PENGEMBANGAN PROGRAM *VIRTUAL ASSISTANT* DENGAN MENGGUNAKAN PYTHON

NAMA : DWIKI ALDANI

NPM : 18360028

PROGRAM STUDI TEKNIK INFORMATIKA

FAKULTAS SAINS DAN TEKNOLOGI INFORMASI

INSTITUT SAINS DAN TEKNOLOGI NASIONAL

JAKARTA

MARET 2022

PENGEMBANGAN PROGRAM *VIRTUAL ASSISTANT* DENGAN MENGGUNAKAN PYTHON

SKRIPSI

Diajukan sebagai salah satu syarat untuk memperoleh gelar Sarjana Komputer (S.Kom)

NAMA : DWIKI ALDANI

NPM : 18360028

PROGRAM STUDI TEKNIK INFORMATIKA

FAKULTAS SAINS DAN TEKNOLOGI INFORMASI

INSTITUT SAINS DAN TEKNOLOGI NASIONAL

JAKARTA

MARET 2022

HALAMAN PERNYATAAN ORISINALITAS

Skripsi ini adalah hasil karya saya sendiri, dan semua sumber baik yang dikutip maupun dirujuk telah saya nyatakan dengan benar.

Nama : Dwiki Aldani

NPM : 18360028

Tanggal : 02 Maret 2022

Jakarta, 02 Maret 2022

Yang menyatakan

Dwiki Aldani

HALAMAN PERNYATAAN NON PLAGIAT

Saya yang bertanda tangan dibawah ini:

Nama : Dwiki Aldani

NPM : 18360028

Program Studi : Teknik Informatika

Tahun Akademik : 2018

Menyatakan bahwa saya tidak melakukan kegiatan plagiat dalam penulisan Tugas Akhir yang berjudul **"Pengembangan Program *Virtual Assistant Dengan* Menggunakan Python".**

Apabila suatu saat nanti terbukti saya melakukan plagiat, maka saya akan menerima sanksi yang telah ditetapkan.

Demikian Surat Pernyataan ini saya buat dengan sebenar-benarnya.

Jakarta, 02 Maret 2022

Dwiki Aldani

HALAMAN PENGESAHAN

Skripsi ini diajukan oleh:

Nama	:	Dwiki Aldani
NPM	:	18360028
Program Studi	:	Teknik Informatika
Judul Skripsi	:	Pengembangan Program *Virtual Assistant* Dengan Menggunakan Python

Telah berhasil dipertahankan di hadapan Dewan Penguji dan diterima sebagai bagian persyaratan yang diperlukan untuk memperoleh gelar Sarjana Komputer pada Program Studi Tenkik Informatika, Fakultas Sains dan Teknologi Informasi, Institut Sains dan Teknologi Nasional.

DEWAN PENGUJI

Pembimbing	: Siti Nurmiati, S.Kom., M.Kom.	( )
Penguji	: Neny Rosmawarni, S.Kom., M.Kom.	( )
Penguji	: Aryo Nur Utomo, S.T., M.Kom.	( )
Penguji	: Siti Madinah L., S.Kom., M.Kom.	( )

Ditetapkan di : Jakarta

Tanggal : 02 Maret 2022

KATA PENGANTAR

Puji dan syukur penulis panjatkan kehadirat Allah SWT yang telah melimpahkan rahmat dan hidayah-Nya, Sehingga penulis dapat menyalesaikan skripsi yang berjudul "Pengembangan Program *Virtual Assistant* Dengan Menggunakan Python" dengan sebaik-baiknya.

Skripsi ini dibuat sebagai salah satu syarat guna memperoleh gelar Sarjana Komputer (S.Kom) Program Studi Teknik Informatika pada Fakultas Sains dan Teknologi Informasi di Institut Sains dan Teknologi Nasional.

Penulis menyadari bahwa tanpa bantuan, bimbingan, dan dukungan dari semua pihak, maka penulisan skripsi ini tidak akan berjalan lancar. Oleh karena itu pada kesempatan ini penulis mengucapkan terimakasih kepada:

1. Ibu Marhaeni S.Kom, M.Kom, selaku Dekan Fakultas Sains dan Teknologi Informasi, Institut Sains dan Teknologi Nasional.
2. Ibu Neny Rosmawarni S.Kom, M.Kom, selaku Kepala Program Studi Teknik Informatika, Institut Sains dan Teknologi Nasional.
3. Ibu Siti Nurmiati, S.Kom, M.Kom, Selaku dosen pembimbing yang telah meluangkan waktu, tenaga, dan pikirannya dalam memberikan bimbingan serta arahan kepada penulis dalam menyusun skripsi ini.
4. Bapak dan Ibu Dosen Program Studi Teknik Informatika, Institut Sains dan Teknologi Nasional.
5. Andi Amrin dan Siti Sumarni orang tua tercinta yang selalu memberikan semangat dan doa sehingga dapat dengan semangat mengerjakan skripsi ini.
6. Magus Tiandar Putra dan Martha Tiffani keluarga tercinta yang memberikan arahan dan selalu memberikan dukungan.
7. Sefta Inda Rauhillah yang telah memberikan semangat dan motivasi dalam mengerjakan skripsi ini.
8. Ahmad Nur Hasan Ghozy, Aras Harnas, Andhika Wichaksono Putra F., Rakha Tri Fadillah, Ibnu Wanafa, Venerdi Rafid Muhajir teman yang menemani dan membantu menyelesaikan skripsi ini.
9. Keluarga besar tercinta yang telah memberikan dukungan baik moril maupun materil.

10. Teman – teman seperjuangan yang telah memberikan dukungan sehingga dapat memberikan semangat.

Akhir kata, saya berharap Tuhan Yang Maha Esa berkenan membalas segala kebaikan semua pihak yang telah membantu. Semoga skripsi ini membawa manfaat bagi pengembangan ilmu pengetahuan.

Jakarta, 02 Maret 2022
Penulis

Dwiki Aldani

HALAMAN PERNYATAAN PERSETUJUAN PUBLIKASI TUGAS AKHIR UNTUK KEPENTINGAN AKADEMIS

Sebagai sivitas akademik Institut Sains dan Teknologi Nasional, saya yang bertanda tangan dibawah ini:

Nama	: Dwiki Aldani
NPM	: 18360028
Progran Studi	: Teknik Informatika
Fakultas	: Sains dan Teknologi Informasi
Jenis Karya	: Skripsi

Demi pengembangan ilmu pengetahuan, dengan ini menyetujui untuk memberikan kepada Institut Sains dan Teknologi Nasional **Hak Bebas Royalti Noneksklusif *(Non-exclusive Royalty-Free Right)*** atas karya ilmiah saya yang berjudul:

"**Pengembangan Program *Virtual Assistant* Dengan Menggunakan Python**"

Beserta perangkat yang ada (jika diperlukan). Dengan Hak Bebas Royalti Noneksekutif ini Institut Sains dan Teknologi Nasional berhak menyimpan, mengalihmedia/format-kan, mengelola dalam bentuk pangkalan data *(database) soft copy* dan *hard copy*, merawat, dan mempublikasikan tugas akhir saya selama tetap mencantumkan nama saya sebagai penulis/pencipta dan sebagai pemilik Hak Cipta.

Demikian pernyataan ini saya buat dengan sebenarnya.

Dibuat di : Jakarta

Pada tanggal : 02 Maret 2022

Yang menyatakan

Dwiki Aldani

ABSTRAK

Nama : Dwiki Aldani

NPM : 18360028

Judul : Pengembangan Program *Virtual Assistant* Dengan Menggunakan Python

Teknologi telah memberikan cara pandang baru terhadap kehidupan dan telah mempengaruhi berbagai aspek kehidupan. Tidak dapat dipungkiri bahwa teknologi memegang peranan yang sangat penting dalam segala aspek kehidupan. Beberapa aspek manual dapat diotomatisasi dan difasilitasi berkat teknologi. Salah satu dari teknologi yang membantu pengguna dalam memudahkan pekerjaan sehari-hari adalah *virtual assistant*. *Virtual assistant* merupakan serangkaian bahasa pemrograman berbasis *Natural Language Processing* (NLP). *Virtual assistant* adalah serangkaian program yang dapat mengerti perintah yang di berikan oleh pengguna hanya dengan menggunakan suara. Penelitian ini bertujuan mengembangkan program *virtual assistant* menggunakan bahasa pemrograman python bertujuan untuk menambahkan *User Interface* yang menjadi tampilan utama untuk program *virtual assistant* dan menambahkan fitur *face recognition* menggunakan modul opencv. Penelitian ini menggunakan metode pengembangan perangkat lunak *spiral*. Hasil dari penelitian ini menyatakan bahwa program *virtual assistant* dapat dibuatkan *User Interface* untuk memudahkan *virtual assistant* berinteraksi dengan pengguna dan menerapkan *face recognition* untuk mengenali pengguna.

Kata kunci: *virtual assistant*, python, *spiral*, *face recognition*

ABSTRACT

Name : Dwiki Aldani
NPM : 18360028
Title : *Virtual Assistant Program Development Using Python*

Technology has given a new perspective on life and has influenced various aspects of life. It is undeniable that technology plays a very important role in all aspects of life. Some manual aspects can be automated and facilitated thanks to technology. One of the technologies that help users in facilitating their daily work is a virtual assistant. Virtual assistant is a series of programming languages based on Natural Language Processing (NLP). Virtual assistant is a series of programs that can understand the commands given by the user only by using the voice. This study aims to develop a virtual assistant program using the python programming language aimed at adding a User Interface which is the main display for the virtual assistant program and adding face recognition features using the opencv module. This research uses spiral software development method. The results of this study indicate that the virtual assistant program can be made a User Interface to make it easier for the virtual assistant to interact with users and apply face recognition to recognize users.

Keywords: *virtual assistant, python, spiral, face recognition*

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB 1
PENDAHULUAN

1.1. Latar Belakang

Teknologi telah memberikan cara pandang baru terhadap kehidupan dan telah mempengaruhi berbagai aspek kehidupan. Tidak dapat dipungkiri bahwa teknologi memegang peranan yang sangat penting dalam segala aspek kehidupan. Beberapa aspek manual dapat diotomatisasi dan difasilitasi berkat teknologi. Selain itu, banyak proses kompleks dapat dibuat lebih sederhana dan lebih efisien dengan bantuan teknologi modern.

Salah satu dari teknologi yang membantu pengguna dalam memudahkan pekerjaan sehari-hari adalah *virtual assistant*. *Virtual assistant* merupakan serangkaian bahasa pemrograman berbasis *Natural Language Processing* (NLP) yang memungkinkan penguna mendapatkan balasan dari aplikasi sesuai dengan apa yang diinginkan oleh pengguna. (Perdana & Irwansyah, 2019).

Salah satu bahasa pemrograman yang dapat digunakan untuk membangun *virtual assistant* adalah python. Python menyediakan berbagai macam modul yang dapat digunakan untuk membangun program seperti wikipedia dan wolframalpha sebagai rujukan pencarian, pysimplegui sebagai modul yang bertugas menampilkan *User Interface* (UI) yang berinteraksi langsung dengan pengguna.

Pada penelitian yang dilakukan oleh (V. et al., 2021) yang berjudul "*The Voice Enabled Personal Assistant for Pc using Python*" program merupakan *personal assistant* yang dibuat untuk *desktop* menggunakan bahasa pemrograman python, hasil dari penelitian ini adalah program yang dapat berfungsi dengan perintah suara dan teks sebagai perintah untuk program misalnya mengenalkan diri pada pengguna, mengambil *screenshot*, dan mengerjakan perhitungan.

Pada penelitian yang dilakukan oleh (Sayyed et al., 2021) yang berjudul "*Desktop Assistant AI Using Python*" adalah program *virtual assistant* yang dapat diaktifkan hanya dengan suara dengan mengatakan "JARVIS" diikuti perintah, hasil dari penelitian ini adalah program *virtual assistant* yang dapat

membantu kegiatan pengguna misalnya pencarian dalam web, memutar musik, dan mecari pada wikipedia.

