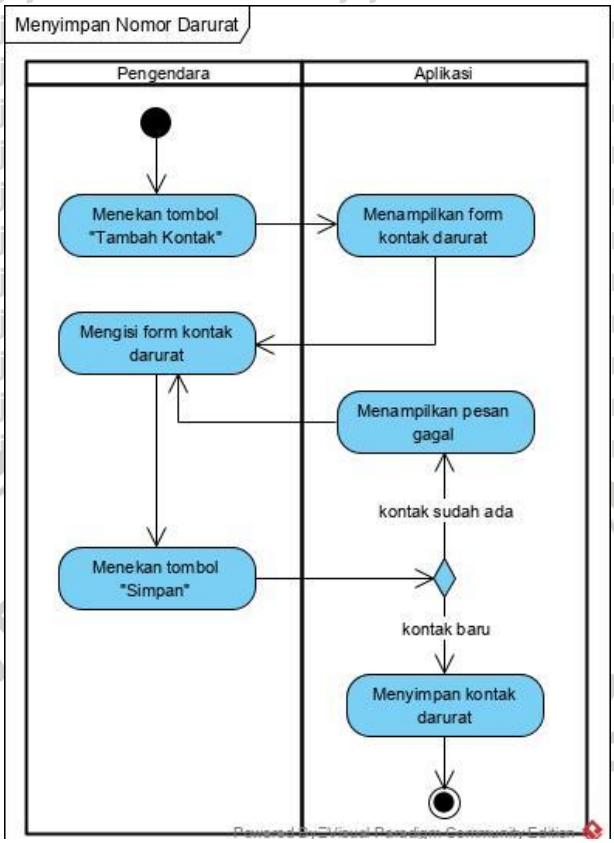
5.1.1.2 Mengirimkan Pesan Darurat (APK-F-002)



Gambar 5.2 *Activity Diagram* Mengirimkan Pesan Darurat

Gambar 5.2 menunjukkan alur dari aktivitas mengirimkan pesan darurat. Dimulai saat pengendara menekan tombol "Mulai" pada aplikasi, kemudian aplikasi akan memerintahkan *smartwatch* untuk memonitor detak jantung pengendara sehingga *smartwatch* mengembalikan data detak jantung ke aplikasi secara *real-time*. Selanjutnya aplikasi akan menghitung nilai *threshold* detak jantung pengendara dan kemudian disimpan, lalu nilai detak jantung yang diterima ditampilkan secara *real-time* pula oleh aplikasi, kemudian aplikasi akan selalu memeriksa apakah nilai detak jantung yang diterima kurang dari atau sama dengan nilai *threshold*, jika tidak maka aplikasi akan terus menampilkan nilai detak jantung, namun jika ya, maka aplikasi akan secara otomatis mendapatkan lokasi dan mengirimkan pesan darurat yang disertai lokasi pengendara ke nomor darurat.

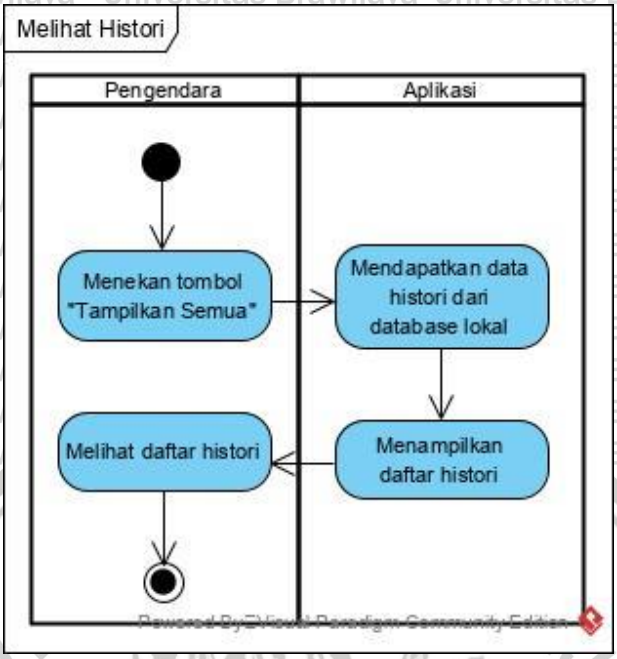
5.1.1.3 Menyimpan Nomor Darurat (APK-F-003)



Gambar 5.3 *Activity Diagram* Menyimpan Nomor Darurat

Alur aktivitas menyimpan nomor darurat yang digambarkan pada Gambar 5.3 diawali ketika pengendara menekan tombol "Tambah Kontak" pada aplikasi. Kemudian aplikasi menampilkan *form* kontak darurat yang terdiri dari nama dan nomor telepon, lalu pengendara mengisi *form* tersebut dan menekan tombol "Simpan". Jika nomor telah terdaftar sebelumnya, aplikasi akan menampilkan pesan gagal, namun jika nomor belum terdaftar maka aplikasi akan menyimpan kontak darurat di *database* lokal.

5.1.1.4 Melihat Histori (APK-F-004)



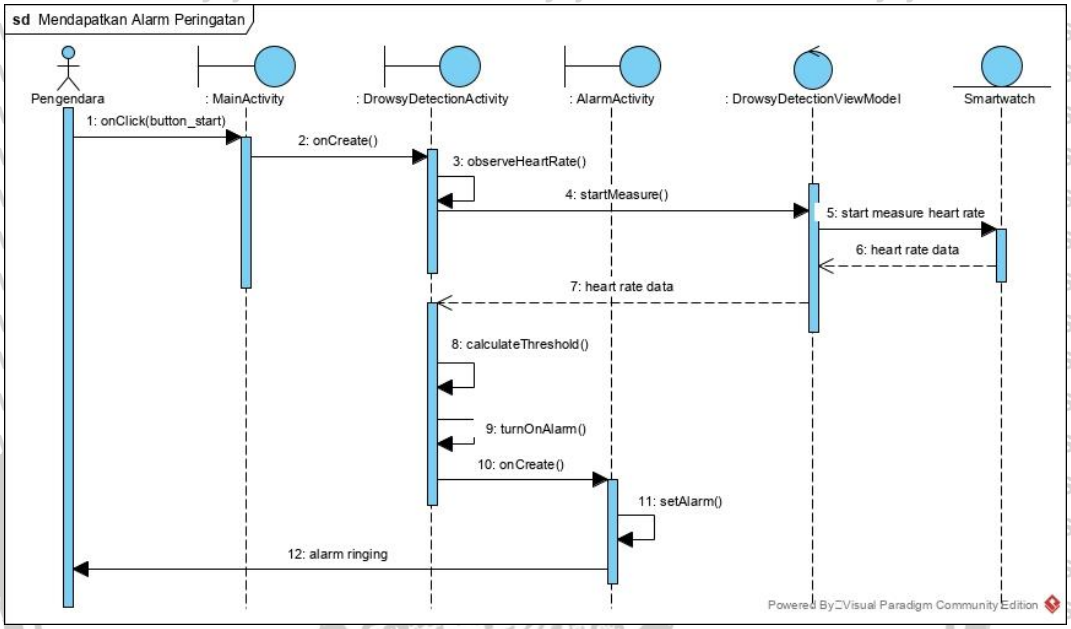
Gambar 5.4 *Activity Diagram* Melihat Histori

Gambar 5.4 menunjukkan alur dari aktivitas melihat histori yang mana dimulai ketika pengendara menekan tombol "Tampilkan Semua", lalu aplikasi akan mendapatkan data daftar histori dari *database* lokal yang kemudian akan ditampilkan dalam bentuk *list*, sehingga pengendara dapat melihat daftar histori pendeteksian kantuk.

5.1.2 *Sequence Diagram*

Sequence diagram adalah diagram yang menggambarkan urutan interaksi antar objek dalam sistem. *Sequence diagram* dari aplikasi pendeteksi kantuk berdasarkan *use case scenario* terdapat pada subbab 5.1.2.1 sampai dengan 5.1.2.4.

5.1.2.1 Mendapatkan Alarm Peringatan (APK-F-001)



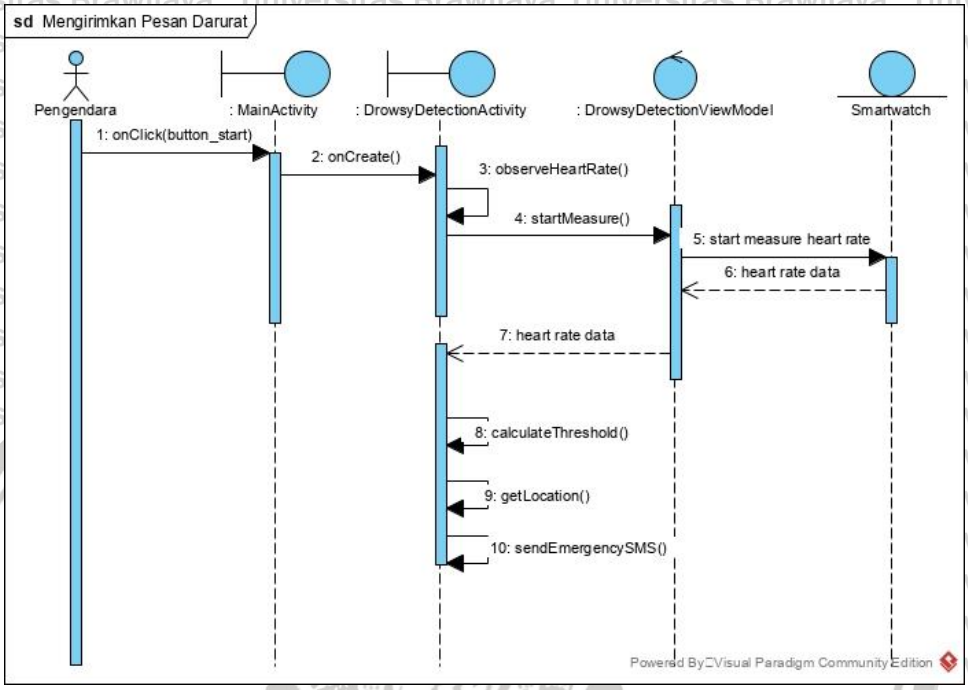
Gambar 5.5 *Sequence Diagram* Mendapatkan Alarm Peringatan

Urutan aktivitas pada diagram di atas adalah sebagai berikut.

1. Pengendara menekan tombol "Mulai" pada MainActivity.
2. Aplikasi akan membuka *activity* baru yaitu DrowsyDetectionActivity.
3. DrowsyDetectionActivity memanggil *method* observeHeartRate() untuk mengobservasi data detak jantung.
4. DrowsyDetectionActivity memanggil *method* startMeasure() pada DrowsyDetectionViewModel yang berfungsi untuk memulai pengukuran detak jantung.
5. DrowsyDetectionViewModel memerintahkan *smartwatch* untuk memulai pengukuran detak jantung.
6. Data detak jantung dikembalikan ke DrowsyDetectionViewModel.
7. Data detak jantung dikembalikan ke DrowsyDetectionActivity.
8. DrowsyDetectionActivity memanggil *method* calculateThreshold() untuk menghitung nilai *threshold* detak jantung.
9. DrowsyDetectionActivity memanggil *method* turnOnAlarm() untuk menyalakan alarm.

10. Aplikasi membuka *activity* baru yaitu AlarmActivity.
11. AlarmActivity memanggil *method* setAlarm() untuk membunyikan alarm.
12. Alarm berbunyi, sehingga peringatan didapatkan oleh aktor.

5.1.2.2 Mengirimkan Pesan Darurat (APK-F-002)



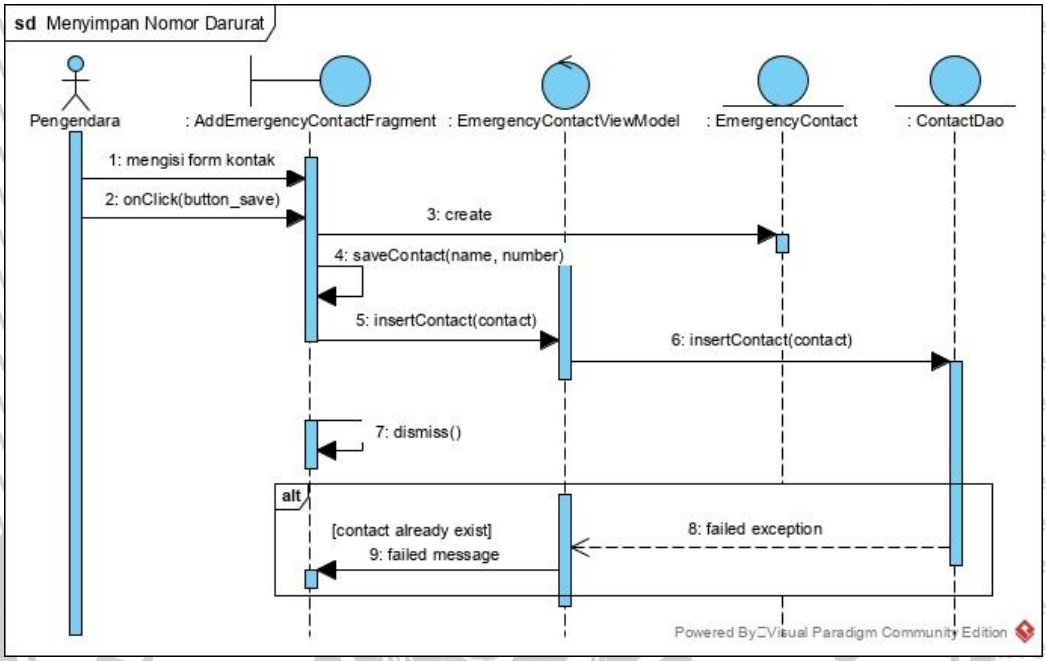
Gambar 5.6 *Sequence Diagram* Mengirimkan Pesan Darurat

Urutan aktivitas pada diagram di atas adalah sebagai berikut.

1. Pengendara menekan tombol "Mulai" pada MainActivity.
2. Aplikasi akan membuka *activity* baru yaitu DrowsyDetectionActivity.
3. DrowsyDetectionActivity memanggil *method* observeHeartRate() untuk mengobservasi data detak jantung.
4. DrowsyDetectionActivity memanggil *method* startMeasure() pada DrowsyDetectionViewModel yang berfungsi untuk memulai pengukuran detak jantung.
5. DrowsyDetectionViewModel memerintahkan *smartwatch* untuk memulai pengukuran detak jantung.
6. Data detak jantung dikembalikan ke DrowsyDetectionViewModel.
7. Data detak jantung dikembalikan ke DrowsyDetectionActivity.
8. DrowsyDetectionActivity memanggil *method* calculateThreshold() untuk menghitung nilai *threshold* detak jantung.
9. DrowsyDetectionActivity memanggil *method* getLocation() untuk mendapatkan lokasi.

10. DrowsyDetectionActivity memanggil *method* sendEmergencySMS() untuk mengirimkan pesan darurat ke nomor darurat.

5.1.2.3 Menyimpan Nomor Darurat (APK-F-003)

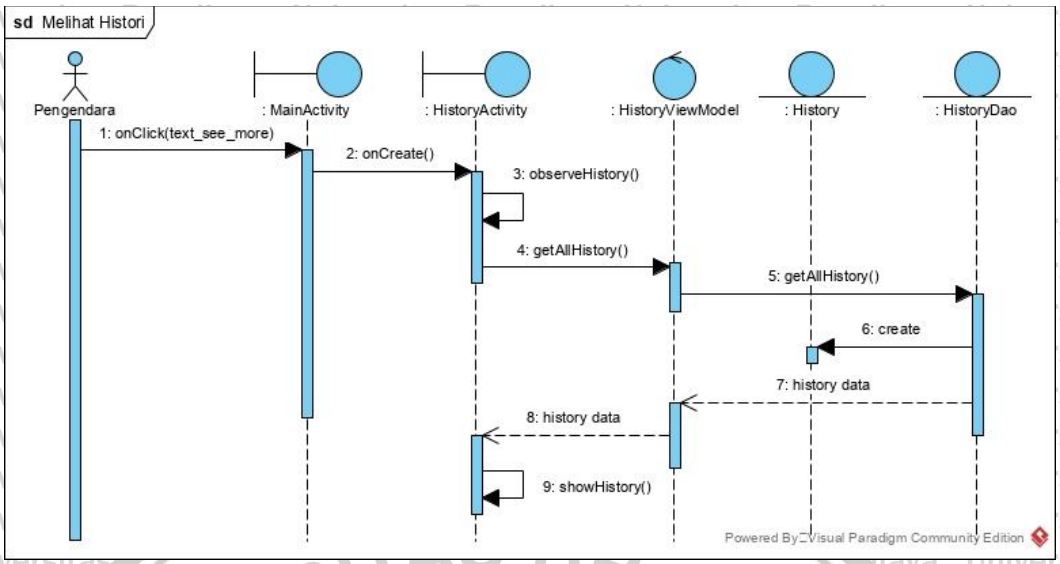


Gambar 5.7 *Sequence Diagram* Menyimpan Nomor Darurat

Urutan aktivitas pada diagram di atas adalah sebagai berikut.

