BAB II
LANDASAN TEORI

2.1 Penelitian Terkait

Penelitian mengenai asisten virtual dan pengelolaan jadwal telah banyak dilakukan sebelumnya dengan berbagai pendekatan dan teknologi. Berikut adalah beberapa penelitian terkait yang menjadi referensi dalam penelitian ini:

2.1.1 Penelitian tentang Aplikasi *Reminder* Jadwal

Edde & Budayawan (2021) melakukan penelitian dengan judul "Pembuatan Aplikasi *Reminder* Jadwal Perkuliahan di Jurusan Teknik Elektronika Berbasis Bahasa *Python*". Penelitian ini menggunakan bahasa pemrograman Kotlin dan Firebase Realtime *Database* (FRD) sebagai *database* server. Aplikasi diberi nama NO-FORGET yang bertujuan membantu dosen dan mahasiswa mengingat jadwal perkuliahan yang dapat berubah secara mendadak. Metode yang digunakan adalah Unified Modeling Language (UML) sebagai bahasa visual untuk pemodelan berorientasi objek. Hasil pengujian menggunakan metode *Black Box Testing* menunjukkan bahwa aplikasi ini layak digunakan dalam membantu penggantian jadwal perkuliahan karena fitur yang ada sangat membantu dosen dan mahasiswa mengatur jadwal yang berhalangan. Teknologi FRD terbukti dapat diandalkan dalam membangun aplikasi yang menggunakan update data secara realtime, namun memiliki kelemahan dari sisi relasional *database*.

Veri Ilhadi et al. (2022) mengembangkan "Aplikasi *Reminder* Jadwal Kegiatan Berbasis *Window*" yang menggunakan bahasa JAVA dan dibuat menggunakan Android Studio dengan *database* lokal SQLite. Aplikasi ini memiliki fitur Pop Up *Reminder* yang dapat membuka otomatis perangkat dari user untuk memberi pengingat. Sistem menggunakan metode *SDLC (Software Development Life Cycle)* dan Waterfall dalam pengembangannya. Berdasarkan hasil penelitian, aplikasi *reminder* ini memudahkan dalam mengelola data jadwal dan

kegiatan harian yang menghasilkan suatu notifikasi *reminder* sebagai pengingat jadwal. Hasil

Amri Dika Irawan & Wahyu Sri U tami (2023) melakukan penelitian "Aplikasi *Reminder* Jadwal Kuliah dan Tugas Mahasiswa Berbasis Android". Penelitian ini menggunakan bahasa pemrograman Kotlin dengan Android Studio dan Firebase Realtime *Database* untuk penyimpanan data. Aplikasi ini dirancang untuk membantu mahasiswa Universitas Teknologi Yogyakarta dalam mengorganisir informasi perkuliahan dan mengingatkan mereka. Sistem memberikan pengingat dalam bentuk notifikasi kepada user beberapa waktu sebelum jadwal kuliah akan dimulai atau tugas sebelum batas deadline. Pengujian dilakukan menggunakan metode *Black Box Testing* dan hasilnya menunjukkan bahwa aplikasi dapat berjalan dengan semestinya dan memberikan *reminder* ke pengguna dalam bentuk notifikasi yang sudah diatur waktu notifikasinya sesuai kebutuhan pengguna.

2.1.2 Penelitian tentang Asisten Virtual

Chopra et al. (2023) melakukan penelitian tentang dimensi kualitas layanan pada *chatbot* berbasis AI yang mengarah pada kepuasan pelanggan di Asia Selatan. Penelitian ini menganalisis berbagai dimensi yang mempengaruhi kepuasan pengguna dalam menggunakan asisten virtual berbasis *chatbot*, termasuk responsivitas, akurasi, dan kemudahan penggunaan. Hasil penelitian menunjukkan bahwa kualitas interaksi dan kemampuan pemahaman bahasa natural menjadi faktor kunci dalam kepuasan pengguna.