Virtual assistant yang sudah ada dapat dikembangkan dengan menambahkan *User Interface* pada program agar memudahkan pengguna untuk berinteraksi dengan program, dan menambahkan fitur *face recognition* pada program agar program dapat mengenali pengguna melalui kamera.

1.2. Rumusan Masalah

Berdasarkan uraian latar belakang masalah di atas dapat dibuat rumusan masalah berikut:

1) Bagaimana merancang *User Interface* (UI) *virtual assistant* menggunakan CorelDRAW ?
2) Bagaimana membuat UI menggunakan pysimplegui pada program *virtual assistant* ?
3) Bagaimana cara menambahkan fitur *face recognition* pada program *virtual assistant* ?

1.3. Tujuan Penelitian

Tujuan yang ingin dicapai dalam penelitian ini untuk mengembangkan program *virtual assistant*, yaitu dengan menambahkan :

1) *User Interface* (UI) untuk mempermudah pengguna dalam berinteraksi dengan program,
2) Fitur baru yaitu *face recognition* untuk menambahkan pengalaman dalam berinteraksi dengan program *virtual assistant*.

1.4. Manfaat Penelitian

Manfaat yang didapat antara lain:

1) Membantu pengembangan program *virtual assistant* yang sudah ada.
2) Membantu pengguna dengan memberikan pengalaman yang lebih baik terkait dengan penambahan fitur *face recognition*.

1.5. Batasan Masalah

Dalam pembuatan skripsi ini terdapat beberapa hal yang menjadi batasan masalah. Adapun batasan masalah tersebut adalah sebagai berikut:

1. Bahasa pemrograman yang digunakan adalah python versi 3.9.0 x86-x64 yang merupakan *open source software*.
2. *Software* yang digunakan untuk merancang *User Interface* adalah CorelDRAW X8.
3. Program yang dihasilkan ditampilkan dalam bahasa Inggris.

BAB 2
TINJAUAN PUSTAKA

2.1. Penelitian Terdahulu

Penelitian terdahulu bertujuan untuk mendapatkan bahan perbandingan dan acuan skripsi. Selain itu, untuk menghindari anggapan kesamaan dengan penelitian ini. Adapun hasil dari penelitian terdahulu dapat di lihat pada tabel 2.1.

Tabel 2. 1 Penelitian terdahulu

No	Nama	Hasil penelitian	Kesimpulan
1	(V. et al., 2021)	Program ini dibuat dengan bahasa pemrograman python dan dengan visual studio code, dan menggunakan modul – modul seperti *speech recognition* untuk mengubah perintah suara menjadi teks agar dapat di olah oleh sistem seperti yang digunakan oleh Bing. Pyttsx3 sebagai modul untuk mengubah text menjadi suara.	Program ini merupakan gambaran dari perkembangan *personal assistant* yang dapat diberikan perintah suara. *Personal assistant* seperti ini pada saat ini dapat menghemat waktu dan dapat mengerjakan sesuatu dengan efisien.
2	(Sayyed et al., 2021)	Program ini dibuat untuk membuat segalanya yang ada pada *desktop* menjadi lebih efisien, program ini dapat diaktifkan dengan mengatakan "JARVIS" dan diikuti perintah dari pengguna.	Program *virtual assistant* ini dikembangkan menggunakan python. *Virtual assistant* ini bekerja online dan dapat melakukan tugas dasar seperti memutar musik, memberikan laporan cuaca, membuka aplikasi, mencari pada wikipedia, mengirim *email* dan masih banyak lagi.

Sumber: hasil penelitian 2022

2.2. Pengembangan

Pengembangan didefinisikan sebagai aplikasi sistematis dalam pengetahuan atau pemahaman, diarahkan pada produksi bahan yang bermanfaat, perangkat,

dan sistem atau metode termasuk desain, pengembangan dan peningkatan prioritas serta proses baru untuk memenuhi persyaratan tertentu (Iii, 2019). Pengembangan adalah suatu usaha untuk meningkatkan mutu dan mendapatkan hasil yang lebih maksimal (Baidi, n.d., 2017).

Secara Etimologi pengembangan berasal dari padanan kata pengembang yang memiliki makna suatu proses, cara pembuatan atau sebuah proses kegiatan bersama yang dilakukan oleh penghuni suatu daerah untuk memenuhi kebutuhannya.

Secara Terminologi pengembangan adalah suatu proses yang mengupayakan peningkatan kemampuan dan keterampilan Sumber Daya Manusia (SDM) guna menghadapi perubahan lingkungan internal maupun eksternal melalui pendidikan keterampilan. Maka dapat diambil kesimpulan bahwa pengembangan adalah sebuah proses yang dilakukan untuk mendapatkan hasil yang bagus dengan melalui beberapa latihan dan pendidikan yang bagus (Henri, 2018).

2.3. Program

Program dapat didefinisikan sebagai sekelompok proyek yang dihubungkan bersama dalam manajemen yang terstruktur. Program dicirikan oleh ukurannya yang besar, kompleksitas yang tinggi**,** durasi pembuatan yang lama, dan biaya yang tinggi (Cha et al., 2018)

Program adalah kumpulan - kumpulan intruksi dalam bentuk bahasa, kode skema, maupum bentuk lain, dimana apabila dijadikan satu dengan media yang bisa di baca oleh komputer akan mampu membuat komputer bekerja untuk melakukan fungsi khusus termasuk persiapan dalam merancang intruksi tersebut (Faiz Zamzami & Nabella Duta Nusa, 2018)

2.4. *Virtual Assistant*

Virtual assistant atau asisten virtual adalah sebuah program yang memiliki banyak layanan yang dapat mengerti bahasa dan seluruh komposisi perintah pengguna agar dapat melakukan tugas yang diberikan (Campagna et al., 2019).

Virtual assistant adalah merupakan serangkaian bahasa pemrograman berbasis *Natural Language Processing* (NLP) yang memungkinkan penguna mendapatkan balasan dari aplikasi sesuai dengan apa yang diinginkan oleh pengguna (Perdana & Irwansyah, 2019).

Intellegent Virtual Assistant (IVA) merupakan salah satu layanan yang populer di kalangan pengguna yang dapat diaktifkan hanya dengan perintah suara. Untuk performa dan manajemen data yang maksimal, IVA yang terkenal seperti Amazon Alexa dan Google Assistant beroperasi menggunakan *system cloud computing architecture*. Pada proses ini, jumlah data yang sangat besar yang berisi kebiasaan dan jejak yang mencakup data dengan suara pengguna dapat di simpan dengan *server remote* yang berada dalam lingkungan IVA (Chung & Lee, 2018).

2.5. *Natural Language Processing* (NLP)

NLP adalah disiplin ilmu komputer yang bertujuan untuk memahami maksud dari bahasa manusia dan konsep. Sementara manusia cukup mahir memahami tata bahasa dan sintaks *linguistic* serta hubungan spasial tersirat, komputer memiliki kesulitan besar pengolahan *query* bahasa alami. Contoh sistem ini dan penggunaan *Application Programming Interface* (API) adalah *Natural Language Toolkit* (NLTK) dan *Core* NLP dari Stanford University (Huda, 2021).

Natural Language Processing atau pengolahan bahasa alami, biasanya di singkat dengan NLP, merupakan bidang kecerdasan buatan dimana komputer didesain untuk dapat berkomunikasi dengan manusia menggunakan bahasa alami, seperti bahasa Indonesia. NLP tidak bertujuan untuk mentransformasikan bahasa yang diterima dalam bentuk teks atau suara menjadi data digital dan/atau sebaliknya, melainkan bertujuan untuk memahami arti dari kalimat yang diberikan dalam bahasa alami dan memberikan respon yang sesuai, misalnya dengan melakukan suatu aksi tertentu atau menampilkan data tertentu (Husamuddin et al., 2020).

2.6. *Speech Recognition*

Speech recognition atau *speech to text* adalah modul yang banyak digunakan untuk memudahkan jika seseorang tidak ingin meng-*input* mengunakan *keyboard* dalam aplikasi, cukup hanya dengan mengunakan perintah suara untuk menjalankan fungsi dalam aplikasi tersebut (, et al., 2019).

Speech recognition adalah metode atau fungsi untuk merekondisi atau mengubah bunyi ucapan yang berisi bahasa tertentu kedalam bentuk teksnya tanpa harus mengetikannya dengan *keyboard*. Masukan berupa suara mampu diubah menjadi teks yang mampu dibaca oleh sistem (, et al., 2019).

2.7. *Search Engine*

Mesin Pencari atau *Search Engine* adalah sebuah sistem *software* yang di desain untuk mencari berbagai informasi yang tersimpan dalam layanan *World Wide Web* (WWW), *File Transfer Protocol* (FTP), *Mailing List*, atau *News Group* yang berada di dalam sebuah atau sejumlah Server dalam suatu batasan jaringan (Nurjaya & Riyanto, 2018).

Search engine atau mesin pencari disediakan untuk membantu para pengguna internet menemukan suatu topik atau informasi melalui beberapa situs yang berkaitan dengan topik dan menggunakan fasilitas pada mesin pencari. *search engine* merupakan mesin pencari informasi yang telah menjadi alat penting bagi pencari informasi, meskipun mereka menyadari keterbatasan dalam ketrampilan menggunakan *search engine* (Saputri, 2021).

2.8. Python

Python adalah bahasa pemrograman interpretatif multiguna dengan filosofi perancangan yang berfokus pada tingkat keterbacaan kode. Python diklaim sebagai bahasa yang menggabungkan kapabilitas, kemampuan, dengan sintaksis kode yang sangat jelas, dan dilengkapi dengan fungsionalitas pustaka standar yang besar serta komprehensif.

Python mendukung multi paradigma pemrograman, namun tidak dibatasi pada pemrograman berorientasi objek, pemrograman imperatif, dan pemrograman fungsional. Salah satu fitur yang tersedia pada python adalah

sebagai bahasa pemrograman dinamis yang dilengkapi dengan manajemen memori otomatis (Syahrudin & Kurniawan, 2018).

Python telah menjadi bahasa pemrograman yang dipilih sebagai pilihan untuk penelitian dan projek industri yang berkaitan dengan *data science, machine learning*, dan *deep learning*. Karena optimasi adalah hal yang tidak dapat di pisahkan dari bidang penelitian, banyak optimasi *framework* yang terkait muncul hanya dalam beberapa tahun kebelakang (Blank & Deb, 2020).

2.9. *Face recognition*

Face recognition atau pengenalan wajah adalah proses mengidentifikasi, mengenali serta membandingkan wajah yang sudah tersimpan dalam *dataset* dengan memanfaatkan kecerdasan buatan. Pengenalan wajah dapat diartikan sebagai pengelompokan wajah "dikenali" dan wajah "tidak dikenali" (B. T. Utomo et al., 2020).

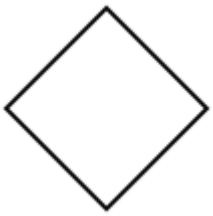
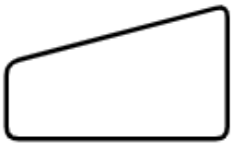
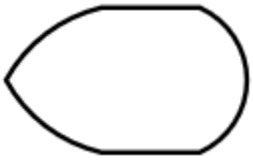
Teknologi pengenalan wajah adalah sebuah teknologi *biometric*, dimana teknologinya berbasis dari identifikasi dari fitur wajah atau seseorang. Orang – orang mengumpulkan gambar wajah, dan teknologi pengenalan wajah akan langsung memprosesnya (Li et al., 2020).