1. Pengendara mengisi *form* yang terdiri dari nama dan nomor telepon pada AddEmergencyContactActivity.
2. Pengendara menekan tombol "Simpan" AddEmergencyContactActivity.
3. AddEmergencyContactFragment membuat objek EmergencyContact.
4. AddEmergencyContactFragment memanggil *method* saveContact().
5. AddEmergencyContactFragment memanggil *method* insertContact() pada EmergencyContactViewModel.
6. EmergencyContactViewModel memanggil *method* insertContact() dari EmergencyContactDao untuk menyimpan kontak darurat di penyimpanan lokal.
7. AddEmergencyContactFragment memanggil *method* dismiss() untuk menutup halaman.
8. Jika kontak gagal disimpan, maka pesan gagal akan dikembalikan ke EmerencyContactViewModel.
9. EmergencyContactViewModel menampilkan pesan kesalahan.

5.1.2.4 Melihat Histori (APK-F-004)



Gambar 5.8 *Sequence Diagram* Melihat Histori

Urutan aktivitas pada diagram di atas adalah sebagai berikut.

1. Pengendara menekan teks "Lihat Semua" pada bagian Histori di MainActivity.
2. Aplikasi membuka *activity* baru yaitu HistoryActivity.
3. HistoryActivity memanggil *method* observeHistory() untuk mengobservasi data histori.
4. HistoryActivity memanggil *method* getAllHistory() pada HistoryViewModel.
5. HistoryViewModel memanggil *method* getAllHistory() pada HistoryDao untuk mendapatkan seluruh data histori yang tersimpan di penyimpanan lokal.
6. HistoryDao membuat objek History.
7. Data histori dikembalikan ke HistoryViewModel.
8. Data histori dikembalikan ke HistoryActivity.
9. HistoryActivity memanggil *method* showHistory() untuk menampilkan data histori dalam bentuk *list*.

5.1.3 Rancangan Algoritma Mendeteksi Kantuk

Untuk dapat mendeteksi kondisi kantuk pengendara, terdapat dua tahap yang dijalankan oleh aplikasi pendeteksi kantuk, yaitu menghitung nilai *threshold* detak jantung pengendara dan mendeteksi kantuk dengan cara membandingkan nilai detak jantuk terkini dengan *threshold*. Algoritma dari kedua tahapan tersebut tertera pada Tabel 5.1 dan Tabel 5.2.

Tabel 5.1 *Pseudocode* Menghitung *Threshold*

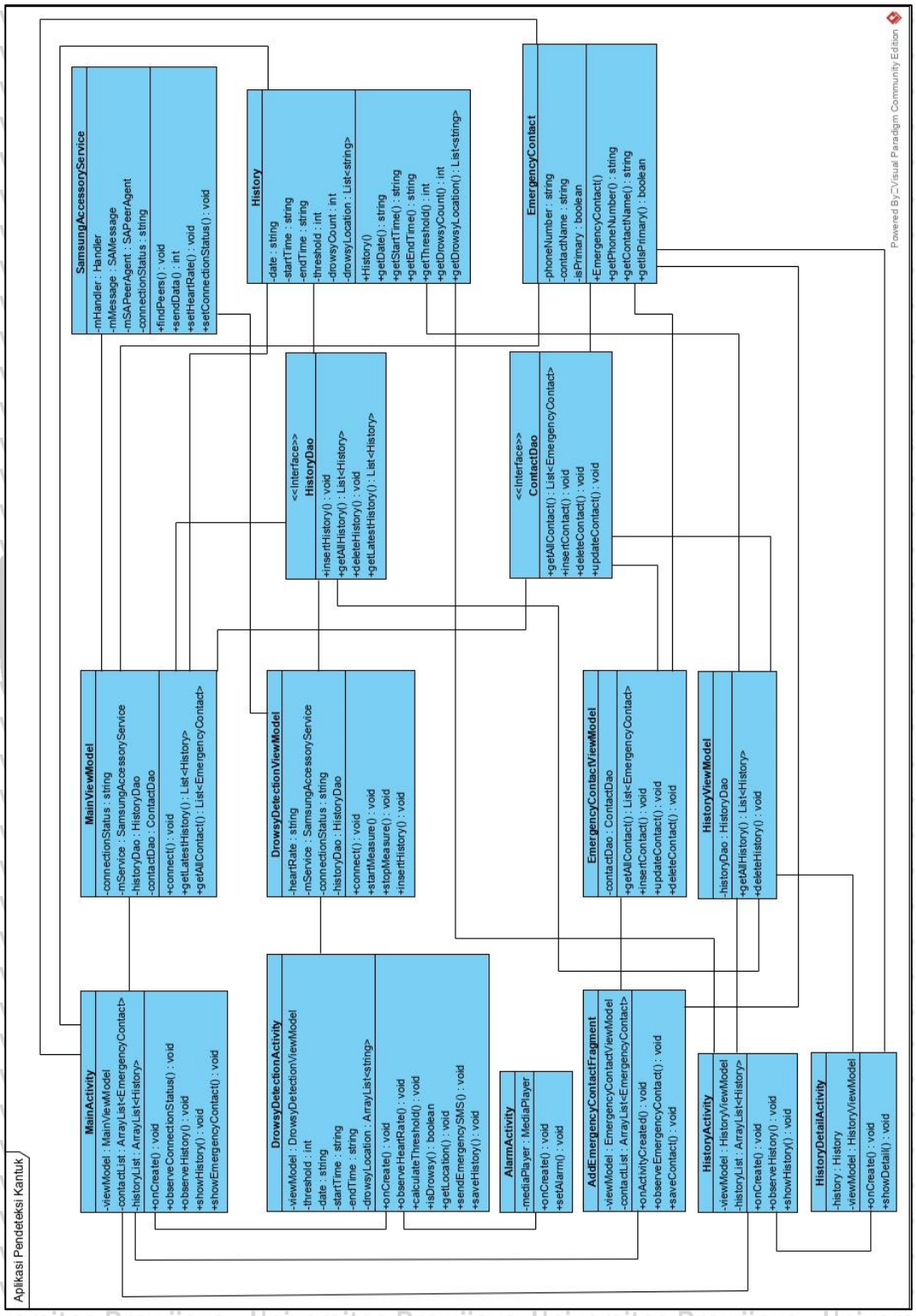
Algoritma Menghitung Threshold	
Input : *heart rate array* `heartRateArr`	
Output : *threshold value*	
1	`totalHeartRate ← 0`
2	`for i ← 0 to i ← length(heartRateArr)`
3	`totalHeartRate = totalHeartRate + heartRateArr[i]`
4	`endfor`
5	`heartRateMean ← totalHeartRate / length(heartRateArr)`
6	`threshold ← heartRateMean - 8`
7	`return threshold`

Tabel 5.2 *Pseudocode* Mendeteksi Kantuk

Algoritma Mendeteksi Kantuk	
Input : *current heart rate* `heartRate`	
Output : *drowsy or not (true/false)*	
1	`if heartRate <= threshold`
2	`  return true`
3	`endif`
4	`return false`

5.1.4 *Class Diagram*

Class diagram merupakan diagram yang mendeskripsikan struktur internal dari sistem yang akan dibangun dengan menggambarkan kelas-kelas dari sistem beserta dengan properti, *method*, dan hubungannya. *Class diagram* aplikasi pendeteksi kantuk tertera pada Gambar 5.9.



Gambar 5.9 *Class Diagram* Aplikasi Pendeteksi Kantuk

Tabel 5.3 Keterangan *Class Diagram*

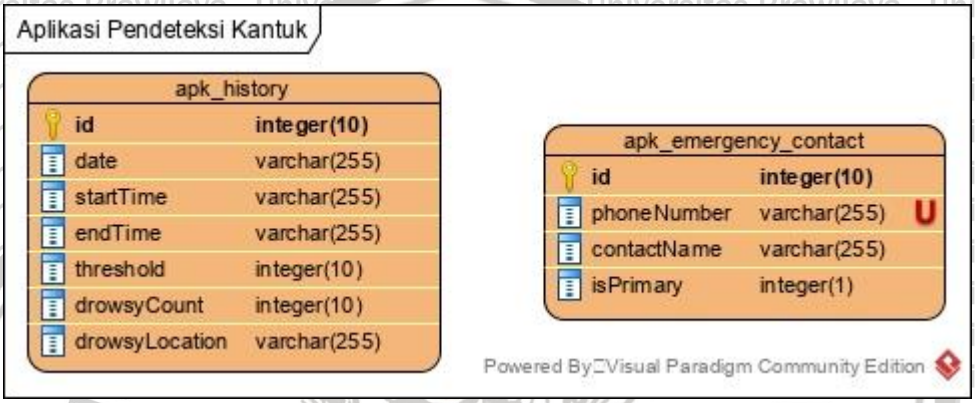
No.	*Use Case*	Kelas yang Terlibat	*Responsibility*
1	Mendapatkan Alarm Peringatan	DrowsyDetectionActivity	Sebagai *user interface* dan terdapat *method* yang dijalankan saat pendeteksian kantuk, diantaranya calculateThreshold() untuk menghitung *threshold,* observeHeartRate() untuk mengobservasi nilai detak jantung, *dan* isDrowsy() untuk menentukan kondisi kantuk.
		DrowsyDetectionViewModel	Menghubungkan *activity* dengan *service* untuk mendapatkan data detak jantung dari *smartwatch. Method* yang terlibat yaitu startMeasure() dan stopMeasure() untuk memulai dan menghentikan pengukuran detak jantung.
2	Mengirimkan Pesan Darurat	DrowsyDetectionActivity	Sebagai *user interface* dan terdapat *method* yang dijalankan saat mengirimkan pesan darurat, diantaranya getLocation() untuk mendapatkan lokasi dan sendEmergencySMS() untuk mengirimkan SMS ke kontak darurat.

		EmergencyContact	Objek yang menampung data kontak darurat.
3	Menyimpan Nomor Darurat	AddEmergencyContactFragment	Sebagai *user interface* dan terdapat *method* yang dijalankan saat menyimpan kontak, yaitu saveContact().
		EmergencyContactViewModel	Menghubungkan *activity* objek *database* lokal, *method* yang terlibat yaitu insertContact() untuk menyimpan kontak dan updateContact() untuk memperbaharui kontak.
		ContactDao	*Interface Database Object* yang berisi *method* untuk menjalankan *query* SQLite.
		EmergencyContact	Objek yang menampung data kontak darurat.
4	Melihat Histori	HistoryActivity	Sebagai *user interface.*
		HistoryViewModel	Menghubungkan *activity* objek *database* lokal, *method* yang terlibat yaitu getAllHistory() untuk mendapatkan seluruh histori dari *database* lokal.
		HistoryDao	*Interface Database Object* yang berisi *method* untuk menjalankan *query* SQLite.

		History	Objek yang menampung data histori.

5.2 Perancangan *Database*

Aplikasi pendeteksi kantuk menggunakan *database* lokal untuk menyimpan data histori dan nomor darurat. Terdapat dua tabel yaitu tabel apk_history dan apk_emergency_contact. Kedua tabel tersebut tidak bergantung satu sama lain sehingga tidak saling berelasi. Struktur dari *database* terdapat pada Gambar 5.10.



Gambar 5.10 *Entity Relationship Diagram Database* Aplikasi Pendeteksi Kantuk

Tabel apk_history memiliki tujuh kolom, diantaranya :

- id : ID histori.
- date : tanggal pendeteksian kantuk.
- startTime : waktu mulai pendeteksian kantuk.
- endTime : waktu selesai pendeteksian kantuk.
- threshold : nilai *threshold* detak jantung.
- drowsyCount : jumlah terdeteksi kantuk.
- drowsyLocation : lokasi terdeteksi kantuk.

Sedangkan tabel apk_emergency_contact terdiri dari tiga kolom, diantaranya :

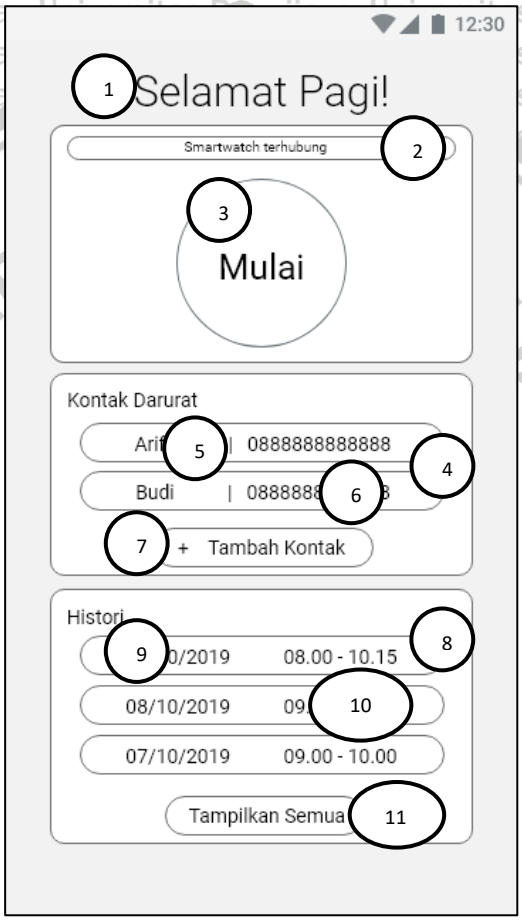
- id : ID kontak darurat.
- phoneNumber : nomor telepon darurat.
- contactName : nama kontak darurat.
- isPrimary : menentukan apakah kontak darurat sebagai kontak utama.

5.3 Perancangan Antarmuka

Pada tahap perancangan antarmuka dilakukan pemodelan tampilan antarmuka aplikasi dalam bentuk *wireframe* berdasarkan *use case scenario.*

5.3.1 Halaman Utama

Gambar 5.11 menunjukkan tampilan dari halaman utama yang merupakan halaman paling awal saat aplikasi dibuka. Terdapat tiga bagian pada halaman utama, yaitu menu untuk memulai pendeteksian kantuk, menu kontak darurat, dan histori. Keterangan dari *wireframe* dapat dilihat pada Tabel 5.4.