Ehsani et al. (2023) melakukan analisis komparatif terhadap *chatbot* layanan pelanggan dengan fokus pada efisiensi, *usability*, dan aplikasi. Penelitian ini membandingkan berbagai platform *chatbot* dan menemukan bahwa asisten virtual yang memiliki antarmuka sederhana namun fungsional cenderung memiliki tingkat adopsi yang lebih tinggi dibandingkan dengan sistem yang kompleks. Studi ini menekankan

Beaver & Cynthia Freeman (2016) menjelaskan bahwa *Virtual Assistant* dapat diterapkan dalam berbagai bentuk seperti yang menggunakan audio, teks, gambar, *user interface*, kontrol, dan konten web sebagai media. Penelitian ini memberikan klasifikasi komprehensif tentang berbagai bentuk implementasi asisten virtual dan bagaimana masing-masing bentuk memiliki kelebihan dan kekurangan tergantung pada konteks penggunaannya.

2.1.3 Penelitian tentang Manajemen Jadwal

Cranshaw et al. (2017) mengembangkan Calendar.help, sebuah agen penjadwalan berbasis workflow dengan manusia dalam loop. Penelitian ini mengeksplorasi bagaimana asisten virtual dapat membantu pengguna dalam mengelola jadwal mereka dengan lebih efisien melalui automasi cerdas yang tetap melibatkan keputusan manusia. Hasil penelitian menunjukkan bahwa pendekatan hybrid antara otomasi dan kontrol manual dapat meningkatkan efektivitas manajemen waktu hingga 40% dibandingkan metode tradisional.

Bagus Setiawan & Hermansyah (2025) mengembangkan "Smart *Reminder*: Sistem Pengingat Tugas Berbasis ReactJS dan Google Apps Script untuk Manajemen Proyek Tim". Penelitian ini mengintegrasikan ReactJS sebagai frontend untuk *input* data, serta Google Apps Script dan Google Calendar sebagai backend. Sistem ini mampu mengotomatisasi penjadwalan dan pengingat tugas berbasis cloud dengan notifikasi sebelum deadline. Berdasarkan hasil survei kepuasan pengguna, 87% responden menyatakan sistem ini mempermudah pengelolaan tugas dan membantu dalam mengingat deadline. Formulir *input* berbasis ReactJS dinilai mudah digunakan dengan validasi data yang efektif dalam meminimalkan kesalahan *input*.

Annisa Zahwaa et al. (2025) melakukan penelitian "Analisis dan Perancangan Aplikasi Pengingat Jadwal Kuliah Mahasiswa STIKOM Tunas Bangsa". Penelitian ini menggunakan pendekatan deskriptif kualitatif dan model pengembangan perangkat lunak Waterfall.

Perancangan dilakukan dengan diagram *UML (use case, activity, dan class diagram)*. Hasil analisis menunjukkan bahwa kebutuhan utama mahasiswa meliputi keakuratan pengingat, antarmuka yang mudah diakses, serta integrasi fitur yang mendukung manajemen waktu secara menyeluruh. Aplikasi ini dirancang untuk meningkatkan efisiensi dan kedisiplinan mahasiswa dalam kegiatan akademik

2.1.4 Penelitian tentang *Natural Language Processing* untuk Bahasa Indonesia

Cahyawijaya et al. (2021) mengembangkan IndoNLG, sebuah benchmark dan resource untuk mengevaluasi Indonesian *Natural Language Generation*. Penelitian ini sangat relevan karena menyediakan dataset dan metodologi untuk mengembangkan sistem *NLP* berbahasa Indonesia. Tantangan utama yang diidentifikasi adalah kompleksitas tata bahasa Indonesia dan ketersediaan dataset yang terbatas dibandingkan dengan bahasa Inggris.

Putri et al. (2022) dalam penelitiannya tentang pengembangan asisten virtual lokal di Indonesia menemukan bahwa meskipun beberapa produk seperti Jala oleh Kata.ai dan Ponsel Pintar oleh Telkom Indonesia telah dikembangkan, penetrasi mereka masih terbatas. Hal ini mengindikasikan adanya ruang untuk pengembangan aplikasi asisten virtual yang lebih accessible dan sesuai dengan kebutuhan lokal.

2.1.5 Penelitian tentang Pengembangan Aplikasi dengan Kivy

Prihantoro Manahan Tobing & Magdalena Ariance Ineke Pakereng (2020) melakukan penelitian tentang pengembangan aplikasi cross-platform menggunakan *Kivy framework*. Penelitian ini menunjukkan bahwa *Kivy framework* tidak memerlukan spesifikasi komputer yang tinggi untuk mengoperasikannya serta dapat berjalan multi-platform, baik pada desktop maupun mobile. Hasil penelitian menunjukkan bahwa aplikasi yang dikembangkan dengan *Kivy* memiliki performa yang baik dan dapat berjalan pada berbagai perangkat dengan spesifikasi yang beragam.