2.10. *Flowchart*

Flowchart merupakan penggambaran secara grafik dari langkah-langkah dan urutan prosedur suatu program. Biasanya mempengaruhi penyelesaian masalah yang khusunya perlu dipelajari dan dievaluasi lebih lanjut. Bagan alir program (*program flowchart*) merupakan bagan yang menjelaskan secara rinci langkah – langkah dari proses program. Bagan alir program dibuat dari derivikasi bagan alir sistem (Soufitri, 2019) .

Flowchart menggunakan simbol – simbol untuk menggambarkan alur suatu program. Adapun simbol – simbol yang ada pada *flowchart* dapat dilihat pada tabel 2.2.

Tabel 2. 2 *Flowchart symbol*

No	Simbol	Nama	Keterangan
1		Arus/*Flow*	Untuk menyatakan jalannya arus suatu proses
2		*Terminal Points*	Menandakan awal / akhir *flowchart*
3		Proses	Sebuah fungsi pemrosesan yang dilaksanakan oleh computer biasanya menghasilkan perubahan terhadap data atau informasi
4		*Decision* / Logika	Untuk menunjukkan suatu kondisi tertentu, dengan dua kemungkinan, YA/TIDAK
5		*Manual input*	*Input* yang dimasukkan secara manual dari *keyboard*
6		*Display*	Berfungsi untuk menyatakan peralatan *output* yang digunakan yaitu layar, *plotter*, *printer*, dan sebagainya
7		*Alternate process*	Simbol ini di gunakan untuk menandakan alir yang memiliki alur alternatif dari alur normal.

Sumber: hasil penelitian 2022

2.11. *Blackbox Testing*

Blackbox testing merupakan pengujian kualitas perangkat lunak yang berfokus pada fungsionalitas perangkat lunak. Pengujian *blackbox* bertujuan untuk menemukan fungsi yang tidak benar, kesalahan antarmuka, kesalahan

pada struktur data, kesalahan performansi, kesalahan inisialisasi dan terminasi (Wijaya & Astuti, 2021).

Blackbox testing merupakan Teknik pengujian perangkat lunak yang berfokus pada spesifikasi fungsional dari perangkat lunak. *Blackbox testing* bekerja dengan mengabaikan struktur kontrol sehingga perhatiannya difokuskan pada informasi domain. *Blackbox testing* memungkinkan pengembang *software* untuk membuat himpunan kondisi input yang akan melatih seluruh syarat - syarat fungsional suatu program (Snadhika Jaya, 2018).

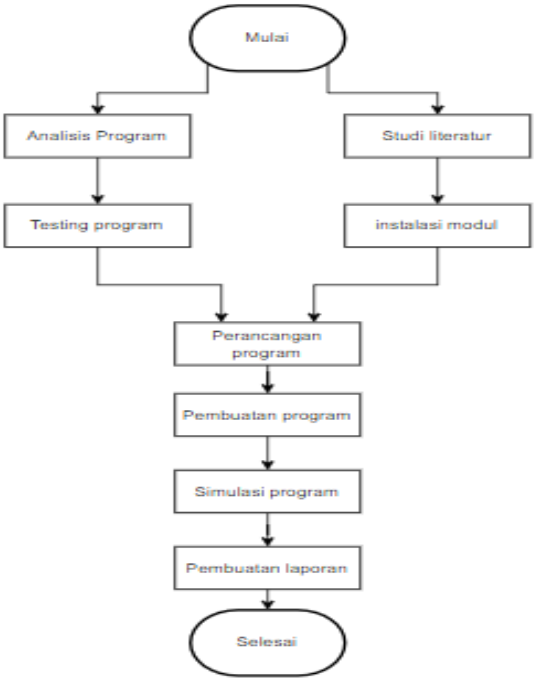
BAB 3
METODOLOGI PENELITIAN

3.1. Metodologi Pengumpulan Data

Pengumpulan data pada penelitian ini dilakukan dengan cara studi literatur, Studi literatur adalah serangkaian kegiatan yang berkenaan dengan metode pengumpulan data pustaka, membaca dan mencatat, serta mengelolah bahan penelitian. Studi Literatur adalah merupakan penelitian yang dilakukan oleh peneliti dengan mengumpulkan sejumlah buku buku, majalah yang berkaitan dengan masalah dan tujuan penelitian (Judithia, 2019).

3.2. Kerangka Penelitian

Kerangka penelitian ini merupakan gambaran tahapan penelitian program *virtual assistant*. Adapun tahapan kerangka penelitian yang digunakan dapat di lihat pada Gambar 3.1.

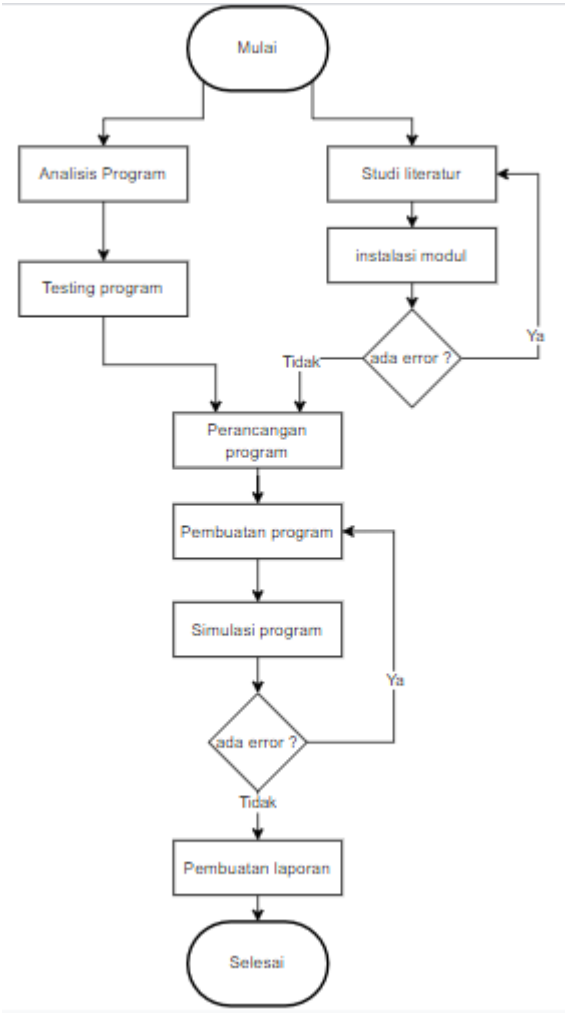


Gambar 3. 1 Kerangka penelitian pengembangan program *virtual assistant*
Sumber: hasil penelitian 2022

Pada Gambar 3.1 Menjelaskan kerangka penelitian dari awal pengembangan *virtual assistant* sampai selesai, penjelasan kerangka penelitian dapat di lihat pada penjelasan gambar 3.2.

3.3. Alur Penelitian

Alur penelitian ini menjelaskan bagaimana penelitian dilakukan untuk mencari hasil dari analisis dinamis. Dapat dilihat pada gambar 3.2



Gambar 3. 2 Alur penelitian pengembangan *virtual assistant*

Sumber: hasil penelitian 2022

Pada gambar 3.2 Menggambarkan tentang alur penelitian pengembangan *virtual assistant*. Adapun penjelasan dari tahapan – tahapan yang ada pada alur penelitian sebagai berikut:

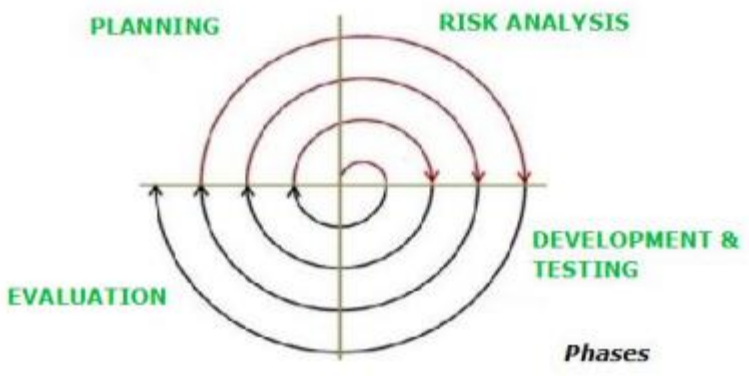
- Analisis program *virtual assistant* yang nantinya akan dikembangkan, dan tentukan apa saja yang akan ditambahkan dalam program.
- Pada saat yang sama dengan analisis program dilakukan juga studi literatur tentang hal yang berkaitan dengan pengembangan aplikasi yang akan di

lakukan, seperti apa spesifikasi *hardware*, dan modul – modul apa saja yang perlu di lengkapi yaitu wolframalpha, Wikipedia, pyttsx3, pyaudio, dan *speech recognition, face recognition* dan yang lainnya.

- Setelah dilakukan analisis program lakukan *testing* pada program sebelumnya untuk mengetahui apa yang akan menjadi model rancangan untuk pengembangan berikutnya.
- Lakukan instalasi modul yang dibutuhkan, jika terdapat *error* pada saat instalasi modul – modul, kembali ke tahap studi literatur dan identifikasi, apa yang menjadi masalah saat instalasi.
- Jika tidak terdapat *error* pada saat instalasi lakukan perancangan program, agar tau program seperti apa yang akan dibuat nantinya, seperti merancang tampilan utama dari program dan bagaimana menambahkan fitur *face recognition*.
- Lalu lakukan pembuatan program dengan membuat tampilan dan menambahkan fitur program seperti pada rancangan.
- Selanjutnya lakukan simulasi program dan perhatikan apakah hasilnya sudah sesuai dengan rancangan atau terdapat *error* di dalam program.
- Jika terjadi *error* kembali ke tahap pembuatan program dan lihat apa yang menjadi *error* pada program.
- Jika tidak ada *error* maka lanjutkan dengan penulisan laporan.

3.4. Metode Pengembangan Perangkat Lunak

Metodologi yang digunakan pada penelitian ini adalah metode *spiral*. Model spiral (*spiral model*) adalah model proses *software* yang evolusioner yang merangkai sifat iteratif dari prototipe dengan cara kontrol dan aspek sistematis dari modelsekuensial linier. Model ini berpotensi untuk pengembangan versi pertambahan *software* secara cepat. Di dalam model spiral, *software* dikembangkan di dalam suatu deretan pertambahan. Selama awal iterasi, rilis inkremental bisa merupakan sebuah model atau prototipe kertas. Selama iterasi berikutnya, sedikit demi sedikit dihasilkan versi sistem rekayasa yang lebih lengkap (A. N. Utomo & Alfaridzi, 2018), alur metode penelitian *spiral* dapat di lihat pada gambar 3.3.



Gambar 3. 3 Model metode *spiral* yang digunakan *(A. N. Utomo & Alfaridzi, 2018)*

Pada Gambar 3.3 menjelaskan proses penelitian pada model metode *spiral* yang digunakan dalam penelitian ini. Adapun penjelasan proses yang terjadi dalam model sebagai berikut :

- *Planning* (Perencanaan)

  Pada fase ini hal yang dilakukan adalah:

  - Melakukan *testing* pada program *virtual assistant* lalu mencatat apa saja yang akan dikembangkan dari program *virtual assistant*.
  - Membuat rancangan program yang nantinya akan di bangun untuk mengembangkan program *virtual assistant*.
- *Risk Analysis* (Analisis Resiko)

  Pada fase ini hal yang dilakukan adalah:

  - Menganalisa dampak yang terjadi saat program *virtual assistant* dikembangkan.
  - Membuat prototipe dari program *virtual assistant* untuk di coba dan di lihat kekurangan dari prototipe.
- *Development and Testing* (Pengembangan dan Testing)

  Pada fase ini hal yang dilakukan adalah:

  - Pengembangan dari program prototipe dan menambahkan kekurangan yang telah di catat pada tahap sebelumnya.

- *Testing* program untuk mengetahui apakah sudah sesuai dengan keinginan.

- *Evaluation* (Evaluasi)

  Pada fase ini hal yang dilakukan adalah:

  - melakukan evaluasi program untuk melihat apakah program *virtual assistant* ini sudah cukup atau terdapat tambahan untuk program.