Gambar 5.11 *Wireframe* Antarmuka Halaman Utama

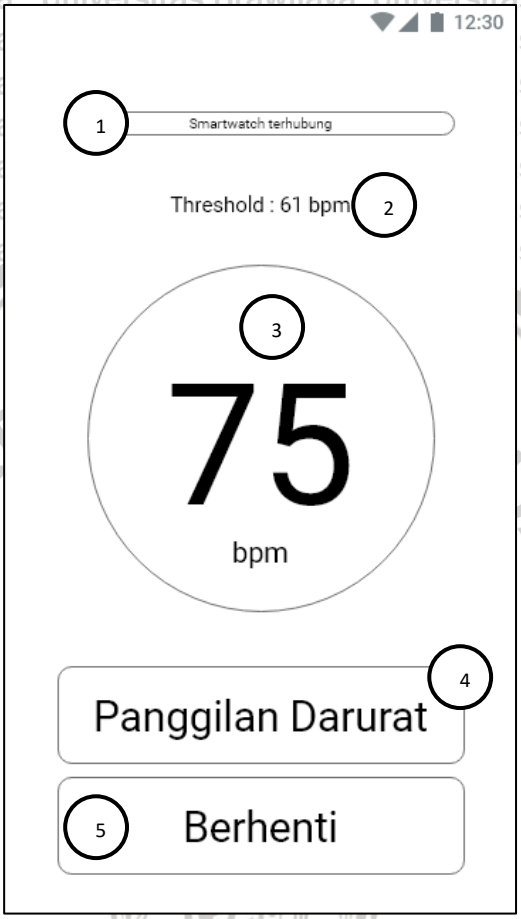
Tabel 5.4 Keterangan Antarmuka Halaman Utama

No.	Nama Objek	Tipe	Keterangan
1	*Greetings*	*TextView*	Pesan sapaan kepada pengguna.
2	Status Koneksi	*TextView*	Status koneksi aplikasi dengan *smartwatch.*

3	Tombol Mulai	*Button*	Tombol untuk memulai pendeteksian kantuk.
4	Daftar Kontak Darurat	*RecyclerView*	Menampilkan daftar kontak darurat yang tersimpan dalam bentuk *list.*
5	Nama Kontak	*TextView*	Nama kontak darurat yang tersimpan.
6	Nomor Telepon Kontak	*TextView*	Nomor telepon kontak yang tersimpan.
7	Tombol Tambah Kontak	*Button*	Tombol untuk menambah kontak darurat.
8	Daftar Histori	*RecyclerView*	Menampilkan daftar histori dalam bentuk *list.*
9	Tanggal Deteksi	*TextView*	Tanggal pendeteksian kantuk.
10	Waktu Deteksi	*TextView*	Waktu pendeteksian kantuk.
11	Tombol Tampilkan Semua Histori	*Button*	Tombol untuk menampilkan semua histori.

5.3.2 Halaman Pendeteksian Kantuk

Pada fungsi mendeteksi kantuk, aplikasi menampilkan status koneksi, detak jantung, nilai *threshold*, serta menu untuk melakukan panggilan darurat dan menghentikan pendeteksian. *Wireframe* antarmuka mendeteksi kantuk tertera pada Gambar 5.12. Penjelasan mengenai *wireframe* dapat dilihat pada Tabel 5.5.



Gambar 5.12 *Wireframe* Antarmuka Halaman Pendeteksian Kantuk

Tabel 5.5 Keterangan Antarmuka Halaman Pendeteksian Kantuk

No.	Nama Objek	Tipe	Keterangan
1	Status Koneksi	*TextView*	Status koneksi aplikasi dengan *smartwatch.*
2	*Threshold*	*TextView*	Nilai *threshold* detak jantung pengguna berdasarkan hasil pengukuran.
3	Detak Jantung	*TextView*	Nilai detak jantung pengguna secara *real time.*
4	Tombol Panggilan Darurat	*Button*	Tombol untuk melakukan panggilan ke kontak darurat.

5	Tombol Berhenti	*Button*	Tombol untuk menghentikan pendeteksian kantuk.

5.3.3 Halaman Alarm Peringatan

Gambar 5.13 menunjukkan *wireframe* antarmuka aplikasi saat menjalankan fungsi menyalakan alarm. Yang ditampilkan pada fungsi ini adalah pesan peringatan serta menu untuk melakukan panggilan darurat dan mengabaikan peringatan. Keterangan antarmuka dapat dilihat pada Tabel 5.6.

Gambar 5.13 *Wireframe* Antarmuka Mendapatkan Alarm Peringatan

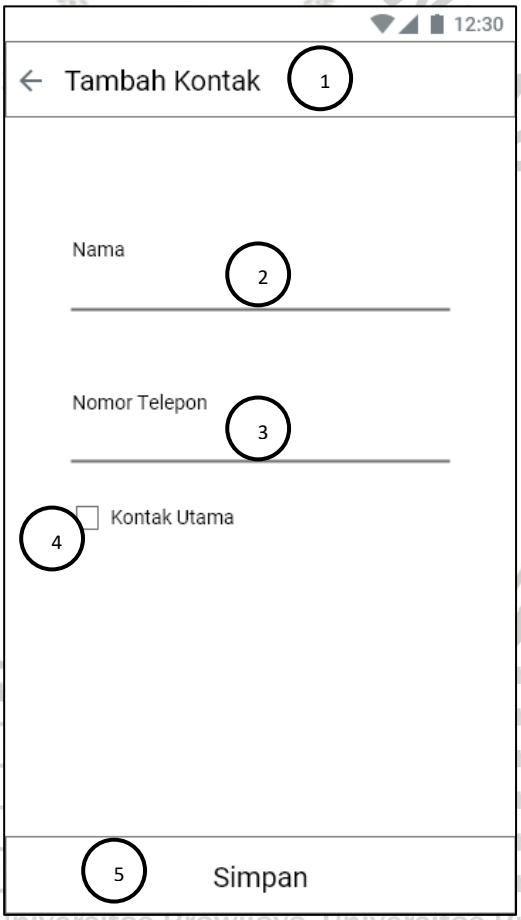
Tabel 5.6 Keterangan Antarmuka Mendapatkan Alarm Peringatan

No.	Nama Objek	Tipe	Keterangan
1	Pesan Hati - Hati	*TextView*	Pesan peringatan "Hati - Hati" untuk pengguna.
2	*Icon Warning*	*ImageView*	Gambar *icon* peringatan.
3	Pesan Mengantuk	*TextView*	Pesan peringatan bahwa pengguna terdeteksi mengantuk.

4	Pesan Notifikasi Darurat	*TextView*	Memberitahu pengguna bahwa pesan darurat akan dikirimkan ke nomor darurat yang tersimpan.
5	Tombol Panggulan Darurat	*Button*	Tombol untuk melakukan panggilan darurat.
6	Tombol Abaikan	*Button*	Tombol untuk mengabaikan peringatan.

5.3.4 Halaman Tambah Kontak Darurat

Gambar 5.14 menunjukkan *wireframe* antarmuka fungsi menyimpan nomor darurat. Pada fungsi ini, aplikasi menampilkan form yang terdiri dari nama dan nomor telepon dari kontak darurat yang hendak disimpan. Keterangan mengenai *wireframe* dapat dilihat pada Tabel 5.7.



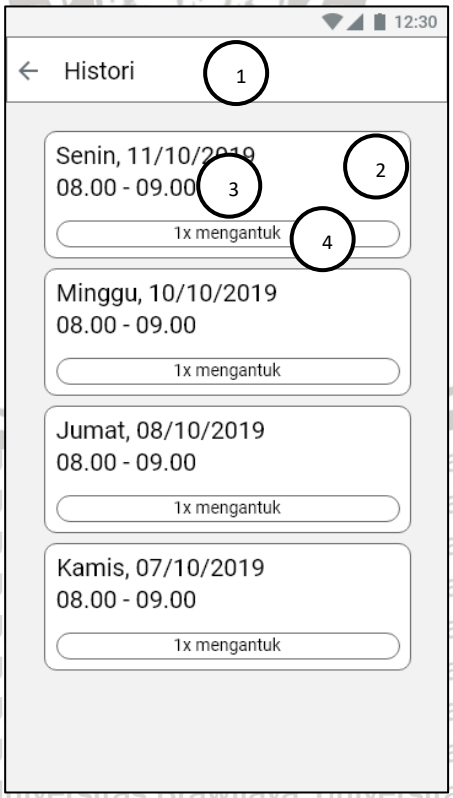
Gambar 5.14 *Wireframe* Antarmuka Menyimpan Nomor Darurat

Tabel 5.7 Keterangan Antarmuka Menyimpan Nomor Darurat

No.	Nama Objek	Tipe	Keterangan
1	*Toolbar* Tambah Kontak	*Toolbar*	*Toolbar* dengan judul "Tambah Kontak".
2	*Form* Nama Kontak	*EditText*	*Form* untuk mengisikan nama kontak yang akan disimpan.
3	*Form* Nomor Telepon	*EditText*	*Form* untuk mengisikan nomor telepon kontak.
4	*Checkbox* Kontak Utama	*CheckBox*	*Checkbox* untuk menyimpan kontak sebagai kontak darurat utama.
5	Tombol Simpan	*Button*	Tombol untuk menyimpan kontak.

5.3.5 Halaman Histori

Pada fungsi menampilkan histori, aplikasi menampilkan daftar histori dalam bentuk *list* yang memuat tanggal, waktu, dan jumlah berapa kali pengguna mengantuk. *Wireframe* antarmuka menampilkan histori tertera pada Gambar 5.15 dan keterangannya terdapat pada Tabel 5.8.



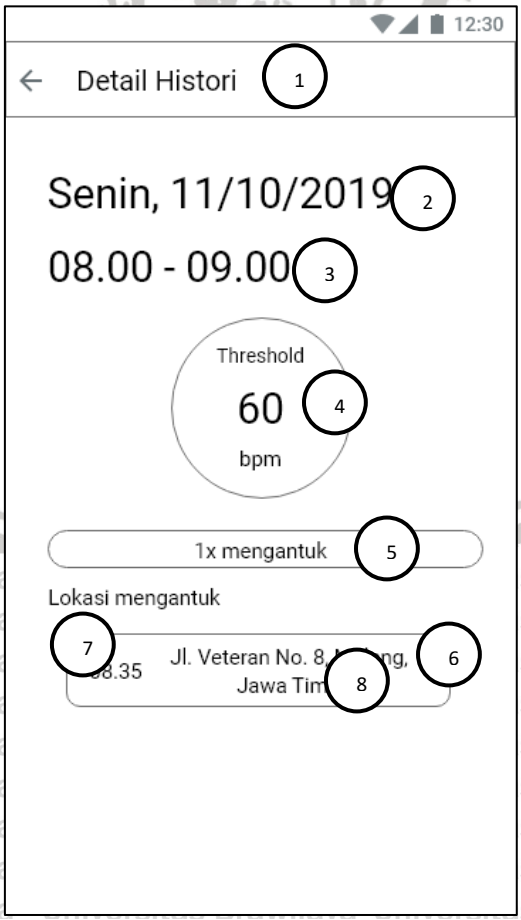
Gambar 5.15 *Wireframe* Antarmuka Menampilkan Histori

Tabel 5.8 Keterangan Antarmuka Menampilkan Histori

No.	Nama Objek	Tipe	Keterangan
1	*Toolbar* Histori	*Toolbar*	*Toolbar* dengan judul "Histori".
2	Daftar Histori	*RecyclerView*	Daftar histori dalam bentuk *list*.
3	Tanggal dan Waktu	*TextView*	Tanggal dan waktu pendeteksian.
4	Jumlah Mengantuk	*TextView*	Jumlah pengguna mengantuk dalam pendeteksian.

5.3.6 Halaman Detail Histori

Halaman detail histori menampilkan keseluruhan informasi dari suatu histori pendeteksian kantuk. Selain tanggal, waktu, dan jumlah kantuk, pada halaman ini juga menampilkan nilai *threshold* dan lokasi pengguna terdeteksi mengantuk. *Wireframe* dari halaman detail histori digambarkan pada Gambar 5.16 dan penjelasannya dijabarkan pada Tabel 5.9.



Gambar 5.16 *Wireframe* Antarmuka Detail Histori

Tabel 5.9 Keterangan Antarmuka Detail Histori

No.	Nama Objek	Tipe	Keterangan
1	*Toolbar* Detail Histori	*Toolbar*	*Toolbar* dengan judul "Detail Histori".
2	Tanggal Deteksi	*TextView*	Tanggal pendeteksian kantuk.
3	Waktu Deteksi	*TextView*	Waktu pendeteksian kantuk.
4	*Threshold*	*TextView*	Nilai *threshold* detak jantung pengguna.
5	Jumlah Mengantuk.	*TextView*	Jumlah kantuk pengguna dalam pendeteksian.
6	Daftar Lokasi Mengantuk	*RecyclerView*	Daftar lokasi pengguna terdeteksi mengantuk.
7	Waktu Mengantuk	*TextView*	Waktu pengguna terdeteksi mengantuk.
8	Nama Lokasi Mengantuk	*TextView*	Alamat lokasi pengguna terdeteksi mengantuk.

BAB 6 IMPLEMENTASI

Pada bab ini membahas tahapan implementasi yang dilakukan dalam pengembangan aplikasi pendeteksi kantuk. Bab Implementasi terbagi ke dalam empat subbab diantaranya Spesifikasi Lingkungan Aplikasi, Batasan Implementasi, Implementasi Kode Program, dan Implementasi Antarmuka.

6.1 Spesifikasi Lingkungan Aplikasi

Spesifikasi lingkungan aplikasi menjelaskan spesifikasi perangkat yang digunakan dalam pengembangan aplikasi pendeteksi kantuk. Spesifikasi lingkungan aplikasi terbagi menjadi dua yaitu spesifikasi perangkat keras dan spesifikasi perangkat lunak.

6.1.1 Spesifikasi Perangkat Keras

Terdapat tiga jenis perangkat keras yang digunakan dalam pengembangan aplikasi pendeteksi kantuk, diantaranya komputer, *smartphone* , dan *smartwatch.* Spesifikasi untuk masing-masing perangkat dapat dilihat pada Tabel 6.1, Tabel 6.2, dan Tabel 6.3.

Tabel 6.1 Spesifikasi Perangkat Keras Komputer

Komponen	**Spesifikasi**
Laptop	DELL Vostro 14-5480
Processor	Intel Core i7-5500U
Memory	8 GB

Tabel 6.2 Spesifikasi Perangkat Keras *Smartphone*

Komponen	**Spesifikasi**
Smartphone	Xiaomi Redmi Note 4X
Processor	Qualcomm Snapdragon 625
Memory	4 GB

Tabel 6.3 Spesifikasi Perangkat Keras *Smartwatch*

Komponen	**Spesifikasi**
Smartwatch	Samsung Gear S2 SM-R720
Processor	Exynos 3250
Memory	512 MB

6.1.2 Spesifikasi Perangkat Lunak

Perangkat lunak yang digunakan dalam pengembangan aplikasi pendeteksi kantuk memiliki spesifikasi seperti yang tertera pada Tabel 6.4.

Tabel 6.4 Spesifikasi Perangkat Lunak

Komponen	Spesifikasi
Operating System Laptop	Windows 10 64-bit
Operating System Smartphone	Android 7.0 Nougat
Operating System Smartwatch	Tizen 2.3
Bahasa Pemrograman	Kotlin
Tools Development	Android Studio

6.2 Batasan Implementasi

Dalam implementasi aplikasi pendeteksi kantuk terdapat sejumlah batasan, diantaranya :

1. Aplikasi pendeteksi kantuk hanya dapat diimplementasikan di perangkat bersistem operasi Android dengan versi minimal 4.1 *Ice Cream Sandwich.*
2. Aplikasi pendeteksi kantuk hanya dapat diimplementasikan dengan *smartwatch* Samsung dengan sistem operasi Tizen dengan versi minimal 2.3.

6.3 Implementasi *Database*

Dalam mengimplementasikan *database* lokal pada perangkat Android, penulis menggunakan *library* Room. Room merupakan *library* yang berjalan diatas SQLite. Room terdiri dari tiga komponen, diantaranya kelas *Room Database* yang merepresentasikan objek *database, interface Data Access Object* (Dao) yang berisi kumpulan *method* untuk mengakses *database*, dan kelas *Entity* yang merepresentasikan tabel dalam *database*. Implementasi *database* dijelaskan pada

Tabel 6.5 Implementasi Kelas *Room Database* APKDatabase.kt

APKDatabase.kt

```
1  @Database(entities = [History::class, EmergencyContact::class],
   version = 1)
2
3  abstract class APKDatabase : RoomDatabase() {
4
   companion object {
```

```
5     @Volatile
6     private var INSTANCE: APKDatabase? = null
7     fun getInstance(context: Context): APKDatabase {
8         return INSTANCE ?: synchronized(this) {
9             val instance = Room.databaseBuilder(
10            context.applicationContext,          APKDatabase::class.java,
"aplikasipendeteksikantuk_db")
11                        .build()
12            INSTANCE = instance
13            instance
14            }
15        }
16    }
17        abstract fun getHistoryDao(): HistoryDao
18        abstract fun getContactDao(): ContactDao
19    }
20
```

Pada kelas APKDatabase terdapat *method* getInstance() untuk mengembalikan *instance Room Database*. Selain itu terdapat *abstract method* getHistoryDao() yang mengembalikan *instance Data Access Object* HistoryDao, serta *method* getContactDao() untuk mengembalikan *instance* ContactDao.