2.1.6 Posisi Penelitian Ini

Berdasarkan penelitian-penelitian terdahulu, penelitian ini memiliki keunikan dalam hal pengembangan asisten virtual sederhana yang khusus dirancang untuk pengelolaan jadwal dengan fitur *Speech Recognition* berbahasa Indonesia. Berbeda dengan asisten virtual global yang kompleks dan ekosistem tertutup, penelitian ini fokus pada pengembangan aplikasi yang sederhana, mudah digunakan, dan dapat disesuaikan dengan kebutuhan pengguna lokal Indonesia. Penggunaan *Python* dengan *framework Kivy* dan *KivyMD* juga memberikan keunggulan dalam hal portabilitas dan kemudahan pengembangan.

2.2 Tinjauan Teoritis

2.2.1 Kecerdasan Buatan (*Artificial Intelligence*)

Kecerdasan buatan atau *Artificial Intelligence* (AI) dapat didefinisikan sebagai cabang ilmu komputer yang mempelajari otomisasi tingkah laku cerdas. Definisi AI pertama kali dikemukakan oleh John McCarthy pada tahun 1956 yang menyatakan bahwa AI adalah "the science and engineering of making intelligent machines", yang berarti ilmu dan teknik untuk membuat mesin yang cerdas.

AI bertujuan untuk menciptakan sistem komputer yang dapat melakukan tugas-tugas yang biasanya memerlukan kecerdasan manusia, seperti pengenalan suara, pengambilan keputusan, penerjemahan bahasa, dan pemecahan masalah. Sistem AI modern menggunakan berbagai teknik termasuk *machine learning*, *deep learning*, dan *Natural Language Processing* untuk mencapai tujuan ini.

Dalam konteks penelitian ini, AI digunakan sebagai dasar pengembangan asisten virtual yang dapat memahami perintah suara pengguna dan mengolahnya menjadi jadwal terstruktur. Kemampuan AI untuk memproses bahasa natural dan mengambil keputusan berdasarkan *input* pengguna menjadi kunci dalam implementasi aplikasi.

2.2.2 Asisten Virtual (*Virtual Assistant*)

Asisten virtual dapat didefinisikan sebagai program perangkat lunak yang mampu mengenali dan merespons *input* pengguna dalam bentuk bahasa alami, dan melakukan tugas-tugas tertentu berdasarkan *input* tersebut. Teknologi ini telah berkembang pesat sejak awal konseptualisasinya di tahun 1960-an dengan sistem ELIZA yang dikembangkan oleh Joseph Weizenbaum di MIT. ELIZA merupakan program komputer sederhana yang dapat bercakap-cakap dengan manusia melalui pola pencocokan kata kunci.

Perkembangan signifikan dalam teknologi asisten virtual mulai terlihat pada awal 2010-an dengan diperkenalkannya Siri oleh Apple pada tahun 2011, diikuti oleh Google Assistant (2016), Amazon Alexa (2014), dan Microsoft Cortana (2014). Era 2020-an kemudian ditandai dengan kemunculan asisten virtual yang lebih canggih dengan kemampuan pemahaman konteks yang lebih baik, seperti ChatGPT (2022) dan berbagai turunannya.

Kelahiran *Virtual Assistant* dimulai pada tahun 1990-an ketika muncul teknologi pengenalan suara digital hingga peluncuran ponsel cerdas pertama IBM Simon pada tahun 1994. *Virtual Assistant* basisnya menggunakan metode Natural Language Programming (*NLP*) yang memungkinkan sistem untuk memahami dan memproses bahasa manusia.

Komponen Utama Asisten Virtual:

1. *Speech Recognition*: Kemampuan untuk mengenali dan mengubah suara menjadi teks
2. *Natural Language Processing* (*NLP*): Kemampuan untuk memahami maksud dan konteks dari teks
3. *Dialog Management*: Kemampuan untuk mengelola percakapan dan konteks interaksi

4. *Text to Speech*: Kemampuan untuk mengubah teks menjadi suara (opsional)
5. *Knowledge Base*: *Database* pengetahuan yang digunakan untuk memberikan respons

Virtual Assistant dapat diterapkan dalam berbagai bentuk seperti yang menggunakan audio, teks, gambar, *user interface*, kontrol, dan konten web sebagai media. Dalam penelitian ini, asisten virtual difokuskan pada pengelolaan jadwal dengan *input* suara berbahasa Indonesia.