3.5. Alat Penelitian

Pada penelitian ini digunakan alat penelitian yang berupa perangkat keras dan perangkat lunak sebagai berikut:

1. Perangkat keras yang digunakan :

   a. Laptop dengan Spesifikasi:

   - Prosesor Intel Core i5 -8250U (3.4 GHz)
   - NVIDIA Geforce 930MX
   - Memori RAM 4 GB
   - 1TB HDD

   b. *Mouse* Alcatroz ASIC5

   c. *Keyboard standard*

2. Perangkat Lunak yang digunakan:

   a. Operating System: Windows 10 64-bit

   b. Python Versi 3.0+

   Berfungsi sebagai bahasa pemrograman pada *virtual assistant*.

   c. Command prompt

   Berfungsi untuk menginstall modul dan tempat menjalankan program python.

   d. Notepad++

   Berfungsi untuk meng-*edit* kode program.

   e. Notepad

   Berfungsi untuk menaruh *note – note* kode program.

   f. Microsoft Word

   Berfungsi untuk menulis laporan penelitian.

g. Snipping Tool

Berfungsi mengambil tangkapan layar.

h. Google Chrome

Berfungsi untuk mencari jurnal pada internet

i . Visual Studio 2019 (desktop development C++ package installation)

Berfungsi sebagai komponen agar dapat meng-*install* modul pyaudio, komponen yang di maksud adalah Microsoft Visual C++ 14.0+.

j. CorelDRAW X8

Berfungsi sebagai alat untuk membuat rancangan *User Interface*

3.6. Perancangan Program

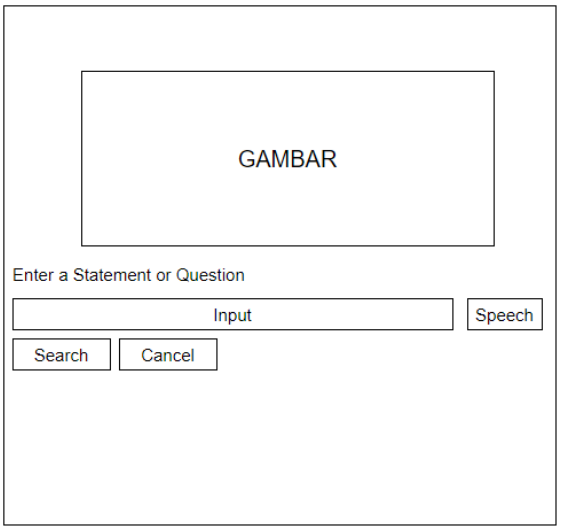
Pada tahap ini dilakukan perancangan *User Interface* program *virtual assistant* dan pembuatan *flowchart*.

3.6.1. Perancangan *User Interface* (UI)

Pada tahap ini dilakukan pembuatan rancagan UI dengan menggunakan CorelDRAW X8.

- Tampilan utama *virtual assistant* python:

  Tampilan program *virtual assistant* python dapat dilihat pada gambar 3.4.

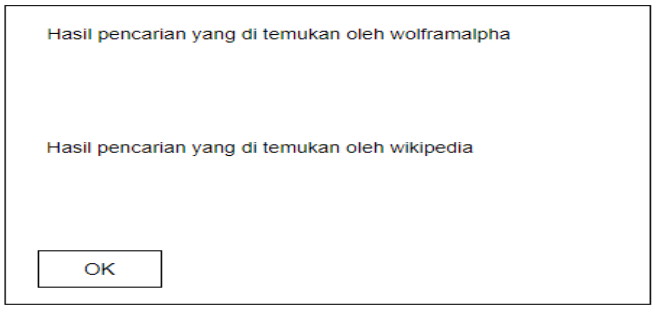


Gambar 3.4 Tampilan utama dari *virtual assistant* python

Sumber: hasil rancangan 2021

Pada Gambar 3.4 Menjelaskan perancangan tampilan utama dari *virtual assistant*, menambahkan gambar pada bagian atas *textbox* agar lebih menarik untuk pengguna, menambahkan tombol *Search* untuk metode pencarian dengan teks, dan menambahkan tombol *Speech* yang nantinya akan berguna sebagai tombol untuk *speech recognition*.

- *Popup messages* hasil pencarian *virtual assistant*



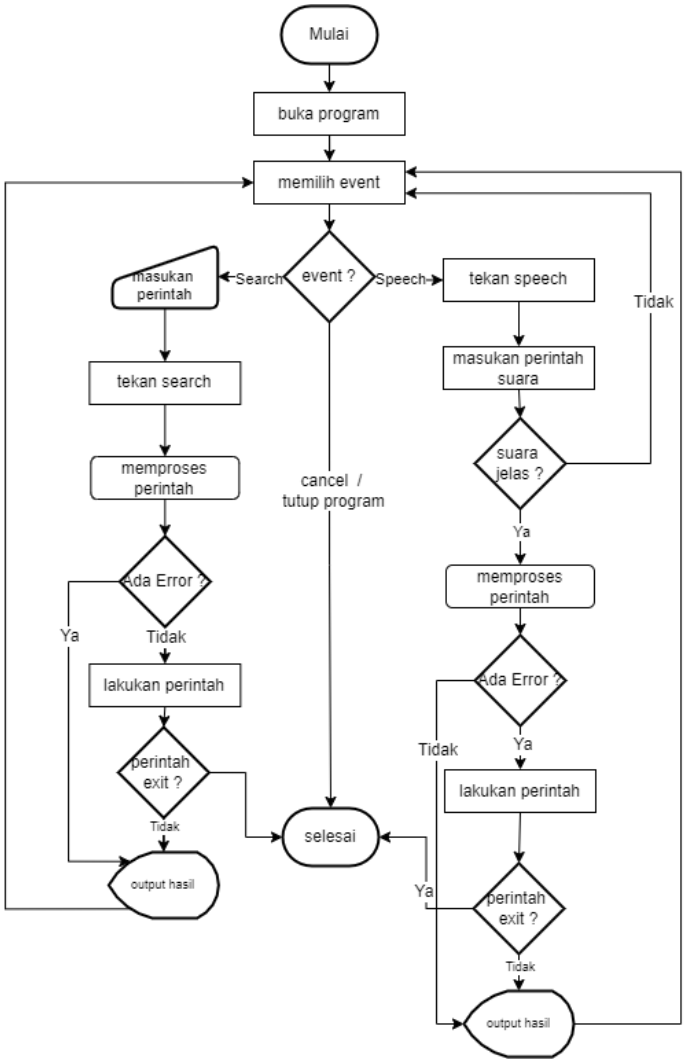
Gambar 3.5 Popup message hasil pencarian *virtual assistant*

Sumber: hasil rancangan 2021

Pada Gambar 3.5 Menjelaskan *popup message* ini akan muncul ketika pengguna menekan tombol *Search* atau *Speech*.

3.6.2 *Flowchart*

Penggambaran *flowchart* program dapat dilihat pada gambar 3.6.



Gambar 3. 6 *Flowchart* utama program *virtual assistant*

Sumber: hasil rancangan tahun 2022

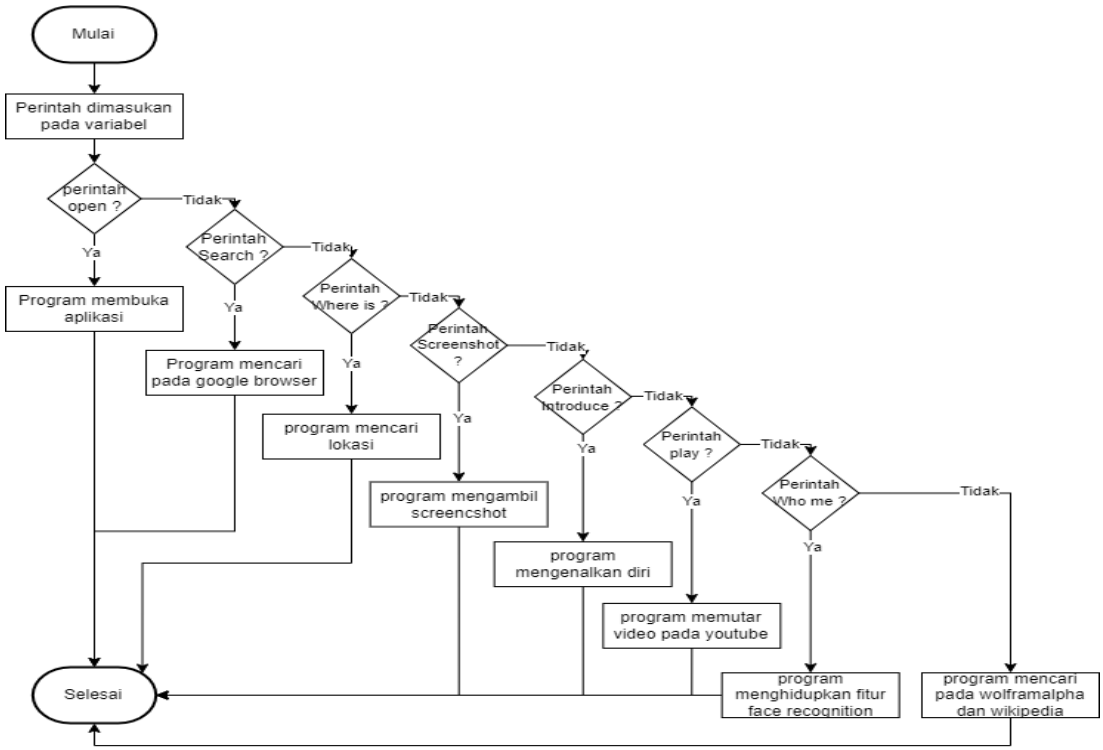
Gambar 3.6 Menjelaskan *flowchart* program *virtual assistant*, adapun proses – proses yang ada pada *flowchart* dijelaskan sebagai berikut:

- Buka program pada *command prompt* (cmd), ubah direktori ke tempat *file* berada, lalu ketikkan **python nama_file.**
- Selanjutnya program akan memproses pysimplegui untuk memunculkan *User Interface* (UI). Pada UI terdapat tombol – tombol yang berisi *event* – *event* yang dapat dipilih untuk memberikan perintah.

- Jika *event* yang ingin dipilih oleh pengguna adalah *search*, maka pengguna harus memasukkan kata kunci ke dalam *textbox* yang tersedia lalu menekan *button search*.
- Selanjutnya perintah akan di proses oleh program.
- Selanjutnya program akan melakukan perintah yang telah di proses.
- Jika di dalam perintah terdapat kata ***exit*** maka program akan langsung selesai.
- Jika tidak, maka program akan memberikan hasil pada layar berupa tindakan seperti membuka aplikasi, mencari video di youtube, mencari pada google chrome, memberitahu suhu pada lingkungan sekitar, membuka *face recognition* untuk mengenali pengguna, dan membuka lokasi dari tempat yang diinginkan oleh pengguna atau *output* hasil pencarian berupa *popup message* jika pengguna hanya menginput kata saja tanpa perintah yang jelas.
- Lalu program akan kembali menunggu *event* dari pengguna untuk perintah selanjutnya.
- Jika *event* adalah *speech*, maka program akan mendengarkan perintah dari pengguna.
- Perintah kemudian diolah dan menjadi teks jika suara jelas, jika suara tidak jelas, maka program akan memunculkan notifikasi *error*.
- Jika suara jelas maka diolah menjadi teks.
- Selanjutnya perintah akan di proses oleh program.
- Selanjutnya perintah akan dijalankan sesuai dengan keinginan pengguna seperti mencari informasi tentang suatu hal, mencari tahu lokasi sebuah tempat, mencari video pada youtube, membuka aplikasi yang diinginkan dan menghidupkan fitur *face recognition* untuk mengenali pengguna.
- Jika pada perintah terdapat kata ***exit*** maka program akan langsung selesai.
- Jika *event* yang diberikan oleh pengguna adalah *cancel* maka program akan selesai.