Tabel 6.6 Implementasi *Interface Data Access Object* HistoryDao.kt

HistoryDao.kt

```
1     @Dao
2     interface HistoryDao {
3         @Insert
4         suspend fun insertHistory(history: History)
5
6         @Query("SELECT * FROM apk_history")
7         fun getAllHistory(): LiveData<List<History>>
8
9         @Query("SELECT * FROM apk_history ORDER BY id DESC LIMIT 3")
10        fun getLatestHistory(): LiveData<List<History>>
11
12        @Delete
13        suspend fun deleteHistory(history: History)
14
15    }
```

Pada *interface* HistoryDao berisi *method-method* yang dibutuhkan untuk memanipulasi data histori, diantaranya insertHistory() untuk memasukkan data histori, getAllHistory() untuk mendapatkan seluruh data histori, getLatestHistory()

untuk mendapatkan tiga data terakhir, dan deleteHistory() untuk menghapus data histori.

Tabel 6.7 Implementasi *Interface Data Access Object* ContactDao.kt

ContactDao.kt

```
1   @Dao
2   interface ContactDao {
3     @Insert
4     suspend fun insertContact(contact: EmergencyContact)
5
6     @Query("SELECT * FROM apk_emergency_contact")
7     fun getAllContact(): LiveData<List<EmergencyContact>>
8
9     @Delete
10    suspend fun deleteContact(contact: EmergencyContact)
11
12    @Update
13    suspend fun updateContact(contact: EmergencyContact)
14  }
15
```

Pada *interface* ContactDao berisi *method-method* untuk memanipulasi data kontak darurat, diantaranya insertContact() untuk memasukkan data kontak, getAllContact() untuk mendapatkan seluruh data kontak, deleteContact() untuk menghapus data kontak, dan updateContact() untuk memperbaharui data kontak.

Tabel 6.8 Implementasi Kelas *Entity* History.kt

History.kt

```
1   @Entity(tableName = "apk_history")
2   data class History(
3     @PrimaryKey(autoGenerate = true) var id: Int = 0,
4     var date: String = "",
5     var startTime: String = "",
6     var endTime: String = "",
7     var threshold: Int = 0,
8     var drowsyCount: Int = 0,
9     var drowsyLocation: List<String> = listOf()
10  )
```

Kelas *entity* History merepresentasikan tabel apk_history, sehingga memiliki atribut id, date, startTime, endTime, threshold, drowsyCount, dan drowsyLocation.

Tabel 6.9 Implementasi Kelas *Entity* EmergencyContact.kt

EmergencyContact.kt

```
1  @Entity(tableName = "apk_emergency_contact", indices = [Index(
2      value = ["phoneNumber"],
3      unique = true
4  )]
5  )
6  data class EmergencyContact(
7      @PrimaryKey(autoGenerate = true) var id: Int = 0,
8      var phoneNumber: String = "",
9      var contactName: String = "",
10     var isPrimary: Int = 0
11 )
```

Kelas *entity* EmergencyContact merepresentasikan tabel apk_emergency_contact, sehingga memiliki atribut ide, phoneNumber, contactName, dan isPrimary.

6.4 Implementasi Kode Program

Tahap implementasi kode program menjelaskan hasil implementasi sistem dari sisi kode program berdasarkan rancangan yang telah dibuat untuk masing-masing fungsional.

6.4.1 Halaman Utama

Halaman utama merupakan halaman yang pertama kali ditampilkan ketika aplikasi dibuka. Halaman utama berfungsi mengarahkan pengguna ke fungsi-fungsi yang tersedia. Implementasi kode program dari halaman utama terdapat pada Tabel 6.10.

Tabel 6.10 Implementasi Kode Program MainActivity.kt

MainActivity.kt

```
1  class MainActivity : AppCompatActivity() {
2  override fun onCreate(savedInstanceState: Bundle?) {
3      super.onCreate(savedInstanceState)
4      setContentView(R.layout.activity_main)
5
6      button_mulai.setOnClickListener {
7          var isPrimaryContactAvailable = false
8          for (i in contactList.indices){
9              if (contactList[i].isPrimary == 1){
```

```
10              isPrimaryContactAvailable = true
11              break
12          }
13      }
14      if (button_mulai.text == "Mulai") {
15          if (isPrimaryContactAvailable){
16              val intent = Intent(this,
DrowsyDetectionActivity::class.java)
17              DrowsyDetectionActivity.contactList = contactList
18              startActivity(intent)
19              finish()
20              }
21          else{
22              Toast.makeText(applicationContext, "Mohon pilih kontak
darurat utama!", Toast.LENGTH_SHORT).show()
23          }
24      }
25      else {
26          viewModel.connect()
27      }
28  }
29  button_add_emergency_contact.setOnClickListener {
30      val addContactFragment =
31      AddEmergencyContactFragment()
32      addContactFragment.show(supportFragmentManager,
"AddEmergencyContactFragment")
33  }
34  text_view_see_more.setOnClickListener {
35      startActivity(Intent(this, HistoryActivity::class.java))
36  }
37  }
38  }
39  }
40  }
```

Penjelasan kode program pada Tabel 6.10 adalah sebagai berikut.

- Baris ke 6-29 mendefinisikan perintah yang akan dijalankan ketika tombol “Mulai” ditekan.
- Baris ke 30-35 perintah untuk menampilkan halaman tambah kontak ketika tombol “Tambah Kontak” ditekan.
- Baris ke 36-38 perintah untuk mengarahkan ke halaman daftar histori ketika teks “Lihat semua” ditekan.

6.4.2 Mendapatkan Alarm Peringatan (APK-F-001)

Implementasi kode program untuk fungsi mendapatkan alarm peringatan dijabarkan pada Tabel 6.11 sampai dengan Tabel 6.13.

Tabel 6.11 Implementasi Kode Program DrowsyDetectionActivity.kt (Mendapatkan Alarm Peringatan)

DrowsyDetectionActivity.kt

```
1  class DrowsyDetectionActivity : AppCompatActivity(){
2  override fun onCreate(savedInstanceState: Bundle?) {
3  super.onCreate(savedInstanceState)
4  setContentView(R.layout.activity_drowsy_detection)
5  if (!isStarted) {
6    viewModel.startMeasure()
7  }
8    progress_bar_detection.visibility = View.VISIBLE
9    observeHeartRate()
10 }
11
12 private fun observeHeartRate() {
13   val heartRateObserver = Observer<String> { heartRate ->
14   if (heartRate != null) {
15     isStarted = true
16     text_view_heart_rate.text = heartRate
17     if (!isCalculatingStarted) {
18       calculateThreshold()
19     }
20     if (!isCalculateTimeOver) {
21       heartRateForThreshold.add(heartRate.toInt())
22     }
23     if (threshold != 0) {
24       if (isDrowsy(heartRate.toInt()) && !isAlarmActive &&
!isPaused) {
25         drowsyCount++
26         turnOnAlarm()
27       }
28     }
29   }
30   viewModel.heartRate.observe(this, heartRateObserver)
31 }
32 private fun calculateThreshold() {
33
```

```
34    text_view_threshold.visibility = View.VISIBLE
35    text_view_threshold.text = "Menghitung nilai threshold..."
36    isCalculatingStarted = true
37    if (isStarted) {
38      Timer().schedule(30000) {
39        isCalculateTimeOver = true
40        var totalRate = 0
41        for (i in 0 until heartRateForThreshold.size) {
42        totalRate += heartRateForThreshold[i]
43        }
44        val mean =
totalRate.toDouble().div(heartRateForThreshold.size.toDouble())
45        threshold = (mean - 8.0).roundToInt()
46        runOnUiThread {
47          text_view_threshold.text =
getString(R.string.threshold_detection, threshold.toString())
48        }
49      }
50    }
51  }
52  private fun isDrowsy(heartRate: Int): Boolean {
53    return heartRate <= threshold
54  }
55  private fun turnOnAlarm() {
56    isAlarmActive = true
57    val intent = Intent(this, AlarmActivity::class.java)
58    startActivity(intent)
59  }
60  }
61
```

Penjelasan kode program pada Tabel 6.11 sebagai berikut.

- Baris ke 2-10 merupakan *method* onCreate(). Method onCreate() ialah *method* yang pertama kali dijalankan saat *activity* baru dibuka. Pada *method* ini dilakukan pemanggilan *method* startMeasure() yang terdapat pada ViewModel yang berfungsi untuk memulai pengukuran detak jantung pada *smartwatch*. Selanjutnya dilakukan pemanggilan *method* observeHeartRate() untuk mendapatkan nilai detak jantung pengguna.
- Baris ke 12-31 merupakan *method* observeHeartRate() yang berfungsi untuk mengobservasi nilai detak jantung pengguna yang didapatkan melalui *smartwatch,* nilai detak jantung ini didapatkan dari variabel heartRate dari

ViewModel. Pada *method* observeHeartRate() ini secara kontinyu melakukan pemanggilan *method* isDrowsy() yang berfungsi mengukur apakah nilai detak jantung kurang dari atau sama dengan *threshold,* jika ya, maka pengguna dianggap mengantuk sehingga memanggil *method* turnOnAlarm() yang berfungsi untuk mengarahkan ke *activity* alarm.

- Baris ke 33-52 merupakan *method* calculateThreshold() yang berfungsi untuk menghitung nilai *threshold* detak jantung pengguna. Pada *method* ini dilakukan perekaman data detak jantung pengguna selama 30 detik pertama yang kemudian dihitung rata-ratanya. Selanjutnya nilai rata-rata tersebut dikurangi dengan 8 sehingga didapatkanlah nilai *threshold,* kemudian nilai *threshold* ditampilkan ke pengguna.
- Baris ke 53-55 merupakan *method* isDrowsy() yang menentukan apakah pengguna dalam kondisi mengantuk atau tidak. Jika nilai detak jantung kurang dari atau sama dengan *threshold* maka *method* isDrowsy() mengembalikan nilai TRUE, namun jika tidak maka akan mengembalikan nilai FALSE.
- Baris ke 56-60 merupakan *method* turnOnAlarm() yang akan dipanggil jika pengguna terdeteksi mengantuk. *Method* ini berfungsi untuk mengarahkan aplikasi ke *activity* alarm.

Tabel 6.12 Kode Program AlarmActivity.kt

AlarmActivity.kt

```
1   class AlarmActivity : AppCompatActivity() {
2       companion object{
3           private val MAX_VOLUME = 1.0f
4       }
5       private lateinit var mediaPlayer: MediaPlayer
6
7       override fun onCreate(savedInstanceState: Bundle?) {
8           super.onCreate(savedInstanceState)
9           setContentView(R.layout.activity_alarm)
10
11          mediaPlayer = MediaPlayer.create(applicationContext,
    R.raw.audio_alarm)
12          setAlarm(true)
13          button_emergency_call_alarm.setOnClickListener {
14              var number = ""
15              for (i in DrowsyDetectionActivity.contactList.indices)
16  {
17                  if
18  (DrowsyDetectionActivity.contactList[i].isPrimary == 1) {
19
```

```
20                      number =
DrowsyDetectionActivity.contactList[i].phoneNumber
21                      break
22                  }
23              }
24              val intent = Intent(Intent.ACTION_DIAL,
Uri.fromParts("tel", number, null))
25              startActivity(intent)
26          }
27          button_ignore.setOnClickListener {
28              setAlarm(false)
29              DrowsyDetectionActivity.isAlarmActive = false
30              DrowsyDetectionActivity().respawnAfterAlarm()
31              finish()
32          }
33      }
34
35      private fun setAlarm(isPlay: Boolean){
36          if (isPlay){
37              mediaPlayer.start()
38              mediaPlayer.isLooping = true
39              mediaPlayer.setVolume(MAX_VOLUME, MAX_VOLUME)
40          }
41          else{
42              mediaPlayer.stop()
43          }
44      }
45  }
```

Penjelasan dari kode program pada Tabel 6.12 adalah sebagai berikut.

- Baris ke 7-33 merupakan *method* onCreate(), *method* ini adalah *method* yang pertama kali dipanggil saat *activity* dibuat. Pada *method* onCreate() terdapat deklarasi variabel mediaPlayer yang digunakan untuk memutar *file* suara alarm. Selain itu pada *method* ini juga dilakukan pemanggilan *method* setAlarm() untuk mulai memutar suara alarm. Perintah yang akan dijalankan ketika tombol "Panggilan Darurat" dan "Abaikan" ditekan juga didefinisikan di *method* ini.
- Baris ke 35-44 merupakan *method* setAlarm() yang berfungsi untuk memulai dan memberhentikan suara alarm. Jika parameter bernilai TRUE, maka *method* menjalankan fungsi untuk memulai pemutaran suara alarm, namun jika FALSE maka akan menjalankan fungsi untuk memberhentikan suara.

Tabel 6.13 Implementasi Kode Program DrowsyDetectionViewModel

DrowsyDetectionViewModelViewModel.kt

```
1  class DrowsyDetectionViewModel(private val context: Context) :
   ViewModel()
2  {
3  fun connect() {
4    val mAgentCallback: SAAgentV2.RequestAgentCallback = object :
   SAAgentV2.RequestAgentCallback {
5      override fun onAgentAvailable(agent: SAAgentV2?) {
6      mService = agent as SamsungAccessoryService
7      mService?.connectionStatusCallback =
   this@DrowsyDetectionViewModel
8      mService?.heartRateMeasureCallback =
   this@DrowsyDetectionViewModel
9      mService?.findPeers()
10     }
11   }
12
13 SAAgentV2.requestAgent(context,SamsungAccessoryService::class.java.
   name, mAgentCallback)
14 }
15
16 fun startMeasure() {
17   connect()
18   mService?.sendData(Constants.COMMAND_START_MEASURE_HR)
19 }
20 }
21
22
23
24
```

Penjelasan dari kode program pada Tabel 6.13 adalah sebagai berikut.

- Baris ke 4-17 merupakan *method* connect() yang berfungsi untuk membuat menghubungkan aplikasi ke *smartwatch* dengan menggunakan protokol yang telah disediakan Samsung Accessory SDK.
- Baris ke 19-22 merupakan *method* startMeasure() yang berfungsi untuk mengirimkan data ke *smartwatch* berupa perintah untuk memulai pengukuran detak jantung .

6.4.3 Mengirimkan Pesan Darurat (APK-F-002)

Implementasi kode program untuk fungsi mengirimkan pesan darurat dijabarkan pada Tabel 6.14.