2.2.3 *Natural Language Processing* (NLP)

Natural Language Processing (*NLP*) adalah cabang dari kecerdasan buatan yang berfokus pada interaksi antara komputer dan bahasa manusia. *NLP* memungkinkan komputer untuk memahami, menginterpretasi, dan menghasilkan bahasa manusia dengan cara yang bermakna dan berguna.

Komponen Utama *NLP*:

1. *Tokenization*: Proses memecah teks menjadi unit-unit kecil seperti kata atau frasa
2. *Part-of-Speech Tagging:* Mengidentifikasi jenis kata (kata benda, kata kerja, dll.)
3. *Named Entity Recognition*: Mengidentifikasi entitas seperti nama orang, tempat, tanggal
4. *Sentiment Analysis*: Menganalisis sentimen atau emosi dalam teks
5. *Intent Recognition*: Memahami maksud atau tujuan dari *input* pengguna

Dalam konteks aplikasi asisten virtual untuk pengelolaan jadwal, *NLP* digunakan untuk memahami perintah pengguna seperti "buatkan jadwal rapat besok jam 2 siang" dan mengekstrak informasi penting seperti jenis kegiatan (rapat), waktu (besok jam 2 siang), untuk kemudian disimpan sebagai data jadwal terstruktur.

2.2.4 *Speech Recognition*

Speech Recognition atau pengenalan suara adalah teknologi yang memungkinkan komputer untuk mengidentifikasi dan memproses suara manusia menjadi teks. Teknologi ini merupakan komponen kunci dalam pengembangan asisten virtual yang dapat menerima input suara.

Proses *Speech Recognition*:

1. Audio *Input*: Menangkap suara melalui mikrofon
2. Preprocessing: Membersihkan noise dan normalisasi audio
3. Feature Extraction: Mengekstrak fitur-fitur penting dari sinyal audio
4. Pattern Recognition: Mencocokkan pola suara dengan model bahasa
5. Text *Output*: Menghasilkan teks dari suara yang dikenali

Library SpeechRecognition *Python* yang digunakan dalam penelitian ini menyediakan antarmuka sederhana untuk berbagai mesin *Speech Recognition*, termasuk Google *Speech Recognition* yang mendukung Bahasa Indonesia. *Library* ini memungkinkan aplikasi untuk mengkonversi *input* suara pengguna menjadi teks yang kemudian dapat diproses lebih lanjut.

Tantangan *Speech Recognition* untuk Bahasa Indonesia:

1. Variasi dialek dan aksen yang beragam
2. Keterbatasan dataset training berbahasa Indonesia
3. Kompleksitas struktur kalimat bahasa Indonesia
4. Pengaruh noise lingkungan terhadap akurasi pengenalan

2.2.5 Python Programming Language

Python adalah bahasa pemrograman tingkat tinggi yang bersifat interpreted, object-oriented, dan memiliki sintaks yang sederhana dan mudah dipahami. *Python* dikembangkan oleh Guido van Rossum dan pertama kali dirilis pada tahun 1991. *Python* menjadi salah satu bahasa pemrograman paling populer karena kesederhanaan, fleksibilitas, dan ekosistem *library* yang sangat luas.

Keunggulan *Python*:

1. Sintaks yang Sederhana: Mudah dipelajari dan dipahami bahkan untuk pemula
2. *Cross-Platform*: Dapat berjalan di berbagai sistem operasi
3. *Library* yang Lengkap: Memiliki ribuan *library* untuk berbagai keperluan
4. *Community Support*: Komunitas yang besar dan aktif
5. *Rapid Development*: Memungkinkan pengembangan aplikasi dengan cepat

Python sangat cocok untuk pengembangan aplikasi asisten virtual karena memiliki *library* yang mendukung AI, *machine learning*, dan pemrosesan bahasa natural. Dalam penelitian ini, *Python* digunakan sebagai bahasa pemrograman utama dengan memanfaatkan berbagai *library* seperti *Kivy*, *KivyMD*, dan SpeechRecognition.

2.2.6 Kivy *Framework*

Kivy adalah *library* open-source dari *Python* yang dapat digunakan untuk membangun aplikasi cross-platform. *Kivy* memungkinkan aplikasi *Python* dapat berjalan pada berbagai sistem operasi seperti Windows, Linux dan macOS dengan menggunakan satu codebase yang sama.