3.6.2 *Alternate Process*

Proses alternatif ini merupakan penjelasan dari *symbol* proses alternatif yang berada pada gambar 3.6, adapun alur dari proses alternatif dapat dilihat pada gambar 3.7.



Gambar 3. 7 *Alternate process* dari *flowchart*

Sumber : rancangan tahun 2022

Gambar 3.7 Menjelaskan alur *alternatif* dari alur "memproses perintah", proses – proses pada alur alternatif di jelaskan sebagai berikut:

- Perintah dimasukkan kedalam variabel.
- Jika terdapat kata *open* pada variabel maka program akan membuka aplikasi yang ditentukan oleh pengguna.
- Jika tidak terdapat kata *open*, maka program akan melihat apakah pada perintah terdapat *search*, jika iya maka program akan membuka browser dan akan mencari kata kunci yang dimasukkan oleh pengguna,
- Jika tidak terdapat kata *search*, maka program akan melihat apakah terdapat kata *where is*, jika iya maka program akan membuka browser dan mencari lokasi dari tempat yang ditentukan oleh pengguna,

- Jika tidak terdapat kata *where is*, maka program akan melihat apakah terdapat kata *screenshot*, maka program akan mengambil *screenshot*,
- Jika tidak terdapat kata *screenshot*, maka program akan melihat apakah terdapat kata *introduce* pada program, jika iya maka program akan mengenalkan diri pada pengguna,
- Jika tidak terdapat kata *introduce*, maka program akan melihat apakah terdapat kata *play*, jika iya maka program akan membuka browser dan mencari video dengan kata kunci yang diberikan oleh pengguna,
- Jika tidak terdapat kata *play*, maka program akan melihat apakah terdapat kata *who* dan *me*, jika iya maka program akan menghidupkan fitur *face recognition*, dan akan mengenali pengguna.

BAB 4

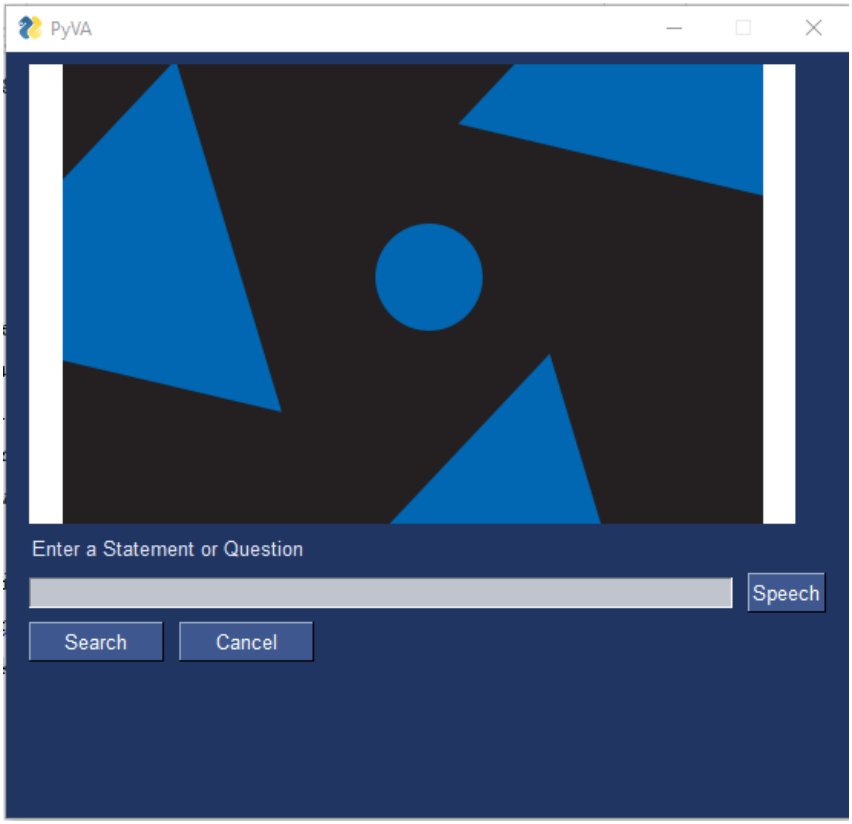
HASIL DAN PEMBAHASAN

4.1. Hasil

Pada tahap ini menunjukkan hasil dari rancangan program *virtual assistant*.

4.1.1. Tampilan Utama *Virtual Assistant*

Pada tahap ini merupakan hasil dari perancangan tampilan utama untuk program *virtual assistant*, hasil dapat dilihat pada gambar 4.1.



Gambar 4. 1 Jendela utama dari *virtual assistant*

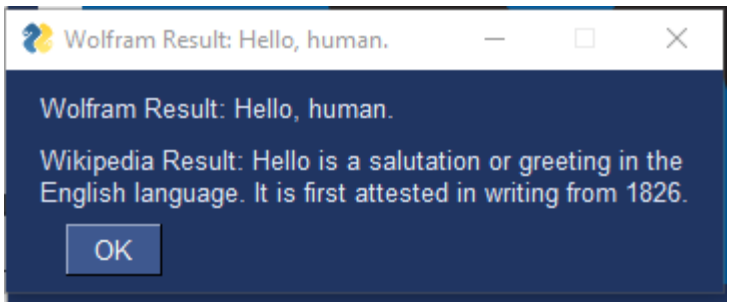
Sumber: hasil penelitian 2022

Pada gambar 4.1 merupakan hasil dari *output* rancangan UI yang dibuat menggunakan pysimplegui, gambar pada bagian atas jendela membuat tampilan menjadi lebih keren dalam penampilan, ukuran jendela yang cukup luas mempermudah pengguna untuk berinteraksi dengan program, tombol *search* berfgungsi untuk melakukan pencarian dengan

teks, dan tombol *speech* sebagai tombol aktivasi fungsi *speech recognition*.

4.1.2. *Output* Hasil Pencarian *Virtual Assistant*

Hasil *output* pencarian program *virtual assistant* menampilkan semua hasil dari wolframalpha dan wikipedia sumber, gambar dari hasil pencarian menambahkan tampilan untuk kedua sumber dari wolframalpha dan wikipedia. Hasil *output* dapat dilihat pada gambar 4.2

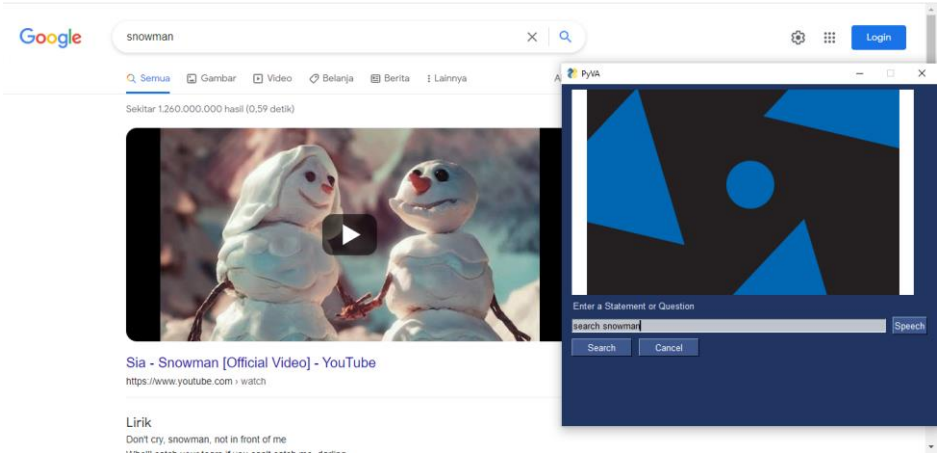


Gambar 4. 2 jendela hasil *ouput* dari pencarian program *virtual assistant*
Sumber: hasil penelitian 2022

Pada gambar 4.2 merupakan jendela hasil *output* yang keluar setelah hasil pencarian di temukan, setelah jendela ini keluar maka modul *speech recognition* akan membacakan hasil *output* untuk pengguna.

4.1.3. Pencarian Pada Google Chrome

Pada tahap ini menunjukkan hasil dari pencarian pada google melalui program *virtual assistant.* Hasil dapat dilihat pada gambar 4.3.



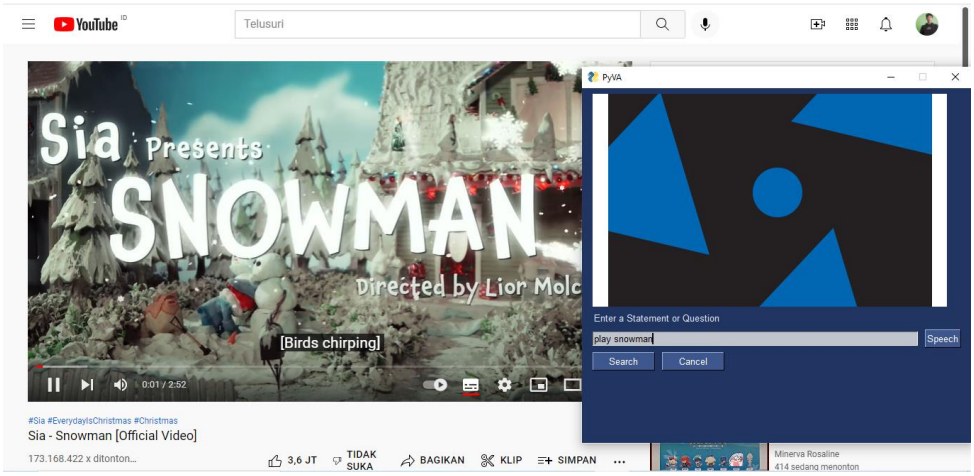
Gambar 4. 3 Pencarian pada google melalui *virtual assistant*

Sumber: hasil penelitian 2022

Pada gambar 4.3 Menjelaskan program *virtual assistant* yang membuka *browser default* yang ada pada *desktop* untuk melakukan pencarian pada google.

4.1.4. Pencarian Video Pada Youtube

Pada tahap ini menunjukkan hasil pencarian video pada youtube melalui program *virtual assistant*. Hasil dapat dilihat pada gambar 4.4.



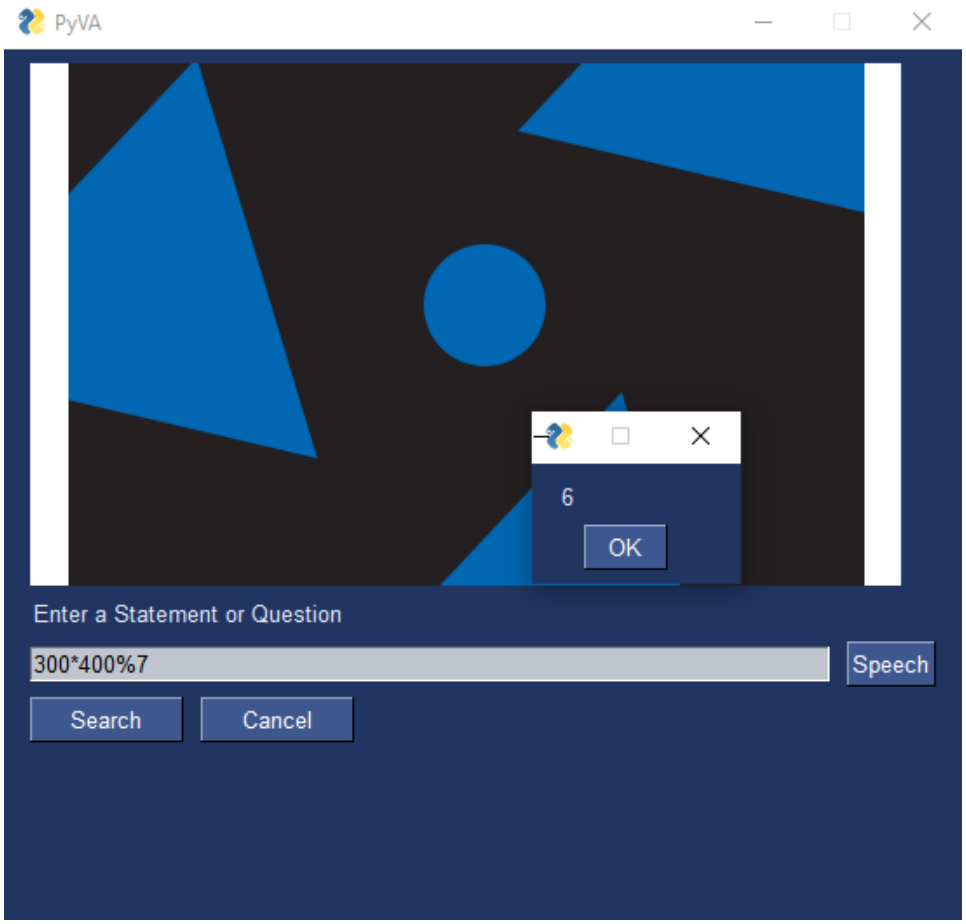
Gambar 4. 4 Hasil pencarian video pada youtube melalui *virtual assistant*

Sumber: hasil penelitian 2022

Gambar 4.4 Menjelaskan program *virtual assistant* yang membuka *browser default* pada *desktop* lalu membuka youtube dan mencari video yang diinginkan.