Tabel 6.14 Kode Program DrowsyDetectionActivity.kt (Mengirimkan Pesan Darurat)

DrowsyDetectionActivity.kt

```
1   class DrowsyDetectionActivity : AppCompatActivity(){
2   override fun onCreate(savedInstanceState: Bundle?) {
3     super.onCreate(savedInstanceState)
4     setContentView(R.layout.activity_drowsy_detection)
5     viewModel = obtainViewModel(this)
6     if (!isStarted) {
7       viewModel.startMeasure()
8     }
9     observeHeartRate()
10  }
11
12  private fun observeHeartRate() {
13    val heartRateObserver = Observer<String> { heartRate ->
14    if (heartRate != null) {
15      isStarted = true
16      text_view_heart_rate.text = heartRate
17      if (!isCalculatingStarted) {
18        calculateThreshold()
19      }
20      if (!isCalculateTimeOver) {
21        heartRateForThreshold.add(heartRate.toInt())
22      }
23      if (threshold != 0) {
24        if (isDrowsy(heartRate.toInt()) && !isAlarmActive &&
!isPaused) {
25          drowsyCount++
26          turnOnAlarm()
27        }
28      }
29    }
30    viewModel.heartRate.observe(this, heartRateObserver)
31  }
32  private fun calculateThreshold() {
33    text_view_threshold.visibility = View.VISIBLE
34    text_view_threshold.text = "Menghitung nilai threshold..."
```

```
35    isCalculatingStarted = true
36    if (isStarted) {
37      Timer().schedule(30000) {
38        isCalculateTimeOver = true
39        var totalRate = 0
40        for (i in 0 until heartRateForThreshold.size) {
41        totalRate += heartRateForThreshold[i]
42        }
43        val mean =
totalRate.toDouble().div(heartRateForThreshold.size.toDouble())
44        threshold = (mean - 8.0).roundToInt()
45        runOnUiThread {
46          text_view_threshold.text =
getString(R.string.threshold_detection, threshold.toString())
47        }
48      }
49    }
50  }
51  private fun isDrowsy(heartRate: Int): Boolean {
52    return heartRate <= threshold
53  }
54  private fun getLocation() {
55    val locationCallback = object : LocationCallback() {
56      override fun onLocationResult(locationResult: LocationResult?)
{
57        locationResult ?: return
58        for (location in locationResult.locations) {
59          saveLocationAddress(location)
60          fusedLocationClient.removeLocationUpdates(this)
61        }
62      }
63    }
64    val locationRequest = LocationRequest.create()?.apply {
65      interval = 10000
66      fastestInterval = 5000
67      priority = LocationRequest.PRIORITY_HIGH_ACCURACY
68    }
69    val builder = LocationSettingsRequest.Builder()
70    .addLocationRequest(locationRequest as LocationRequest)
71    val client: SettingsClient =
LocationServices.getSettingsClient(this)
72    val task: Task<LocationSettingsResponse> =
client.checkLocationSettings(builder.build())
73
74
75
```

```
76    task.addOnSuccessListener {
77      fusedLocationClient.requestLocationUpdates(locationRequest,
78  locationCallback, null)
79    }
80    task.addOnFailureListener { exception ->
81      if (exception is ResolvableApiException) {
82        try {
83          startResolutionForResult(),
84          exception.startResolutionForResult(this,
85  REQUEST_LOCATION_AND_SMS)
86        } catch (sendEx: IntentSender.SendIntentException) {}
87      }
88    }
89  private fun sendEmergencySMS() {
90    val address = drowsyLocation[drowsyCount - 1].substringAfter("/")
91    val time = drowsyLocation[drowsyCount - 1].substringBefore("/")
92    try {
93      val smsManager: SmsManager = SmsManager.getDefault()
94      val sentIntentFilter = IntentFilter(SMS_SENT_ACTION)
95      val messageSentReceiver: BroadcastReceiver = object :
96  BroadcastReceiver() {
97        override fun onReceive(context: Context?, intent: Intent?) {
98          when (resultCode) {
99            Activity.RESULT_OK -> Toast.makeText(context,"Pesan darurat
100 terkirim", Toast.LENGTH_SHORT).show()
101           SmsManager.RESULT_ERROR_GENERIC_FAILURE ->
102 Toast.makeText(context, "Pesan gagal
103 dikirim",Toast.LENGTH_SHORT).show()
104         }
105       }
106   }
107     registerReceiver(messageSentReceiver, sentIntentFilter)
108     val messageParts =
109 smsManager.divideMessage(getString(R.string.emergency_message,
110 address, time))
111     val sentPIList = arrayListOf<PendingIntent>()
112     for (i in messageParts.indices) {
113       sentPIList.add(PendingIntent.getBroadcast(applicationContext,
114 0, Intent(SMS_SENT_ACTION), 0))
115     }
116     for (i in contactList.indices) {
117       smsManager.sendMultipartTextMessage(
118       contactList[i].phoneNumber, null, messageParts, sentPIList,
119 null)
120     }
```

```
117     } catch (ex: Exception) {
118         Toast.makeText(applicationContext, ex.message.toString(),
    Toast.LENGTH_LONG).show()
119         ex.printStackTrace()
120     }
121   }
122 }
123
124
```

Penjelasan kode program pada Tabel 6.14 sebagai berikut.

- Baris ke 2-10 merupakan *method* onCreate(). Method onCreate() ialah *method* yang pertama kali dijalankan saat *activity* baru dibuka. Pada *method* ini dilakukan pemanggilan *method* startMeasure() yang terdapat pada ViewModel yang berfungsi untuk memulai pengukuran detak jantung pada *smartwatch*. Selanjutnya dilakukan pemanggilan *method* observeHeartRate() untuk mendapatkan nilai detak jantung pengguna.
- Baris ke 12-31 merupakan *method* observeHeartRate() yang berfungsi untuk mengobservasi nilai detak jantung pengguna yang didapatkan melalui *smartwatch,* nilai detak jantung ini didapatkan dari variabel heartRate dari ViewModel. Pada *method* observeHeartRate() ini secara kontinyu melakukan pemanggilan *method* isDrowsy() yang berfungsi mengukur apakah nilai detak jantung kurang dari atau sama dengan *threshold,* jika ya, maka pengguna dianggap mengantuk sehingga memanggil *method* turnOnAlarm() yang berfungsi untuk mengarahkan ke *activity* alarm.
- Baris ke 33-52 merupakan *method* calculateThreshold() yang berfungsi untuk menghitung nilai *threshold* detak jantung pengguna. Pada *method* ini dilakukan perekaman data detak jantung pengguna selama 30 detik pertama yang kemudian dihitung rata-ratanya. Selanjutnya nilai rata-rata tersebut dikurangi dengan 8 sehingga didapatkanlah nilai *threshold,* kemudian nilai *threshold* ditampilkan ke pengguna.
- Baris ke 53-55 merupakan *method* isDrowsy() yang menentukan apakah pengguna dalam kondisi mengantuk atau tidak. Jika nilai detak jantung kurang dari atau sama dengan *threshold* maka *method* isDrowsy() mengembalikan nilai TRUE, namun jika tidak maka akan mengembalikan nilai FALSE.
- Baris ke 56-88 merupakan *method* getLocation() yang berfungsi untuk mendapatkan lokasi. Untuk mendapatkan lokasi perangkat, pada *method* ini mengimplementasikan API FusedLocationService.
- Baris ke 89-123 merupakan *method* sendEmergencySMS() yang berfungsi untuk mengirimkan SMS berupa pesan darurat ke nomor kontak darurat. Pesan yang dikirim berisi pesan peringatan, alamat, dan waktu pengguna terdeteksi mengantuk.

6.4.4 Menyimpan Nomor Darurat (APK-F-003)

Implementasi kode program untuk fungsi menyimpan nomor darurat dijabarkan pada Tabel 6.15 sampai dengan Tabel 6.16.

Tabel 6.15 Kode Program AddEmergencyContactFragment.kt

AddEmergencyContactFragment.kt

```
1  class AddEmergencyContactFragment : SuperBottomSheetFragment() {
2  override fun onActivityCreated(savedInstanceState: Bundle?) {
3    super.onActivityCreated(savedInstanceState)
4    if (activity != null){
5      observeEmergencyContact()
6    }
7    button_save_contact.setOnClickListener {
8      if (isInputValid()){
9        saveContact()
10       this@AddEmergencyContactFragment.dismiss()
11     }
12   }
13 }
14 private fun observeEmergencyContact() {
15   val emergencyContactObserver = Observer<List<EmergencyContact>> {
   contactList ->
16     if (contactList != null){
17       this.contactList.clear()
18       if (contactList.isNotEmpty()){
19         this.contactList.addAll(contactList)
20       }
21       else{
22         check_box_is_primary.visibility = View.GONE
23         check_box_is_primary.isChecked = true
24       }
25     }
26   }
27   runBlocking {
28 viewModel.getAllContact().observe(this@AddEmergencyContactFragment,
   emergencyContactObserver)
29   }
30 }
31 private fun saveContact(){
32   var isPrimary = 0
33   if (check_box_is_primary.isChecked){
34
```

```
35      for (i in contactList.indices){
36          contactList[i].isPrimary = 0
37          runBlocking {
38              viewModel.updateContact(contactList[i])
39          }
40      }
41      isPrimary = 1
42  }
43  val contact = EmergencyContact(
44  phoneNumber = edit_text_contact_number.text.toString().trim(),
45  contactName = edit_text_contact_name.text.toString().trim(),
46  isPrimary = isPrimary)
47  runBlocking {
48      viewModel.insertContact(contact)
49  }
50  }
51  }
```

Penjelasan dari kode program pada Tabel 6.15 adalah sebagai berikut.

- Baris ke 2-13 adalah *method* onActivityCreated() yang merupakan *method* yang dipanggil ketika *activity* dari *fragment* dibuat. Pada *method* ini mendefinisikan perintah yang akan dijalankan ketika tombol "Simpan" ditekan.
- Baris ke 14-31 merupakan *method* observeEmergencyContact() yang berfungsi untuk mengobservasi daftar kontak darurat yang tersimpan di *database* lokal dengan cara memanggil *method* getAllContact() dari ViewModel.
- Baris ke 32-50 merupakan *method* saveContact() yang berfungsi untuk menyimpan kontak darurat ke *database* lokal. Pada *method* ini juga terdapat mekanisme untuk melakukan pembaharuan data terhadap kontak-kontak yang telah tersimpan sebelumnya jika kontak baru yang akan disimpan dipilih pengguna sebagai kontak utama untuk menghindari adanya lebih dari satu kontak utama. Untuk menyimpan kontak, *method* ini memanggil *method* insertContact() dari ViewModel.

Tabel 6.16 Implementasi Kode Program EmergencyContactViewModel.kt

EmergencyContactViewModel.kt	
1	class EmergencyContactViewModel(private val context: Context) : ViewModel()
2	{
3	

```
4   suspend fun insertContact(contact: EmergencyContact) {
5     try {
6       apkDao.insertContact(contact)
7     } catch (ex: SQLiteConstraintException) {
8       Toast.makeText(context, "Kontak darurat sudah ada!",
    Toast.LENGTH_SHORT).show()
9     }
10  }
11  fun getAllContact(): LiveData<List<EmergencyContact>> {
12    return apkDao.getAllContact()
13  }
14  suspend fun updateContact(contact: EmergencyContact) {
15    apkDao.updateContact(contact)
16  }
17  suspend fun deleteContact(contact: EmergencyContact) {
18    apkDao.deleteContact(contact)
19  }
20
```

Penjelasan dari kode program pada Tabel 6.16 adalah sebagai berikut.

- Baris ke 4-11 merupakan *method* insertContact() yang berfungsi untuk menyimpan kontak ke *database* lokal. *Method* insertContact() memanggil *method* insertContact() dari kelas ContactDao untuk menjalankan *query* SQLite.
- Baris 12-14 merupakan *method* getAllContact() yang berfungsi untuk mendapatkan daftar kontak darurat yang tersimpan. *Method* ini memanggil *method* getAllContact() dari kelas ContactDao.
- Baris 15-17 merupakan *method* updateContact() yang berfungsi untuk memperbaharui data kontak darurat yang tersimpan. *Method* ini memanggil *method* updateContact() dari kelas ContactDao.
- Baris 18-20 merupakan *method* deleteContact() yang berfungsi untuk menghapus data kontak darurat yang tersimpan. *Method* ini memanggil *method* deleteContact() dari kelas ContactDao.

6.4.5 Melihat Histori (APK-F-004)

Implementasi kode program untuk fungsi melihat histori tertera pada Tabel 6.17 sampai dengan Tabel 6.18.

Tabel 6.17 Implementasi Kode Program HistoryActivity.kt

HistoryActivity.kt	
1	`class HistoryActivity : AppCompatActivity() {`

```
2   private fun observeHistory() {
3     val historyObserver = Observer<List<History>> { historyList ->
4       if (historyList != null){
5         this.historyList.clear()
6         if (historyList.isNotEmpty()){
7           this.historyList.addAll(historyList)
8         }
9         showHistory()
10      }
11    }
12    runBlocking {
13      viewModel.getAllHistory().observe(this@HistoryActivity,
14  historyObserver)
15    }
16  }
17  private fun showHistory(){
18    val reverseLayoutManager = LinearLayoutManager(applicationContext)
19    reverseLayoutManager.reverseLayout = true
20    reverseLayoutManager.stackFromEnd = true
21    recycler_view_history.apply {
22      layoutManager = reverseLayoutManager
23      adapter = HistoryAdapter(historyList, object :
24  HistoryAdapter.HistoryListener{
25        override fun onItemClicked(history: History) {
26          val intent = Intent(applicationContext,
27  HistoryDetailActivity::class.java)
28          intent.putExtra(HistoryDetailActivity.HISTORY_DATA_EXTRA,
29  history)
30          startActivity(intent)
31        }
32      })
33    }
  }
```

Penjelasan dari kode program pada Tabel 6.17 adalah sebagai berikut.

- Baris 2-16 merupakan *method* observeHistory() yang berfungsi untuk mengobservasi daftar histori yang tersimpan di *database* lokal. Untuk mendapatkan daftar histori, *method* ini memanggil *method* getAllHistory() dari ViewModel. Kemudian dilakukan pemanggilan *method* showHistory() untuk menampilkan daftar histori dalam bentuk *list*.

- Baris 17-31 merupakan *method* showHistory() yang berfungsi untuk menampilkan daftar histori dalam bentuk *list*. Data histori yang telah didapatkan diimplementasikan ke dalam *recycler view.*

Tabel 6.18 Implementasi Kode Program HistoryViewModel.kt

HistoryViewModel.kt

```
1  class HistoryViewModel(private val context: Context) : ViewModel()
2  {
3  suspend fun insertHistory(history: History) {
4    historyDao.insertHistory(history)
5  }
6  fun getAllHistory(): LiveData<List<History>> {
7    return historyDao.getAllHistory()
8  }
9  suspend fun deleteHistory(history: History) {
10   historyDao.deleteHistory(history)
11 }
```

Penjelasan dari kode program pada Tabel 6.18 adalah sebagai berikut.

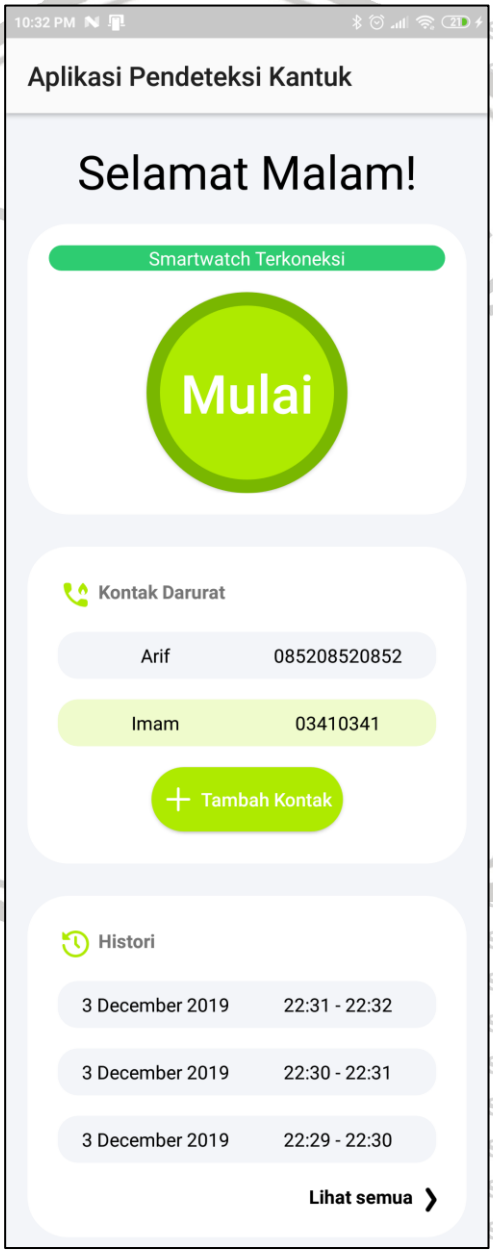
- Baris 3-5 merupakan *method* insertHistory() yang berfungsi untuk menyimpan data histori ke dalam *database* lokal dengan cara melakukan pemanggilan *method* insertHistory() dari HistoryDao.
- Baris 6-8 merupakan *method* getAllHistory() yang berfungsi untuk mendapatkan daftar seluruh histori yang tersimpan di *database* lokal dengan cara memanggil *method* getAllHistory() dari HistoryDao.
- Baris 9-11 merupakan *method* deleteHistory() yang berfungsi untuk menghapus data histori yang tersimpan dalam *database* lokal dengan cara memanggil *method* deleteHistory() dari HistoryDao.