Karakteristik *Kivy*:

1. *Cross-Platform*: Satu kode dapat dijalankan di berbagai platform
2. *Open Source*: Gratis dan dapat dikustomisasi sesuai kebutuhan
3. *Multi-Touch Support*: Mendukung *input* multi-touch untuk perangkat mobile
4. *GPU Accelerated*: Menggunakan akselerasi GPU untuk rendering yang cepat
5. *Flexible Architecture:* Arsitektur yang fleksibel dan mudah diperluas

Kivy framework tidak memerlukan spesifikasi komputer yang tinggi untuk mengoperasikannya serta dapat berjalan multi-platform, baik pada

desktop maupun mobile. Hal ini menjadikan *Kivy* sebagai pilihan yang tepat untuk pengembangan aplikasi asisten virtual yang perlu berjalan di berbagai perangkat dengan spesifikasi yang beragam.

Komponen Utama *Kivy*:

1. Widget: Elemen UI dasar seperti button, label, text *input*
2. Layout: Pengaturan tata letak widget seperti BoxLayout, GridLayout
3. Properties: Sistem binding data yang reaktif
4. Events: Sistem event handling untuk interaksi pengguna
5. Graphics: API untuk rendering grafis custom

2.2.7 KivyMD (Kivy Material Design)

KivyMD adalah *framework* tambahan untuk *Kivy* yang menyediakan komponen-komponen Material Design untuk memudahkan dalam mendesain tampilan aplikasi. Material Design adalah sistem desain yang dikembangkan oleh Google yang menekankan pada penggunaan grid-based layouts, responsive animations, dan depth effects seperti lighting dan shadows.

Keunggulan *KivyMD*:

1. *Material Design Components*: Menyediakan widget yang mengikuti guideline Material Design
2. *Responsive Design:* Tampilan yang adaptif terhadap berbagai ukuran layar
3. *Pre-built Widgets*: Komponen UI yang sudah jadi dan siap pakai
4. *Theming System:* Sistem tema yang mudah dikustomisasi
5. *Consistent Look:* Tampilan yang konsisten dan modern

KivyMD memungkinkan developer untuk membuat aplikasi dengan tampilan yang profesional dan modern tanpa harus mendesain dari awal. Komponen-komponen seperti MDTextField, MDButton, MDCard, dan MDDialog sudah tersedia dan dapat langsung digunakan dengan styling Material Design yang menarik.

2.2.8 *Database Management*

Database management adalah proses penyimpanan, pengorganisasian, dan pengelolaan data secara terstruktur. Dalam aplikasi asisten virtual untuk pengelolaan jadwal, *database* digunakan untuk

Jenis *Database* yang Umum Digunakan:

1. *SQLite: Database* relasional yang ringan dan tidak memerlukan server terpisah
2. *MySQL/PostgreSQL*: *Database* relasional untuk aplikasi skala besar
3. *MongoDB*: *Database* NoSQL untuk data yang tidak terstruktur
4. *Firebase*: *Database* cloud untuk aplikasi mobile

Untuk aplikasi jadwal seperti yang dikembangkan dalam penelitian ini, *Firebase* merupakan platform pengembangan aplikasi berbasis cloud yang dikembangkan dan dikelola oleh Google. Platform ini hadir sebagai solusi bagi para pengembang yang ingin membangun aplikasi dengan fitur backend yang lengkap tanpa harus repot mengelola server sendiri. Singkatnya, Firebase memungkinkan pengembang untuk lebih fokus pada pengembangan fitur dan tampilan aplikasi karena urusan infrastruktur sudah ditangani oleh Google.

Salah satu layanan utama Firebase yang paling banyak digunakan adalah Firebase Realtime *Database* dan Cloud Firestore. Keduanya merupakan jenis *database* NoSQL yang menyimpan data dalam format JSON. Yang menjadi keunggulan utamanya adalah kemampuan sinkronisasi data secara real-time, artinya setiap perubahan data akan langsung tercermin di semua perangkat yang terhubung tanpa perlu melakukan permintaan ulang ke server. Hal ini tentu sangat bermanfaat untuk aplikasi yang membutuhkan pembaruan data yang cepat dan responsif.