4.1.5. Pencarian Jawaban Dari Formula Matematika

Pada tahap ini menunjukkan hasil pencarian jawaban dari formula matematika yang diberikan melalui program *virtual assistant*. Hasil dapat dilihat pada gambar 4.5.



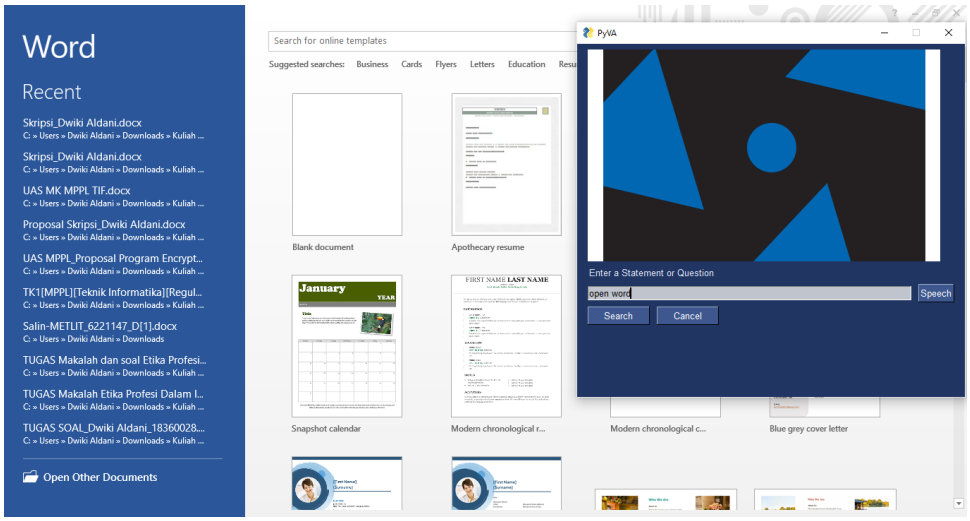
Gambar 4. 5 Hasil pencarian jawaban formula matematika melalui *virtual assistant*

Sumber: hasil penelitian 2022

Gambar 4.5 Menjelaskan hasil pencarian jawaban dari formula matematika yang diberikan, contoh diatas adalah 300 di kali 400 modulus 7 maka hasilnya adalah 6.

4.1.6. Membuka Aplikasi Microsoft Word

Pada tahap ini menunjukkan hasil program *virtual assistant* membuka aplikasi microsoft word. Hasil dapat dilihat pada gambar 4.6.



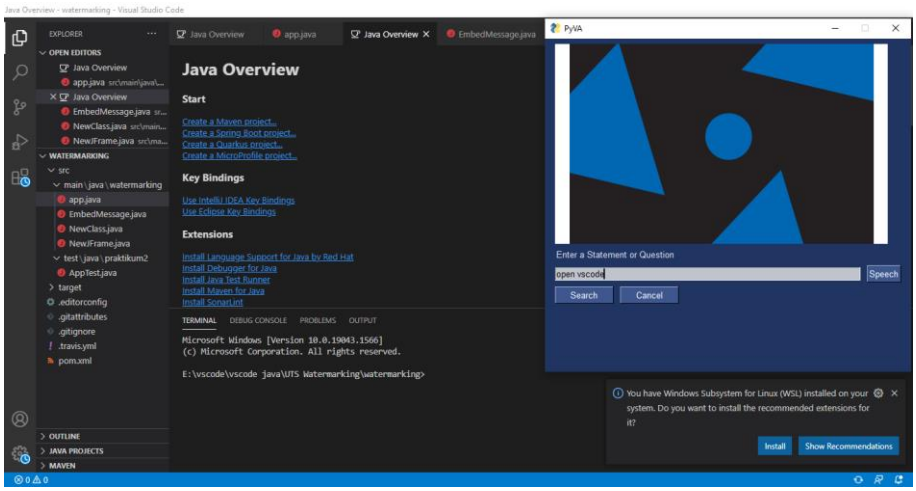
Gambar 4. 6 Program *virtual assistant* membuka microsoft word

Sumber: hasil penelitian 2022

Gambar 4.6 Menjelaskan program *virtual assistant* yang membuka aplikasi microsoft word.

4.1.7. Membuka Aplikasi Visual Studio Code

Pada tahap ini menunjukkan hasil program *virtual assistant* membuka aplikasi visual studio code. Hasil dapat dilihat pada gambar 4.7.



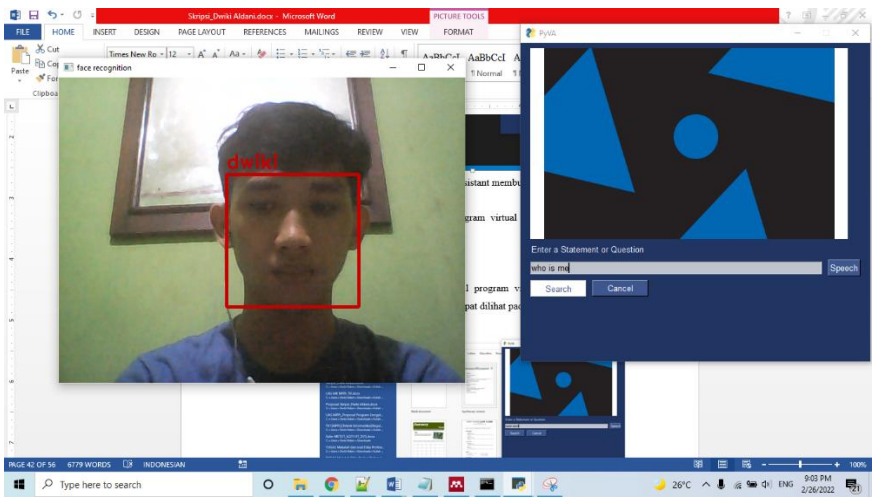
Gambar 4. 7 Program *virtual assistant* membuka visual studio code

Sumber: hasil penelitian 2022

Gambar 4.7 Menjelaskan program *virtual assistant* yang membuka aplikasi visual studio code.

4.1.8. Hasil *Face Recognition*

Pada tahap ini menunjukkan hasil *face recognition* program *virtual assistant*. Hasil dapat dilihat pada gambar 4.8.



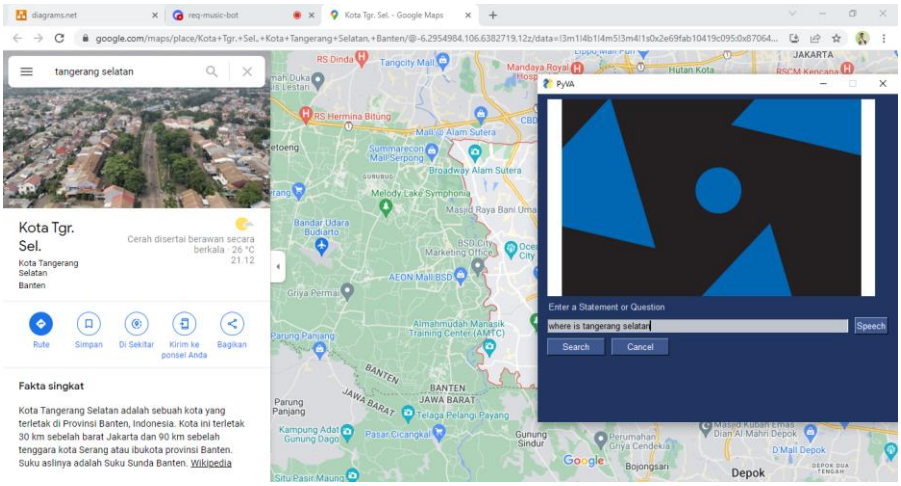
Gambar 4. 8 Fitur *face recognition* program *virtual asisstant*

Sumber: hasil penelitian 2022

Gambar 4.8 Menjelaskan program *virtual assistant* yang membuka kamera dan menggunakannya sebagai media *face recognition* yang dapat mengenali wajah yang ada dalam kamera.

4.1.9. Hasil Pencarian Lokasi

Pada tahap ini menunjukkan hasil pencarian lokasi suatu tempat pada program *virtual assistant*. Hasil dapat dilihat pada gambar 4.9.



Gambar 4. 9 Pencarian suhu sekitar pada *virtual assistant*

Sumber: hasil penelitian 2022

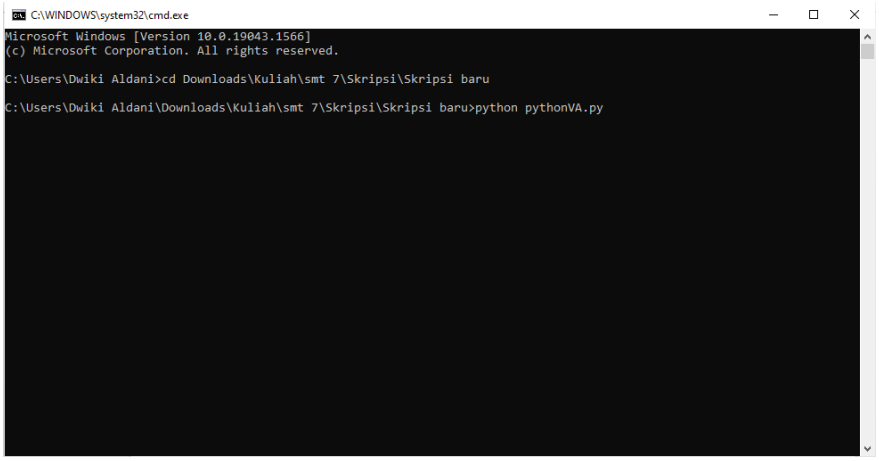
Gambar 4.9 Menjelaskan program *virtual assistant* yang membuka *browser* untuk menunjukan lokasi dari tempat yang ingin di cari oleh pengguna.

4.2. Pembahasan

Pada tahap ini membahas apa saja yang ada pada hasil.

4.2.1. Tampilan Utama Program

Pada tahap ini program di buka menggunakan command prompt, buka command prompt lalu ketikkan "**cd (direkori file utama)**" lalu tekan *enter*. Selanjutnya ketikkan "**python (nama file)**" lalu tekan *enter* dan program akan berjalan, program akan meng-import seluruh modul yang terdapat pada kode.