6.5 Implementasi Antarmuka

Tahap implementasi antarmuka menyajikan hasil implementasi sistem dari segi antarmuka sesuai dengan rancangan yang telah dibuat.

6.5.1 Halaman Utama

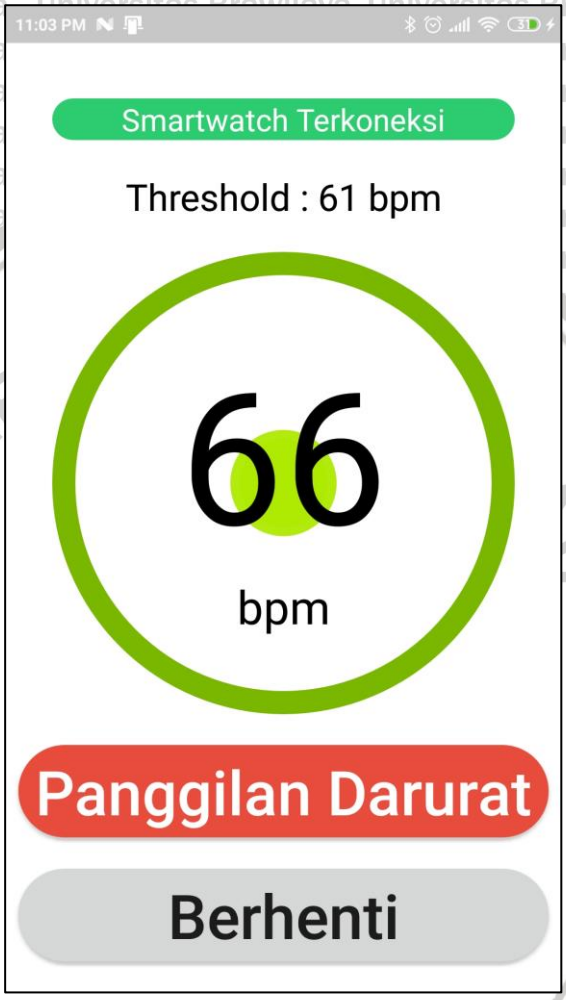
Terdapat tiga bagian pada halaman utama, yaitu tombol untuk memulai pendeteksian kantuk, kontak darurat, dan histori. Tombol pendeteksian kantuk akan mengarahkan ke fungsi deteksi kantuk, bagian kontak darurat menampilkan seluruh kontak darurat yang tersimpan serta tombol untuk menambah kontak, dan bagian histori hanya menampilkan tiga histori terbaru. Tampilan antarmuka halaman utama terdapat pada Gambar 6.1.



Gambar 6.1 Antarmuka Halaman Utama

6.5.2 Halaman Pendeteksian Kantuk

Halaman pendeteksian kantuk menampilkan beberapa informasi dan menu yang dapat diakses pengguna saat menjalankan fungsi pendeteksian kantuk. Informasi yang ditampilkan diantaranya status koneksi, nilai *threshold,* nilai detak jantung, tombol panggilan darurat, dan tombol berhenti. Tampilan antarmuka halaman pendeteksian kantuk dapat dilihat pada Gambar 6.2



Gambar 6.2 Antarmuka Halaman Pendeteksian Kantuk

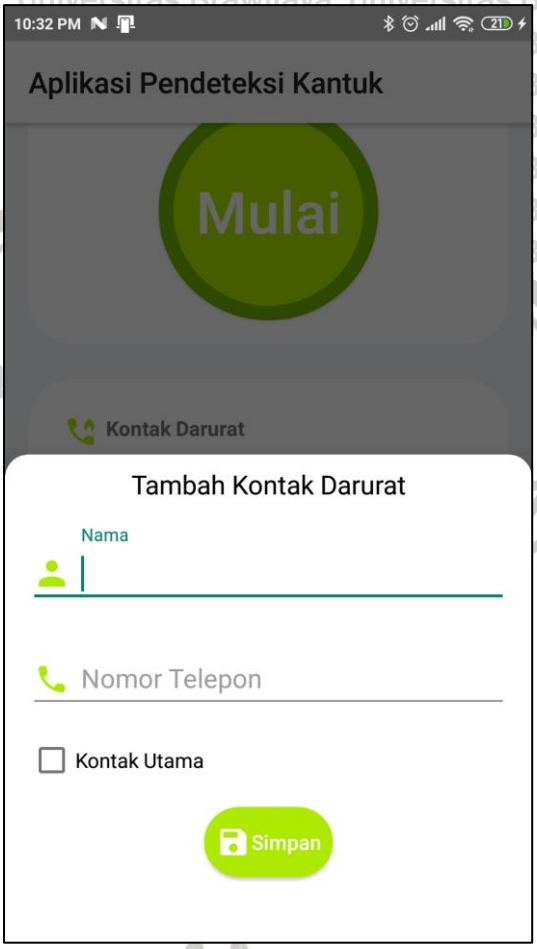
6.5.3 Halaman Alarm Peringatan

Halaman alarm peringatan menampilkan pesan peringatan untuk pengendara serta tombol untuk melakukan panggilan darurat dan mengabaikan alarm. Hasil implementasi antarmuka halaman alarm peringatan terdapat pada Gambar 6.3.

Gambar 6.3 Antarmuka Halaman Alarm Peringatan

6.5.4 Halaman Tambah Kontak Darurat

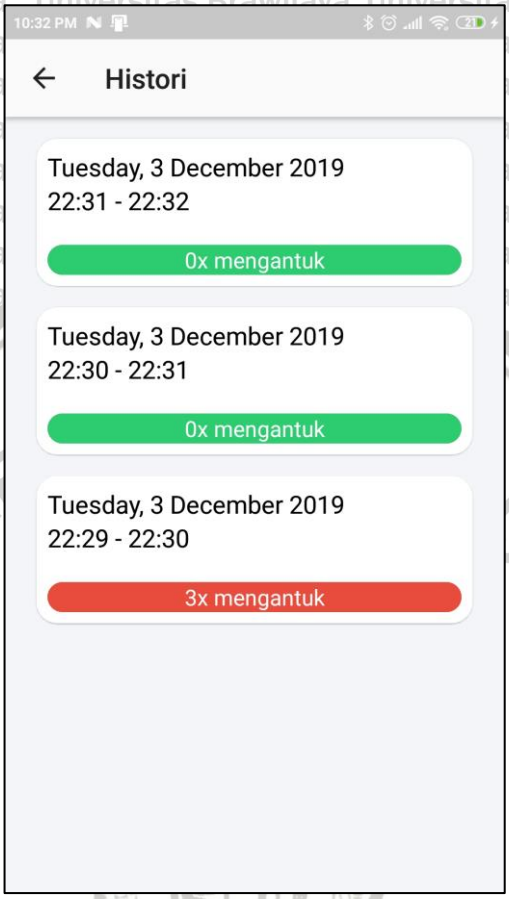
Halaman tambah kontak darurat berupa *fragment* yang ditampilkan dalam bentuk *bottom sheet*. Pada halaman ini terdapat *form* untuk mengisi data kontak darurat yang terdiri dari nama, nomor telepon, dan *checkbox* untuk menjadikan kontak sebagai kontak darurat utama. Hasil implementasi antarmuka halaman tambah kontak darurat dapat dilihat pada Gambar 6.4.



Gambar 6.4 Antarmuka Halaman Tambah Kontak Darurat

6.5.5 Halaman Histori

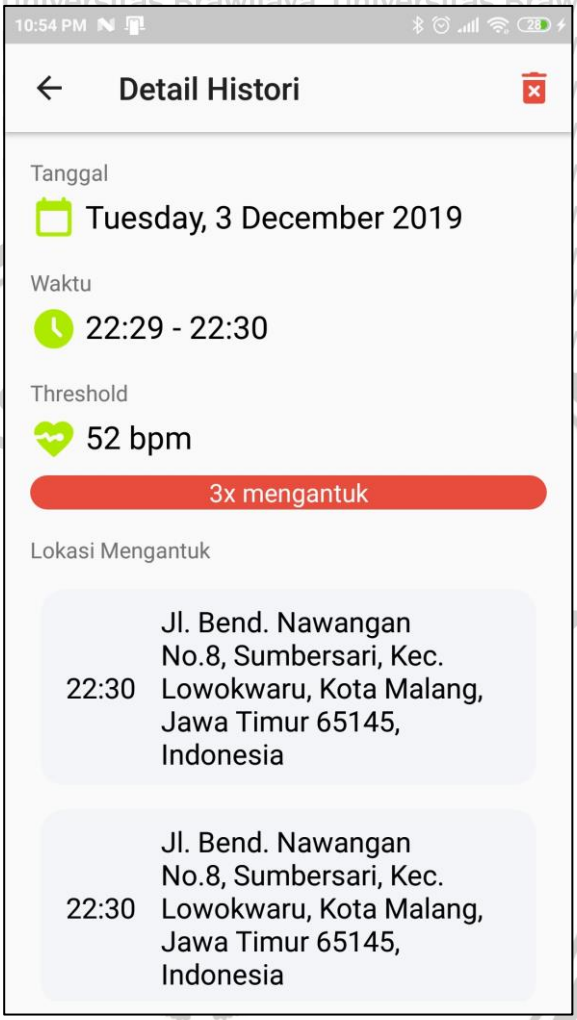
Halaman histori menampilkan seluruh daftar histori pendeteksian kantuk pengguna dalam bentuk *list,* antarmuka halaman histori terdapat pada Gambar 6.5.



Gambar 6.5 Antarmuka Halaman Histori

6.5.6 Halaman Detail Histori

Halaman detail histori menampilkan informasi detail tentang histori pendeteksian kantuk yang dipilih, informasi yang ditampilkan diantaranya tanggal, waktu, *threshold*, jumlah kantuk, dan lokasi kantuk. Hasil implementasi antarmuka halaman detail histori terdapat pada Gambar 6.6.



Gambar 6.6 Antarmuka Halaman Detail Histori

BAB 7 PENGUJIAN

Pada bab ini menjelaskan teknik, hasil, serta analisis hasil pengujian yang dilakukan terhadap aplikasi pendeteksi kantuk. Pembahasan terbagi ke dalam tiga subbab yaitu Pengujian Validasi, Pengujian Akurasi, dan Analisis Hasil Pengujian.

7.1 Pengujian *Black Box*

Pengujian *black box* bertujuan untuk mengetahui apakah perangkat lunak yang dibuat telah memenuhi kebutuhan fungsional yang telah ditentukan. Sesuai dengan jumlah kebutuhan fungsional, pada pengujian validasi aplikasi pendeteksi kantuk terdapat empat kasus uji.

7.1.1 Kasus Uji Mendapatkan Alarm Peringatan

Kasus uji dari fungsi mendapatkan alarm peringatan tertera pada Tabel 7.1.

Tabel 7.1 Kasus Uji Mendapatkan Alarm Peringatan

No. Kasus Uji	APK-UJI-001
Nama Kasus Uji	Mendapatkan Alarm Peringatan
Objek Uji	*Use case* APK-F-001
Tujuan Pengujian	Membuktikan bahwa aplikasi dapat memberikan alarm peringatan kepada pengguna ketika mengantuk.
Prosedur Pengujian	1. Pengguna menekan tombol “Mulai”. 2. Aplikasi mulai mendeteksi kantuk untuk pengguna.
Hasil yang diharapkan	Alarm peringatan berbunyi ketika pengguna mengantuk dan tidak berbunyi ketika pengguna tidak mengantuk.
Hasil yang didapatkan	Alarm peringatan berbunyi ketika pengguna mengantuk dan tidak berbunyi ketika pengguna tidak mengantuk.

7.1.2 Kasus Uji Mengirimkan Pesan Darurat

Kasus uji untuk fungsi mengirimkan pesan darurat tertera pada Tabel 7.2.

Tabel 7.2 Kasus Uji Mengirimkan Pesan Darurat

No. Kasus Uji	APK-UJI-002
Nama Kasus Uji	Mengirimkan Pesan Darurat
Objek Uji	*Use case* APK-F-002
Tujuan Pengujian	Membuktikan bahwa aplikasi dapat mengirimkan pesan darurat ketika pengguna mengantuk.

Prosedur Pengujian	1. Pengguna menekan tombol "Mulai". 2. Aplikasi mulai mendeteksi kantuk pengguna.
Hasil yang diharapkan	Pesan darurat dikirimkan ketika pengguna terdeteksi mengantuk.
Hasil yang didapatkan	Pesan darurat dikirimkan ketika pengguna terdeteksi mengantuk.

7.1.3 Kasus Uji Menyimpan Nomor Darurat

Kasus uji untuk fungsi menyimpan nomor darurat tertera pada Tabel 7.3.

Tabel 7.3 Kasus Uji Menyimpan Nomor Darurat

No. Kasus Uji	APK-UJI-003
Nama Kasus Uji	Menyimpan Nomor Darurat
Objek Uji	*Use case* APK-F-003
Tujuan Pengujian	Membuktikan bahwa aplikasi dapat menyimpan nomor darurat.
Prosedur Pengujian	1. Pengguna menekan tombol "Tambah Kontak" pada bagian kontak darurat di halaman utama. 2. Aplikasi menampilkan *form* kontak yang terdiri dari nama dan nomor telepon. 3. Pengguna mengisi *form* kontak darurat dan menekan tombol "Simpan".
Hasil yang diharapkan	Kontak darurat berhasil disimpan.
Hasil yang didapatkan	Kontak darurat berhasil disimpan.

7.1.4 Kasus Uji Melihat Histori

Tabel 7.4 Kasus Uji Melihat Histori

No. Kasus Uji	APK-UJI-004
Nama Kasus Uji	Melihat Histori
Objek Uji	*Use case* APK-F-004
Tujuan Pengujian	Membuktikan bahwa aplikasi dapat menampilkan histori pendeteksian kantuk.

Prosedur Pengujian	1. Pengguna menekan teks "Lihat Semua" pada bagian histori di halaman utama. 2. Aplikasi menampilkan daftar histori pendeteksian kantuk.
Hasil yang diharapkan	Histori pendeteksian kantuk ditampilkan.
Hasil yang didapatkan	Histori pendeteksian kantuk ditampilkan.