Selain layanan *database*, Firebase juga dilengkapi dengan berbagai fitur pendukung lainnya seperti Firebase Authentication untuk mengelola

akun dan login pengguna, Firebase Storage untuk menyimpan file seperti gambar atau dokumen, serta Firebase Cloud Messaging (FCM) yang berguna untuk mengirimkan notifikasi ke perangkat pengguna. Semua

Dalam penelitian ini, Firebase digunakan sebagai media penyimpanan data jadwal pengguna secara cloud. Alasan pemilihan Firebase antara lain karena kemudahannya dalam diintegrasikan dengan Python menggunakan *library* pyrebase atau firebase-admin, kemampuan sinkronisasi data secara real-time, serta tidak memerlukan pengaturan server yang rumit. Dengan Firebase, data jadwal pengguna dapat tersimpan dengan aman dan dapat diakses dari mana saja selama perangkat terhubung ke internet.

2.2.9 *Black Box Testing*

Black Box Testing adalah metode pengujian perangkat lunak yang berfokus pada fungsionalitas aplikasi tanpa melihat struktur internal kode. Pengujian dilakukan dengan memberikan *input* dan memverifikasi apakah *output* yang dihasilkan sesuai dengan yang diharapkan.

Karakteristik *Black Box Testing*:

1. *Functional Testing*: Menguji fungsi-fungsi aplikasi sesuai spesifikasi
2. *User Perspective*: Pengujian dari sudut pandang pengguna
3. *No Code Knowledge*: Tidak memerlukan pengetahuan tentang kode program
4. *Input-Output Verification*: Fokus pada validasi *input* dan *output*

Teknik *Black Box Testing*:

1. *Equivalence Partitioning:* Membagi *input* menjadi kelompok yang valid dan tidak valid
2. *Boundary Value Analysis:* Menguji nilai batas dari *input*
3. *Decision Table Testing:* Menggunakan tabel keputusan untuk berbagai kombinasi *input*

State Transition *Testing*: Menguji perubahan state aplikasi

Dalam penelitian ini, *Black Box Testing* digunakan untuk memastikan semua fitur aplikasi seperti *input* jadwal manual, *input* jadwal dengan

2.2.10 *System Usability Scale* (SUS)

System Usability Scale (SUS) adalah metode standar untuk mengukur *usability* atau kemudahan penggunaan suatu sistem. SUS dikembangkan oleh John Brooke pada tahun 1986 dan telah menjadi salah satu metode paling populer untuk evaluasi *usability* karena kesederhanaan dan reliabilitasnya.

Karakteristik SUS:

1. Quick and Easy: Cepat dan mudah dilakukan
2. Standardized: Menggunakan kuesioner standar dengan 10 pertanyaan
3. Reliable: Memiliki reliabilitas yang tinggi
4. Valid: Hasil yang valid untuk berbagai jenis sistem
5. Comparable: Dapat membandingkan *usability* antar sistem

Kuesioner SUS:

1. SUS menggunakan 10 pertanyaan dengan skala Likert 1-5 (Sangat Tidak Setuju hingga Sangat Setuju):
2. Saya pikir saya akan sering menggunakan sistem ini
3. Saya merasa sistem ini terlalu kompleks
4. Saya pikir sistem ini mudah digunakan
5. Saya memerlukan bantuan dari orang teknis untuk menggunakan sistem ini
6. Saya merasa berbagai fungsi dalam sistem ini terintegrasi dengan baik
7. Saya pikir ada terlalu banyak inkonsistensi dalam sistem ini
8. Saya membayangkan kebanyakan orang akan belajar menggunakan sistem ini dengan cepat
9. Saya merasa sistem ini sangat rumit untuk digunakan
10. Saya merasa sangat percaya diri menggunakan sistem ini

11. Saya perlu belajar banyak hal sebelum saya bisa menggunakan sistem ini

Perhitungan Skor SUS:

Skor SUS dihitung dengan rumus tertentu dan menghasilkan nilai antara 0-100. Interpretasi skor:

- 0-25: Worst Imaginable (Terburuk)
- 26-39: Poor (Buruk)
- 40-52: OK (Cukup)
- 53-73: Good (Baik)
- 74-85: Excellent (Sangat Baik)
- 86-100: Best Imaginable (Terbaik)

Skor rata-rata SUS untuk sistem yang baik adalah 68. Dalam penelitian ini, SUS digunakan untuk mengevaluasi tingkat kemudahan penggunaan aplikasi asisten virtual dari perspektif pengguna.