Gambar 4. 10 Membuka program menggunakan cmd

Sumber: hasil penelitian 2022

Gambar 4.10 Menjelaskan proses menjalankan program melalui command prompt. Selanjutnya program akan membaca *dataset* yang ada pada direktori "**Source code\image**" dan meng-*encode* gambar sebagai *dataset* yang akan digunakan dalam *face regontion*, untuk lebih lengkap dapat dilihat pada lampiran-01.b.

```
import cv2;
from simple_facerec import SimpleFacerecognition

sfrec = SimpleFacerecognition()
sfrec.load_encoding_images("source code/images/")
```

Selanjutnya program akan menjalankan pysimplegui untuk menampilkan jendela utama.

```
import PySimpleGUI as sg
sg.theme('DarkBlue13')
layout =[[sg.Image(r'"logoVA.png'",
size=(500,300))],[sg.Text('Enter a Statement or Question')],
[sg.InputText(size=(65,1)),
sg.Button('Speech')],[sg.Button('Search', size=(10,1)),
sg.Button('Cancel', size=(10,1))]]
window = sg.Window('PyVA', layout, size=(550,500))
```

4.2.2. Output Hasil pencarian

Pada tahap ini membahas *popup message* hasil pencarian pada program yang dilakukan oleh modul wolframalpha dan wikipedia.

```
import wikipedia
import wolframalpha
client = wolframalpha.Client("964PPA-8G9GGPL9YA")

rawres_wolfram = client.query(values[1]).results
        result2_wikipedia = wikipedia.summary(values[1],
auto_suggest=False, sentences=2)
        result1_wolframalpha = next(rawres_wolfram).text
        engine.say(result1_wolframalpha)
        sg.PopupNonBlocking("Wolfram Result:
"+result1_wolframalpha,"Wikipedia Result: "+result2_wikipedia)
```

Program akan meng-*import* modul wikipedia dan wolframalpha, wolframalpha membutuhkan *Application Programming Interface* (API) agar dapat berfungsi, API wolframalpha bisa di dapatkan dengan *sign up* ke situs resmi wolframalpha dan pilih menu API.

4.2.3. Pencarian Jawaban Dari Formula Matematika

Pada tahap ini membahas tentang program yang dapat mengerjakan persamaan matematika yang dimasukkan oleh pengguna, ini merupakan fitur dari modul wolframalpha, ketika data berupa persamaan matematika muncul, maka modul wolframalpha akan memberikan hasilnya kepada pengguna melalui *popup message*.

4.2.4. Pencarian Pada Google Chrome

Pada tahap ini membahas program yang dapat membuka chrome dan mencari kata kunci tanpa harus membuka google chrome secara manual, cukup dengan memasukkan kata kunci "*search*" pada pencarian teks maupun suara dan diikuti dengan kata kunci yang ingin di cari, maka program akan otomatis mencarikannya untuk pengguna.

```
import webbrowser
```

```
elif ("search" in texts.lower()):
                query = texts.lower().replace('search', ' ')
                url                                                    =
"https://www.google.com.tr/search?q={}".format(query)
                webbrowser.open_new_tab(url)
                engine.say('searching')
                engine.say(query)
```

Program akan meng-*import* modul *web browser* dan jika kata kunci menggunakan kata "*search*" diikuti dengan kata kunci maka program akan otomatis membuka browser dan mencarikan kata kunci tersebut pada google.

4.2.5. Pencarian Video Pada Youtube

Pada tahap ini membahas program yang dapat membuka chrome dan mencari kata kunci tanpa harus membuka browser secara manual, cukup dengan memasukkan kata kunci "*play*" pada pencarian teks maupun suara dan diikuti dengan kata kunci yang ingin di cari, maka program akan otomatis mencarikannya untuk pengguna.

```
import webbrowser

elif ("play" in texts.lower()):
                video = texts.lower().replace('play',' ')
                engine.say('playing')
                engine.say(video)
                pywhatkit.playonyt(video)
```

Program akan meng-import modul webbrowser dan jika kata kunci menggunakan kata "*play*" diikuti dengan kata kunci maka program akan otomatis membuka browser dan mencarikan kata kunci tersebut pada youtube dan memutarnya.

4.2.6. Membuka Aplikasi Microsoft Word

Pada tahap ini program membukakan otomatis aplikasi microsoft word cukup dengan perintah "*open*" diikuti dengan "*word*", maka

program akan membuka kan aplikasi tersebut jika aplikasinya memang ada pada *windows*.

```
elif ("word" in texts.lower()):
                    engine.say("opening");
                    engine.say("Microsoft Word")
                    engine.runAndWait()
                    os.startfile("WINWORD")
                elif ("vscode" in texts.lower()) or ("visual studio code"
in texts.lower()):
                    engine.say("opening");
                    engine.say("Visual Studio Code")
                    engine.runAndWait()
                    os.startfile("Code")
```

4.2.7. Membuka aplikasi Visual Studio code

Pada tahap ini program membukakan otomatis aplikasi microsoft word cukup dengan perintah "*open*" diikuti dengan "*Code*", maka program akan membuka kan aplikasi tersebut jika aplikasinya memang ada pada *windows*.

```
elif ("vscode" in texts.lower()) or ("visual studio code" in
texts.lower()):
                    engine.say("opening");
                    engine.say("Visual Studio Code")
                    engine.runAndWait()
                    os.startfile("Code")
```

4.2.8. *Face Recognition*

Pada tahap ini membahas hasil dari *face recognition*, jika pada perintah terdapat kata "*who*" dan "*me*" maka program akan menjalankan fitur *face recognition*.

```
elif ("who" in texts.lower()) and ("me" in texts.lower()):
                cap = cv2.VideoCapture(0)
                stop_time = 0 #count for camera

                while True:
                    ret, frame = cap.read()

                    # Detect Faces
```

```
                face_locations, face_names =
sfrec.detect_known_faces(frame)
                for face_loc, name in zip(face_locations,
face_names):
                    y1, x2, y2, x1 = face_loc[0], face_loc[1],
face_loc[2], face_loc[3]

                    cv2.putText(frame, name,(x1, y1 - 10),
cv2.FONT_HERSHEY_DUPLEX, 1, (0, 0, 200), 2)
                    cv2.rectangle(frame, (x1, y1), (x2, y2), (0, 0,
200), 4)

                stop_time+=1
                cv2.imshow("Frame", frame)

                key = cv2.waitKey(1)
                if stop_time == 50:
                    cap.release()
                    cv2.destroyAllWindows()
                    break

            if name != "Unknown" :
                engine.say("You Are ")
                engine.say(name)
                engine.say("Hello")
                engine.say(name)
                engine.say("how can I help You ?")
                sg.PopupNonBlocking("You Are "+ name, "Hello
"+name, "How can I help you ?")
            else :
                engine.say("I dont recognize you")
                engine.say("your face is not in my dataset, but")
                engine.say("how can I help you ?")
                sg.PopupNonBlocking("I dont recognize you",
"Your face is not in my dataset", "But, how can I help you ?")
```

Setelah itu program akan membuka kamera menggunakan pada variabel cap, dan setelah kamera dibuka program akan masuk kedalam loop *while* dan menjalankan class *detect_known_face* yang terdapat pada file *simple_facerec* yang bertugas untuk membandingkan wajah yang tertangkap kamera dengan yang ada pada *dataset*, jika wajah dikenali maka program akan menampilkan nama dari pemilik wajah, dan jika tidak maka nama yang tampil adalah *unknown*.

```
def detect_known_faces(self, frame):
    small_frame = cv2.resize(frame, (0, 0), fx=self.frame_resizing, fy=self.frame_resizing)
    # Find all the faces and face encodings in the current frame of video
    # Convert the image from BGR color (which OpenCV uses) to RGB color (which face_recognition uses)
    rgb_small_frame = cv2.cvtColor(small_frame, cv2.COLOR_BGR2RGB)
    face_locations = face_recognition.face_locations(rgb_small_frame)
    face_encodings = face_recognition.face_encodings(rgb_small_frame, face_locations)

    face_names = []
    for face_encoding in face_encodings:
        # See if the face is a match for the known face(s)
        matches = face_recognition.compare_faces(self.known_face_encodings, face_encoding)
        name = "Unknown"

        # # If a match was found in known_face_encodings, just use the first one.
        # if True in matches:
        #     first_match_index = matches.index(True)
        #     name = known_face_names[first_match_index]

        # Or instead, use the known face with the smallest distance to the new face
        face_distances = face_recognition.face_distance(self.known_face_encodings, face_encoding)
        best_match_index = np.argmin(face_distances)
        if matches[best_match_index]:
            name = self.known_face_names[best_match_index]
        face_names.append(name)

    # Convert to numpy array to adjust coordinates with frame resizing quickly
    face_locations = np.array(face_locations)
    face_locations = face_locations / self.frame_resizing
    print(face_locations.astype(int))
    return face_locations.astype(int), face_names
```

Pada potongan kode diatas warna pada gambar yang tertangkap *frame* kamera akan di *convert* oleh modul opencv dari warna RGB menjadi BGR supaya bisa di baca oleh modul *face recognition* dan di *encode* oleh modul *face recognition*, setelah gambar di *convert* maka program akan masuk ke *for loop* yang akan membandingkan semua wajah yang tertangkap pada kamera dan dibandingkan dengan gambar yang ada di *dataset*, jika hasilnya match maka yang akan muncul nama dari *dataset*, dan jika wajah tidak di kenali maka hasilnya akan *unknown*.

4.2.9. Pencarian Lokasi Tempat

Pada tahap ini membahas hasil dari pencarian lokasi dari sebuah tempat yang diinginkan oleh pengguna, fitur ini dijalankan ketika dalam kata kunci terdapat kata "*where*", maka program akan menjalankan fitur ini.

```
elif ("where" in texts.lower()):
                query = texts.lower().replace('search', ' ')
                url =
"https://www.google.com/maps/place/{}".format(query)
                webbrowser.open_new_tab(url)
                engine.say('searching')
                engine.say(query)
```

4.3. Tabel Pengujian

Pengujian pada program *virtual assistant* dilakukan dengan metode *blackbox testing*. Adapun hasilnya dapat dilihat pada tabel 4.1.

Tabel 4. 1 Pengujian program menggunakan metode *blackbox testing*

No	Pengujian	Hasil	Sesuai/ Tidak Sesuai
1	Memasukkan persamaan matematika 300*400%7	6	✓

2	Menggunakan face recognition untuk mengenali wajah Dwiki	Hasil sesuai karena wajah Dwiki sudah ada pada dataset	✓
3	Menggunakan face recognition untuk mengenali wajah Ibnu	Hasil tidak sesuai karena wajah Ibnu belum terdapat pada dataset	×
4	Menggunakan fitur suhu untuk mengecheck suhu saat ini	Hasil sesuai	✓
5	Menggunakan fitur speech recognition dengan mengatakan “ZZZZZZZZZZ”	Program mengatakan bahwa suara tidak jelas dan meminta pengguna mencoba ulang	×
6	Menggunakan fitur speech recognition untuk mencari nama Dwiki Aldani	Hasil tidak ditemukan karena nama Dwiki Aldani tidak terdapat pada artikel manapun	×
7	Mencari lokasi dari tangerang selatan	Hasil sesuai, program membuka google chrome dan menunjukkan peta kota tangerang selatan	✓
8	Mencari video musik snowman	Hasil sesuai, program membuka browser dan membuka youtube untuk mencari video snowman	✓
9	Meminta program mengambil screenshot	Hasil sesuai, program mengambil screenshot dari layar terkini	✓
10	Meminta program unutk menyanyikan lagu	Hasil tidak sesuai, karena pada program belum terdapat fitur bernyanyi	×
11	Meminta program untuk membacakan sejarah bumi	Hasil sesuai, program mencari dengan kata kunci bumi pada modul wolframalpha dan membacakan hasilnya	✓

12	Memanggil nama program “Python VA” untuk menghidupkan program	Hasil tidak sesuai karena program belum memiliki fitur real-time activation speech recognition	×
13	Mencari kata gold pada program virtual assistant	Hasil sesuai, program menjelaskan apa itu emas danmembacakannya untuk pengguna	✓

Sumber: hasil penelitian 2022

BAB 5

PENUTUP

5.1. Kesimpulan

Berikut ini merupakan hasil yang didapatkan setelah adanya penelitian ini, antara lain:

1. Rancangan User Interface yang akan digunakan pada program virtual assistant dapat dibuat pada aplikasi CorelDRAW X8.
2. Dengan menggunakan pysimplegui dapat menghasilkan *User Interface* (UI) yang menjadi penghubung antara program virtual assistant dengan pengguna.
3. Fitur face recognition dapat ditambahkan pada program dan dapat berfungsi dengan baik, dan dengan tingkat keakuratan yang cukup tinggi.