7.1.5 Hasil Pengujian *Black Box*

Hasil dari pengujian *black box* aplikasi pendeteksi kantuk dapat dilihat pada Tabel 7.5

Tabel 7.5 Hasil Pengujian *Black Box*

No. Kasus Uji	Kasus Uji	Hasil yang diharapkan	Hasil yang didapatkan	Status
APK-UJI-001	Pengendara melakukan pendeteksian kantuk saat kondisi mengantuk.	Alarm peringatan berbunyi.	Alarm peringatan berbunyi.	Valid
	Pengendara melakukan pendeteksian kantuk saat kondisi normal.	Alarm peringatan tidak berbunyi.	Alarm peringatan tidak berbunyi.	Valid
APK-UJI-002	Pengendara melakukan pendeteksian kantuk saat kondisi mengantuk.	SMS darurat dikirimkan ke nomor kontak darurat.	SMS darurat dikirimkan ke nomor kontak darurat.	Valid
APK-UJI-003	Pengendara memasukkan nama dan nomor telepon kontak.	Kontak darurat tersimpan di aplikasi.	Kontak darurat tersimpan di aplikasi.	Valid

	Pengendara memasukkan nomor yang telah tersimpan sebelumnya.	Aplikasi menampilkan pesan "Kontak darurat sudah ada!"	Aplikasi menampilkan pesan "Kontak darurat sudah ada!"	Valid
APK-UJI-004	Pengendara menekan teks "Lihat semua" pada bagian histori di halaman utama.	Aplikasi menampilkan seluruh histori pendeteksian kantuk dalam bentuk *list.*	Aplikasi menampilkan seluruh histori pendeteksian kantuk dalam bentuk *list.*	Valid

7.2 Pengujian Akurasi

Pada pengujian akurasi metode yang digunakan ialah dengan membandingkan hasil pendeteksian aplikasi pendeteksi kantuk dengan tingkat kantuk pengguna yang diukur dengan *Karolinska Sleepiness Scale* (KSS), penjelasan mengenai KSS dapat dilihat pada Bab 2. Pengujian dilakukan terhadap lima responden dengan durasi selama 5-15 menit. Penulis membandingkan tingkat kantuk responden dengan hasil pendeteksian aplikasi setiap 5 menit sekali atau ketika alarm peringatan aplikasi berbunyi. Tingkat kantuk ditentukan sendiri oleh responden dengan menggunakan skala KSS, responden dianggap mengantuk jika berada pada skala 7 *(Sleepy, but no effort to keep awake)*, 8 *(Sleepy, but some effort to keep awake)*, atau 9 *(Very sleepy, great effort to keep awake, fighting sleep)*. Pengujian dianggap valid jika tingkat kantuk responden antara 1-5 dan alarm aplikasi tidak berbunyi atau tingkat kantuk antara 7-9 dan alarm aplikasi berbunyi.

7.2.1 Hasil Pengujian Akurasi

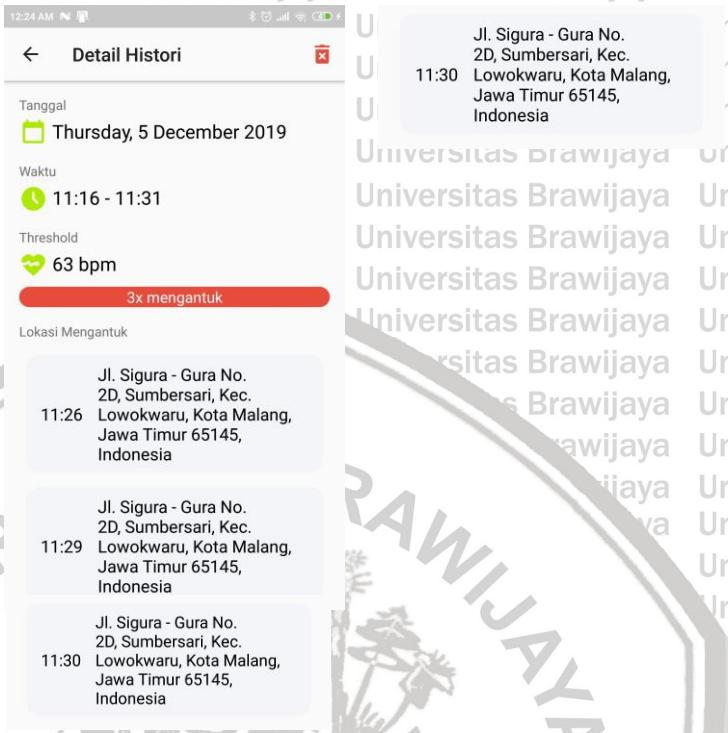
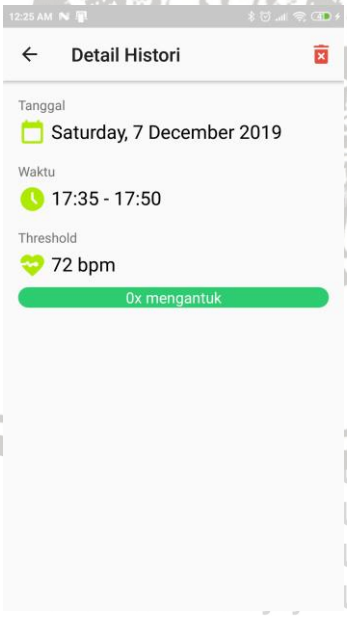
Hasil pengujian akurasi aplikasi pendeteksi kantuk terdapat pada Tabel 7.6 dan Tabel 7.7.

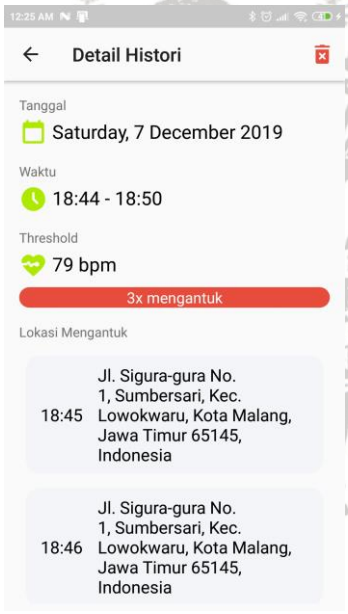
Tabel 7.6 Hasil Pengujian Akurasi

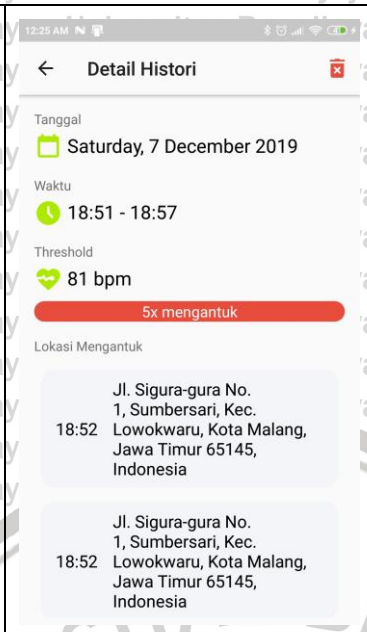
Responden	Durasi (menit)	Menit	Tingkat Kantuk	Alarm Menyala (Y/T)	Status
1	15	5	5	T	Valid
		10	5	Y	Tidak Valid

		13	7	Y	Valid
		14	7	Y	Valid
		15	6	T	Valid
2	15	5	4	T	Valid
		10	5	T	Valid
		15	6	T	Valid
3	15	2	4	Y	Tidak Valid
		5	4	T	Valid
		10	6	T	Valid
		15	7	Y	Valid
4	5	1	7	Y	Valid
		2	7	Y	Valid
		4	7	Y	Valid
		5	6	T	Valid
5	6	1	7	Y	Valid
		1	7	Y	Valid
		3	8	Y	Valid
		4	8	Y	Valid
		5	8	T	Tidak Valid
		6	8	Y	Valid

Tabel 7.7 Validasi Pengujian Akurasi

Responden	*Screenshot* Aplikasi Pendeteksi Kantuk
1	
2	

3	 Detail Histori Tanggal Saturday, 7 December 2019 Waktu 17:55 - 18:10 Threshold 82 bpm 2x mengantuk Lokasi Mengantuk 17:57 Jl. Sigura-gura III No. 1, Sumbersari, Kec. Lowokwaru, Kota Malang, Jawa Timur 65145, Indonesia 18:10 Jl. Sigura-gura III No. 1, Sumbersari, Kec. Lowokwaru, Kota Malang, Jawa Timur 65145, Indonesia
4	 Detail Histori Tanggal Saturday, 7 December 2019 Waktu 18:44 - 18:50 Threshold 79 bpm 3x mengantuk Lokasi Mengantuk 18:45 Jl. Sigura-gura No. 1, Sumbersari, Kec. Lowokwaru, Kota Malang, Jawa Timur 65145, Indonesia 18:46 Jl. Sigura-gura No. 1, Sumbersari, Kec. Lowokwaru, Kota Malang, Jawa Timur 65145, Indonesia 18:49 Jl. Sigura-gura No. 1, Sumbersari, Kec. Lowokwaru, Kota Malang, Jawa Timur 65145, Indonesia

5	

7.3 Pengujian *Usability*

Untuk menguji tingkat *usability* dari aplikasi pendeteksi kantuk, metode yang digunakan ialah dengan merekrut lima orang responden untuk mencoba seluruh fitur dari aplikasi dan selanjutnya mengisikan kuesioner *System Usability Scale* (SUS). Kuesioner terdiri dari sepuluh pernyataan yang masing-masing harus diberi skor oleh responden, dengan skala dari 1 sampai 5, di mana 1 berarti sangat tidak setuju dan 5 sangat setuju. Kuesioner SUS dapat dilihat pada Tabel 7.8.

Tabel 7.8 Kuesioner *System Usability Scale* (SUS)

No.	Pernyataan	Skor				
		Sangat Setuju				Sangat Tidak Setuju
		1	2	3	4	5
1	Saya pikir saya akan sering menggunakan aplikasi ini.					
2	Saya rasa aplikasi ini terlalu rumit, padahal bisa lebih sederhana.					
3	Menurut saya aplikasi ini mudah digunakan.					

4	Saya rasa saya akan membutuhkan bantuan ahli untuk menggunakan aplikasi ini.						
5	Menurut saya fitur-fitur dalam aplikasi ini telah terintegrasi dengan baik.						
6	Menurut saya terlalu banyak hal yang tidak konsisten dalam aplikasi ini.						
7	Menurut saya mayoritas pengguna akan mudah memahami aplikasi ini dengan baik.						
8	Saya rasa aplikasi ini terlalu sulit untuk digunakan.						
9	Saya sangat yakin dapat menggunakan aplikasi ini.						
10	Saya perlu belajar banyak hal agar bisa menggunakan aplikasi ini.						

7.3.1 Hasil Pengujian *Usability*

Hasil pengujian *usability* dapat dilihat pada Tabel 7.9.

Tabel 7.9 Hasil Pengujian *Usability*

Responden	Skor tiap pernyataan									
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1	3	2	4	1	4	3	4	2	5	1
2	4	2	4	2	4	3	5	1	5	1
3	4	2	5	2	4	1	4	1	5	2
4	5	1	5	1	3	2	5	1	5	2
5	5	2	5	2	4	2	4	2	5	2

7.4 Analisis Hasil Pengujian

7.4.1 Analisis Hasil Pengujian *Black Box*

Dari keempat kasus uji yang tersedia pada pengujian *black box*, seluruhnya menunjukkan status valid, sehingga dapat disimpulkan bahwa aplikasi pendeteksi kantuk telah memenuhi 100% dari kebutuhan fungsional yang telah ditentukan.

7.4.2 Analisis Hasil Pengujian Akurasi

Berdasarkan hasil pengujian akurasi yang dilakukan terhadap lima orang responden dengan total durasi selama 56 menit, didapatkan jumlah pengujian sebanyak 22 pengujian. Rekapitulasi hasil pengujian akurasi tertera pada Tabel 7.10.

Tabel 7.10 Rekapitulasi Hasil Pengujian Akurasi

Responden	Jumlah Pengujian	Jumlah Valid	Jumlah Tidak Valid
1	5	4	1
2	3	3	0
3	4	3	1
4	4	4	0
5	6	5	1
Total	**22**	**19**	**3**

Tingkat akurasi dapat dihitung dengan rumus sebagai berikut.

$$\frac{\sum Pengujian\ valid}{\sum Jumlah\ pengujian}\ x\ 100\%$$

Tingkat akurasi berdasarkan hasil pengujian adalah sebagai berikut.

$$\frac{19}{22}\ x\ 100\% = 86{,}3\%$$

Sehingga dapat ditarik kesimpulan berdasarkan hasil pengujian terhadap lima responden, aplikasi pendeteksi kantuk memiliki tingkat akurasi sebesar 86,3%.

7.4.3 Analisis Hasil Pengujian *Usability*

Untuk mendapatkan hasil akhir SUS, hal yang pertama harus dilakukan ialah mengonversi skor dari responden dengan cara mengurangi skor dengan 1 untuk pertanyaan bernomor ganjil, dan mengurangi 5 dengan skor untuk pertanyaan bernomor genap. Selanjutnya keseluruhan hasil yang didapat untuk tiap responden dijumlahkan. Hasil perhitungan tertera pada

Tabel 7.11 Hasil Konversi Skor Responden SUS

Responden	Konversi skor										Total
	1	2	3	4	5	6	7	8	9	10	
1	2	3	3	4	3	2	3	3	4	4	31
2	3	3	3	3	3	2	4	4	4	4	33
3	3	3	4	3	3	4	3	4	4	3	34
4	4	4	4	4	2	3	4	4	4	3	36
5	4	3	4	3	3	3	3	3	4	3	33
Total											167

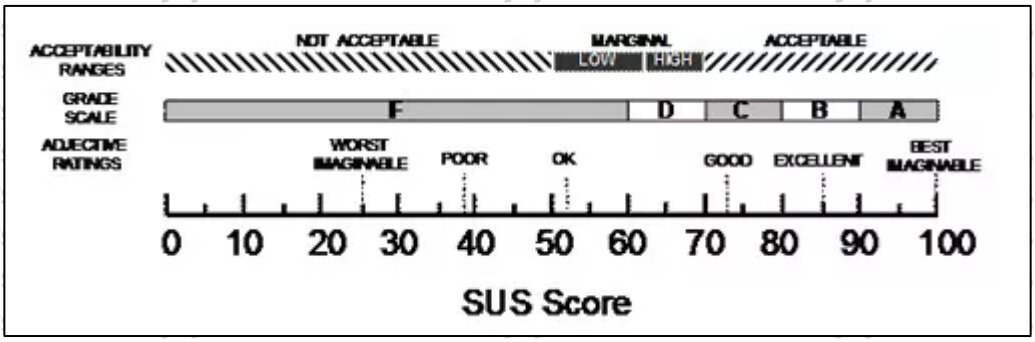
Kemudian skor akhir SUS dapat dihitung dengan rumus berikut.

$$\frac{\sum Konversi\ skor\ x\ 2{,}5}{\sum Responden}$$

Sehingga didapatkan skor akhir SUS :

$$\frac{167\ x\ 2{,}5}{5} = 83{,}5$$

Skor akhir SUS dari pengujian *usability* aplikasi pendeteksi kantuk adalah 83,5, yang mana termasuk ke dalam skala B *(Excellent)*. Interpretasi skor SUS dapat dilihat pada Gambar 7.1.



Gambar 7.1 Interpretasi Skor SUS

Sumber : Bangor, Kortum dan Miller, 2009

BAB 8 PENUTUP

Pada bab ini dipaparkan kesimpulan serta saran dari keseluruhan hasil penelitian yang telah dilakukan. Kesimpulan dan saran yang didapat diharapkan dapat membantu dalam penelitian selanjutnya.

8.1 Kesimpulan

Dari serangkaian tahap penelitian yang telah dilakukan yang terdiri dari analisis kebutuhan, perancangan, implementasi, dan pengujian, dapat ditarik kesimpulan sebagai berikut.

1. Berdasarkan hasil analisis kebutuhan, aplikasi pendeteksi kantuk memiliki empat kebutuhan fungsional, yaitu mendapatkan alarm peringatan, mengirimkan pesan darurat, menyimpan nomor darurat, dan melihat histori, serta memiliki dua kebutuhan non fungsional yaitu *reliability* dan *usability*.
2. Berdasaran hasil perancangan, diperoleh arsitektur sistem yang terdiri dari *smartwatch* sebagai media pengukur detak jantung dan perangkat Android untuk menjalankan aplikasi pendeteksi kantuk. Data dalam aplikasi pendeteksi kantuk dirancang dalam dua tabel yang disimpan dalam *database* lokal. Perancangan menggunakan metode *Object Oriented Design* (OOD) yang terbagi ke dalam dua tahap yaitu perancangan arsitektur dan perancangan antarmuka. Perancangan arsitektur menghasilkan *activity diagram, sequence diagram,* dan *class diagram*, sedangkan perancangan antarmuka menghasilkan *wireframe* tampilan antarmuka aplikasi pendeteksi kantuk.
3. Implementasi aplikasi pendeteksi kantuk menerapkan metode *Object Oriented Programming* (OOP) dengan menggunakan bahasa pemrograman Kotlin. Tahap implementasi menghasilkan aplikasi pendeteksi kantuk berbasis Android yang memuat fitur-fitur sesuai dengan yang telah didefinisikan pada tahap perancangan.
4. Berdasarkan hasil pengujian *black box*, aplikasi pendeteksi kantuk telah memenuhi seluruh kebutuhan fungsional yang ditentukan, sehingga memiliki tingkat validitas sebesar 100%. Berdasarkan hasil pengujian akurasi yang dilakukan terhadap lima responden, aplikasi pendeteksi kantuk memiliki tingkat akurasi pendeteksian kantuk sebesar 86,3%. Sedangkan berdasarkan hasil pengujian *usability,* aplikasi pendeteksi kantuk mendapatkan skor SUS 83,5 yang termasuk ke dalam skala B *(Excellent).*

8.2 Saran

Berdasarkan hasil penelitian yang diperoleh, penulis memberikan beberapa saran untuk penelitian selanjutnya, yaitu :

1. Untuk dapat mendeteksi kantuk, metode yang digunakan pada penelitian ini ialah *physiological measures*, yaitu mengukur dari sisi psikologis

pengendara, dalam hal ini detak jantung. Agar dapat meningkatkan akurasi hasil pendeteksian, disarankan untuk mengombinasikan dengan metode lainnya seperti *vehicle-based measures* misalnya dengan menambahkan parameter kecepatan kendaraan, atau dengan metode *behavioral measures* misalnya dengan memonitor pergerakan pengendara.

2. Aplikasi pendeteksi kantuk yang dikembangkan pada penelitian ini hanya dapat berfungsi jika dikoneksikan dengan *smartwatch* Samsung dengan sistem operasi Tizen. Agar dapat menjangkau lebih banyak pengguna, untuk penelitian selanjutnya disarankan agar dikembangkan pada sistem operasi *wearable* lainnya seperti Google Wear yang digunakan oleh lebih banyak manufaktur.

DAFTAR REFERENSI

Åkerstedt, T. dan Gillberg, M., 1990. Subjective dan Objective Sleepiness in the Active Individual. *International Journal of Neuroscience*, 52(1–2), pp.29–37.

Bangor, A., Kortum, P. dan Miller, J., 2009. Determining What Individual SUS Scores Mean: Adding an Adjective Rating Scale. *J. Usability Studies*, 4(3), pp.114–123.

Brodbeck, V., Kuhn, A., von Wegner, F., Morzelewski, A., Tagliazucchi, E., Borisov, S., Michel, C.M. dan Laufs, H., 2012. EEG microstates of wakefulness dan NREM sleep. *NeuroImage*, 62(3), pp.2129–2139.

Bumgardner, W., 2019. *What Is a Heart Rate Monitor?* [online] Verywell Fit. Tersedia di: <https://www.verywellfit.com/heart-rate-monitor-3436583> [Diakses 24 Mar. 2019].

Callaham, J., 2018. *The history of Android OS: its name, origin dan more*. [online] Android Authority. Tersedia di: <https://www.androidauthority.com/history-android-os-name-789433/> [Diakses 24 Mar. 2019].

Dumas, J.S., Dumas, J.S. dan Redish, J., 1999. *A Practical Guide to Usability Testing*. Intellect Books.

Elprocus, 2013. *Heart Beat Sensor - How to Measure Heart Beat: Working dan Application*. [online] ElProCus - Electronic Projects for Engineering Students. Tersedia di: <https://www.elprocus.com/heartbeat-sensor-working-application/> [Diakses 24 Mar. 2019].

Experience UX, 2019. *What Is Usability Testing?* [online] Tersedia di: <https://www.experienceux.co.uk/faqs/what-is-usability-testing/> [Diakses 10 Dec. 2019].

Fan, X., Yin, B. dan Sun, Y., 2007. Yawning Detection for Monitoring Driver Fatigue. In: *2007 International Conference on Machine Learning dan Cybernetics*. 2007 International Conference on Machine Learning dan Cybernetics. pp.664–668.

Harvard Health Publishing, 2019. *What your heart rate is telling you*. [online] Harvard Health. Tersedia di: <https://www.health.harvard.edu/heart-health/what-your-heart-rate-is-telling-you> [Diakses 24 Mar. 2019].

Holst, S.C. dan Landolt, H.-P., 2015. Sleep Homeostasis, Metabolism, dan Adenosine. *Current Sleep Medicine Reports*, 1(1), pp.27–37.

Liu, C.C., Hosking, S.G. dan Lenné, M.G., 2009. Predicting driver drowsiness using vehicle measures: Recent insights dan future challenges. *Journal of Safety Research*, 40(4), pp.239–245.

MacGill, M., 2017. *Heart rate: What is a normal heart rate?* [online] Medical News Today. Tersedia di: <https://www.medicalnewstoday.com/articles/235710.php> [Diakses 24 Mar. 2019].

MedicineNet, 2019. *Medical Definition of Heart rate*. [online] MedicineNet. Tersedia di: <https://www.medicinenet.com/script/main/art.asp?articlekey=3674> [Diakses 24 Mar. 2019].

Miley, A.Å., Kecklund, G. dan Åkerstedt, T., 2016. Comparing two versions of the Karolinska Sleepiness Scale (KSS). *Sleep dan Biological Rhythms*, 14(3), p.257.

Nidhra, S. dan Dondeti, J., 2012. Black box dan white box testing techniques-a literature review. *International Journal of Embedded Systems dan Applications (IJESA)*, 2(2), pp.29–50.

Owens, J.M., Dingus, T.A., Guo, F., Fang, Y., Perez, M., McClafferty, J. dan Tefft, B., 2018. Prevalence of Drowsy-Driving Crashes: Estimates from a Large-Scale Naturalistic Driving Study.

Oxford Press, 2019. *smartwatch | Definition of smartwatch in English by Oxford Dictionaries*. [online] Oxford Dictionaries | English. Tersedia di: <https://en.oxforddictionaries.com/definition/smartwatch> [Diakses 24 Mar. 2019].

Palatini, P., 2010. Recommendations on how to measure resting heart rate. *Medicographia*, 31(4), pp.414–419.

Peters, B., 2019. *Definition dan Possible Causes of Sleepiness?* [online] Verywell Health. Tersedia di: <https://www.verywellhealth.com/what-is-sleepiness-and-what-are-the-most-common-causes-3014824> [Diakses 26 Sep. 2019].

Purnamasari, P.D. dan Hazmi, A.Z., 2018. Heart Beat Based Drowsiness Detection System for Driver. *2018 International Seminar on Application for Technology of Information dan Communication*, pp.585–590.

Rouse, M., 2019. *Smartwatch*. [online] IoT Agenda. Tersedia di: <https://internetofthingsagenda.techtarget.com/definition/smartwatch> [Diakses 24 Mar. 2019].

Sahayadhas, A., Sundaraj, K. dan Murugappan, M., 2012. Detecting Driver Drowsiness Based on Sensors: A Review. *Sensors (Basel, Switzerland)*, 12(12), pp.16937–16953.

Samsung Developers, 2018. *Accessory | SAMSUNG Developers*. [online] Tersedia di: <https://developer.samsung.com/galaxy/accessory> [Diakses 26 Sep. 2019].

Techopedia, 2019. *What is Android OS? - Definition from Techopedia*. [online] Techopedia.com. Tersedia di: <https://www.techopedia.com/definition/14873/android-os> [Diakses 24 Mar. 2019].

Tizen, 2019. *About | Tizen*. [online] Tersedia di: <https://www.tizen.org/about> [Diakses 24 Mar. 2019].

Toban, R. dan Finandhita, A., 2017. PEMBANGUNAN APLIKASI PENDETEKSI KANTUK BERBASIS ANDROID. *Jurnal Ilmiah Komputer dan Informatika (KOMPUTA)*.

Valencell, T., 2015. *Valencell | Optical Heart Rate Monitoring: What You Need to Know*. [online] Valencell. Tersedia di: <https://valencell.com/blog/2015/10/optical-heart-rate-monitoring-what-you-need-to-know/> [Diakses 24 Mar. 2019].

Waldeck, M.R. dan Lambert, M.I., 2003. Heart Rate During Sleep: Implications for Monitoring Training Status. *Journal of Sports Science & Medicine*, 2(4), pp.133–138.

WHO, 2018a. *Road traffic injuries*. [online] Tersedia di: <https://www.who.int/news-room/fact-sheets/detail/road-traffic-injuries> [Diakses 24 Mar. 2019].

WHO, 2018b. *The top 10 causes of death*. [online] The top 10 causes of death. Tersedia di: <https://www.who.int/news-room/fact-sheets/detail/the-top-10-causes-of-death> [Diakses 24 Mar. 2019].

Zakiyyatuddin, A., Aldo, D.A. dan Ramelan, A., 2018. Wristband Inovatif Penghilang Kantuk Saat Belajar dengan Sensor Detak Jantung Berbasis IOTc. *JIIF (Jurnal Ilmu dan Inovasi Fisika)*, 2(2), pp.108–116.

LAMPIRAN A KUESIONER PENGUJIAN *USABILITY*

1.

Nama Responden : Ageng Muhammad Gemintang

Usia : 21 tahun

No.	Pernyataan	Skor				
		Sangat Setuju				Sangat Tidak Setuju
		1	2	3	4	5
1	Saya pikir saya akan sering menggunakan aplikasi ini.			✓		
2	Saya rasa aplikasi ini terlalu rumit, padahal bisa lebih sederhana.		✓			
3	Menurut saya aplikasi ini mudah digunakan.				✓	
4	Saya rasa saya akan membutuhkan bantuan ahli untuk menggunakan aplikasi ini.	✓				
5	Menurut saya fitur-fitur dalam aplikasi ini telah terintegrasi dengan baik.				✓	
6	Menurut saya terlalu banyak hal yang tidak konsisten dalam aplikasi ini.		✓			
7	Menurut saya mayoritas pengguna akan mudah memahami aplikasi ini dengan baik.				✓	
8	Saya rasa aplikasi ini terlalu sulit untuk digunakan.		✓			
9	Saya sangat yakin dapat menggunakan aplikasi ini.					✓
10	Saya perlu belajar banyak hal agar bisa menggunakan aplikasi ini.	✓				

2.

Nama Responden : Nyoman Putra Utama

Usia : 21 tahun

No.	Pernyataan	Skor				
		Sangat Setuju				Sangat Tidak Setuju
		1	2	3	4	5
1	Saya pikir saya akan sering menggunakan aplikasi ini.				✓	
2	Saya rasa aplikasi ini terlalu rumit, padahal bisa lebih sederhana.		✓			
3	Menurut saya aplikasi ini mudah digunakan.				✓	
4	Saya rasa saya akan membutuhkan bantuan ahli untuk menggunakan aplikasi ini.		✓			
5	Menurut saya fitur-fitur dalam aplikasi ini telah terintegrasi dengan baik.				✓	
6	Menurut saya terlalu banyak hal yang tidak konsisten dalam aplikasi ini.			✓		
7	Menurut saya mayoritas pengguna akan mudah memahami aplikasi ini dengan baik.					✓
8	Saya rasa aplikasi ini terlalu sulit untuk digunakan.	✓				
9	Saya sangat yakin dapat menggunakan aplikasi ini.					✓
10	Saya perlu belajar banyak hal agar bisa menggunakan aplikasi ini.	✓				

3.

Nama Responden : Wildan Oktavian

Usia : 20 tahun

No.	Pernyataan	Skor				
		Sangat Setuju				Sangat Tidak Setuju
		1	2	3	4	5
1	Saya pikir saya akan sering menggunakan aplikasi ini.				✓	
2	Saya rasa aplikasi ini terlalu rumit, padahal bisa lebih sederhana.		✓			
3	Menurut saya aplikasi ini mudah digunakan.					✓
4	Saya rasa saya akan membutuhkan bantuan ahli untuk menggunakan aplikasi ini.		✓			
5	Menurut saya fitur-fitur dalam aplikasi ini telah terintegrasi dengan baik.				✓	
6	Menurut saya terlalu banyak hal yang tidak konsisten dalam aplikasi ini.	✓				
7	Menurut saya mayoritas pengguna akan mudah memahami aplikasi ini dengan baik.				✓	
8	Saya rasa aplikasi ini terlalu sulit untuk digunakan.	✓				
9	Saya sangat yakin dapat menggunakan aplikasi ini.					✓
10	Saya perlu belajar banyak hal agar bisa menggunakan aplikasi ini.		✓			

4.

Nama Responden : Vivy Junita

Usia : 21 tahun

No.	Pernyataan	Skor				
		Sangat Setuju				Sangat Tidak Setuju
		1	2	3	4	5
1	Saya pikir saya akan sering menggunakan aplikasi ini.					✓
2	Saya rasa aplikasi ini terlalu rumit, padahal bisa lebih sederhana.	✓				
3	Menurut saya aplikasi ini mudah digunakan.					✓
4	Saya rasa saya akan membutuhkan bantuan ahli untuk menggunakan aplikasi ini.	✓				
5	Menurut saya fitur-fitur dalam aplikasi ini telah terintegrasi dengan baik.			✓		
6	Menurut saya terlalu banyak hal yang tidak konsisten dalam aplikasi ini.		✓			
7	Menurut saya mayoritas pengguna akan mudah memahami aplikasi ini dengan baik.					✓
8	Saya rasa aplikasi ini terlalu sulit untuk digunakan.	✓				
9	Saya sangat yakin dapat menggunakan aplikasi ini.					✓
10	Saya perlu belajar banyak hal agar bisa menggunakan aplikasi ini.		✓			

5.

Nama Responden : Reinhard Jonathan

Usia : 20 tahun

No.	Pernyataan	Skor				
		Sangat Setuju				Sangat Tidak Setuju
		1	2	3	4	5
1	Saya pikir saya akan sering menggunakan aplikasi ini.					✓
2	Saya rasa aplikasi ini terlalu rumit, padahal bisa lebih sederhana.		✓			
3	Menurut saya aplikasi ini mudah digunakan.					✓
4	Saya rasa saya akan membutuhkan bantuan ahli untuk menggunakan aplikasi ini.		✓			
5	Menurut saya fitur-fitur dalam aplikasi ini telah terintegrasi dengan baik.				✓	
6	Menurut saya terlalu banyak hal yang tidak konsisten dalam aplikasi ini.		✓			
7	Menurut saya mayoritas pengguna akan mudah memahami aplikasi ini dengan baik.				✓	
8	Saya rasa aplikasi ini terlalu sulit untuk digunakan.		✓			
9	Saya sangat yakin dapat menggunakan aplikasi ini.					✓
10	Saya perlu belajar banyak hal agar bisa menggunakan aplikasi ini.		✓			

LAMPIRAN B *FORM* PENGUJIAN AKURASI

1.

Nama Responden : Ageng Muhammad Gemintang

Usia : 21 tahun

Menit	Tingkat Kantuk	Alarm Menyala (Y/T)
5	5	T
10	5	Y
13	7	Y
14	7	Y
15	6	T

2.

Nama Responden : Nyoman Putra Utama

Usia : 21 tahun

Menit	Tingkat Kantuk	Alarm Menyala (Y/T)
5	4	T
10	5	T
15	6	T

3.

Nama Responden : Wildan Oktavian

Usia : 20 tahun

Menit	Tingkat Kantuk	Alarm Menyala (Y/T)
2	4	Y
5	4	T
10	6	T
15	7	Y
15	6	T

4.

Nama Responden : Vivy Junita

Usia : 21 tahun

Menit	Tingkat Kantuk	Alarm Menyala (Y/T)
1	7	Y
2	7	Y
4	7	Y
5	6	T

5.

Nama Responden : Reinhard Jonathan

Usia : 20 tahun

Menit	Tingkat Kantuk	Alarm Menyala (Y/T)
1	7	Y
1	7	Y
3	8	Y
4	8	Y
5	8	T
6	8	Y