5.2. Saran

Pada peneltian ini masih terdapat banyak kekurangan seperti fitur pencarian dengan hasil gambar, pengaktifan secara real-time dengan hanya mengucapkan kata saja tanpa harus menekan button. Sehingga diharapkan adanya penelitian selanjutnya yang dapat mengoptimalkan potensi dari program *virtual assistant* ini.

DAFTAR PUSTAKA

, M., Hidayat, S., & Amrullah, A. Z. (2019). Speech Recognition Untuk Aplikasi Kamus Bahasa Indonesia-Sumbawa Berbasis Android. *Jurnal Bumigora Information Technology (BITe)*, *1*(2), 126–137. https://doi.org/10.30812/bite.v1i2.606

(Faiz Zamzami & Nabella Duta Nusa, 2017 : 32. (2018). Bab Ii Landasan Teori. *Journal of Chemical Information and Modeling*, *53*(9), 8–24.

Baidi, M. H. (n.d.). *Kata Kunci: Pengembangan, media pembelajaran, membaca puisi*. *c*.

Blank, J., & Deb, K. (2020). Pymoo: Multi-Objective Optimization in Python. *IEEE Access*, *8*, 89497–89509. https://doi.org/10.1109/ACCESS.2020.2990567

Campagna, G., Xu, S., Moradshahi, M., Socher, R., & Lam, M. S. (2019). Genie: A generator of natural language semantic parsers for virtual assistant commands. *Proceedings of the ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI)*, 394–410. https://doi.org/10.1145/3314221.3314594

Cha, Y., Kim, J., Hyun, C. T., & Han, S. (2018). Development of a Program Definition Rating Index for the performance prediction of construction programs. *Sustainability (Switzerland)*, *10*(8). https://doi.org/10.3390/su10082747

Chung, H., & Lee, S. (2018). *Intelligent Virtual Assistant knows Your Life*. 1–6. http://arxiv.org/abs/1803.00466

Henri. (2018). 済無No Title No Title No Title. *Angewandte Chemie International Edition, 6(11), 951–952.*, 32–54.

Huda, I. (2021). Implementasi Natural Language Processing (Nlp) Untuk Aplikasi Pencarian Lokasi. *Jurnal Nasional Teknologi Terapan (JNTT)*, *3*(2), 15. https://doi.org/10.22146/jntt.35036

Husamuddin, H., Prasetyo, D. B., & Rustamadji, H. C. (2020). Otomatisasi Layanan Frequently Ask Questions Berbasis Natural Langugae Processing Pada Telegram Bot. *Telematika*, *17*(2), 145. https://doi.org/10.31315/telematika.v1i1.3383

Iii, B. A. B. (2019). *Implementation Evalution Design*. 12–17.

Judithia, D. (2019). Proses Adaptasi Ikatan Mahasiswa Fakfak Di Kota Bandung. *Journal of Chemical Information and Modeling*, *53*(9), 54–69. https://elibrary.unikom.ac.id/id/eprint/1558/

Li, L., Mu, X., Li, S., & Peng, H. (2020). A Review of Face Recognition Technology. *IEEE Access*, *8*, 139110–139120. https://doi.org/10.1109/ACCESS.2020.3011028

Nurjaya, W., & Riyanto, A. (2018). Analisis Dan Penerapan Search Engine Optimization Pada Website Menggunakan Metode White Hat SEO (Studi Kasus Di PT. Surya Putra Adipradana). *Jurnal Teknologi Informasi*, *1*(1), 1–6.

Perdana, R. P., & Irwansyah, I. (2019). Implementasi Asisten Virtual Dalam Komunikasi Pelayanan Pelanggan (Studi Kasus Pada Layanan Pelanggan Telkomsel). *Jurnal Komunikasi*, *11*(2), 183. https://doi.org/10.24912/jk.v11i2.5491

Saputri, E. (2021). Strategi Penelusuran Informasi Melalui Search Engine (Google). *Jurnal Adabiya*, *23*(2), 232. https://doi.org/10.22373/adabiya.v23i2.10137

Sayyed, A., Sangamnere, S., & Bhangale, J. H. (2021). Desktop Assistant AI Using Python. *International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)*, *6*(2). https://doi.org/10.48175/568

Snadhika Jaya, T. (2018). Pengujian Aplikasi dengan Metode Blackbox Testing Boundary Value Analysis (Studi Kasus: Kantor Digital Politeknik Negeri Lampung). *Jurnal Informatika: Jurnal Pengembangan IT (JPIT)*, *03*(02), 45–48. https://doi.org/10.30591/jpit.v3i1.647

Soufitri, F. (2019). Perancangan Data Flow Diagram Untuk Sistem Informasi Sekolah (Studi Kasus Pada Smp Plus Terpadu). *Ready Star*, *2*(1), 240–246.

Syahrudin, A. N., & Kurniawan, T. (2018). Input Dan Output Pada Bahasa. *Jurnal Dasar Pemrograman Python STMIK*, *January*, 1–7.

Utomo, A. N., & Alfaridzi, M. (2018). Perancangan Sistem Informasi Pada Percetakan CV Citra Kencana Jakarta Timur Berbasis Web. *Jurnal Rekayasa Informasi*, *7*(1), ISSN 2252-7354.

Utomo, B. T., Fitri, I., & Mardiani, E. (2020). Penerapan Face Recognition Pada Aplikasi Akademik Online. *Informatik : Jurnal Ilmu Komputer*, *16*(3), 195. https://doi.org/10.52958/iftk.v16i3.2259

V., G., C K, G., Kottamasu, M. S. V., & Kumar, N. P. (2021). The Voice Enabled Personal Assistant for Pc using Python. *International Journal of Engineering and Advanced Technology*, *10*(4), 162–165. https://doi.org/10.35940/ijeat.d2425.0410421

Wijaya, Y. D., & Astuti, M. W. (2021). Pengujian Blackbox Sistem Informasi Penilaian Kinerja Karyawan Pt Inka (Persero) Berbasis Equivalence Partitions. *Jurnal Digital Teknologi Informasi*, *4*(1), 22. https://doi.org/10.32502/digital.v4i1.3163

Dwilestari, S., & Nurmiati, S. (2018). Sistem Pakar Penentuan Style Pada Tanaman Bonsai Menggunakan Metode Certainty Factor. *Sainstech: Jurnal Penelitian Dan Pengkajian Sains Dan Teknologi*, *28*(2), 49–56. https://doi.org/10.37277/stch.v28i2.242

LAMPIRAN

LAMPIRAN-01

a. pythonVA.py

```
#source for searching
import wolframalpha
client = wolframalpha.Client("964PPA-8G9GGPL9YA")
import wikipedia

#webbrowser for searching with google
import webbrowser

#for path
import glob

#taking screenshot
import pyautogui

#for startfile opening apps
import os;

#opening youtube
import pywhatkit;

#facerecognition module
import cv2;
from simple_facerec import SimpleFacerecognition

sfrec = SimpleFacerecognition()
sfrec.load_encoding_images("source code/images/")

#for layout setting
import PySimpleGUI as sg
sg.theme('DarkBlue13')
layout =[[sg.Image(r'"logoVA.png"', size=(500,300))],[sg.Text('Enter a Statement or Question')], [sg.InputText(size=(65,1)), sg.Button('Speech')],[sg.Button('Search', size=(10,1)), sg.Button('Cancel', size=(10,1))]]
window = sg.Window('PyVA', layout, size=(550,500))

#for system sound
import pyttsx3
engine = pyttsx3.init()
voices = engine.getProperty('voices')
engine.setProperty('voice', voices[0].id)

#speech recognition
import speech_recognition as sr
recognize = sr.Recognizer()
```

```
while True:
    event, values = window.read()

    if event in (None, 'Cancel'):
        break

    if event == 'Speech' :
        engine.say("im listening")
        sg.PopupNonBlocking("im listening...")
        engine.runAndWait()

        try :
            with sr.Microphone() as source:
                audio = recognize.listen(source)
                texts = recognize.recognize_google(audio)
                if ("open" in texts.lower()) :
                    if ("google" in texts.lower()) or ("chrome" in texts.lower()) or
("google chrome" in texts.lower()) or ("browser" in texts.lower()) :
                        engine.say("opening");
                        engine.say("Google Chrome");
                        engine.runAndWait();
                        os.startfile("chrome");
                    else:
                        erroropening = "Program unavailable, Make sure your windows
recognized the program."
                        engine.say(erroropening)
                        sg.PopupNonBlocking(erroropening)

                elif ("search" in texts.lower()):
                    query = texts.lower().replace('search', ' ')
                    url = "https://www.google.com.tr/search?q={}".format(query)
                    webbrowser.open_new_tab(url)
                    engine.say('searching')
                    engine.say(query)

                elif ("where" in texts.lower()):
                    query = texts.lower().replace('search', ' ')
                    url = "https://www.google.com/maps/place/{}".format(query)
                    webbrowser.open_new_tab(url)
                    engine.say('searching')
                    engine.say(query)

                elif ("play" in texts.lower()):
                    video = texts.lower().replace('play',' ')
                    engine.say('playing')
                    engine.say(video)
                    pywhatkit.playonyt(video)
```

```
            elif ("screenshot" in texts.lower()):
                save_path = glob.glob(os.path.join('screenshot/', "*.*"))
                save_path_len = len(save_path)
                Screenshot = pyautogui.screenshot()
                Screenshot.save('screenshot/'+str(save_path_len+1)+'.png')
                engine.say('succesfull')

            elif ("introduce yourself" in texts.lower()):
                engine.say("Hello")
                engine.say("i am your personal assistant")
                engine.say("python V A")

            elif ("who" in texts.lower()) and ("me" in texts.lower()):
                cap = cv2.VideoCapture(0)
                stop_time = 0 #count for camera

                while True:
                    ret, frame = cap.read()

                    # Detect Faces
                    face_locations, face_names = sfrec.detect_known_faces(frame)
                    for face_loc, name in zip(face_locations, face_names):
                        y1, x2, y2, x1 = face_loc[0], face_loc[1], face_loc[2],
face_loc[3]

                        cv2.putText(frame, name,(x1, y1 - 10),
cv2.FONT_HERSHEY_DUPLEX, 1, (0, 0, 200), 2)
                        cv2.rectangle(frame, (x1, y1), (x2, y2), (0, 0, 200), 4)

                    stop_time+=1
                    cv2.imshow("Frame", frame)

                    key = cv2.waitKey(1)
                    if stop_time == 50:
                        cap.release()
                        cv2.destroyAllWindows()
                        break

                if name != "Unknown" :
                    engine.say("You Are ")
                    engine.say(name)
                    engine.say("Hello")
                    engine.say(name)
                    engine.say("how can I help You ?")
                    sg.PopupNonBlocking("You Are "+ name, "Hello "+name,
"How can I help you ?")
                else :
```

```
                engine.say("I dont recognize you")
                engine.say("your face is not in my dataset, but")
                engine.say("how can I help you ?")
                sg.PopupNonBlocking("I dont recognize you", "Your face is not
in my dataset", "But, how can I help you ?")

            elif ("exit" in texts.lower()):
                engine.say('exiting')
                break

            else :
                try:
                    rawres_wolfram = client.query(texts).results
                    result2_wikipedia = wikipedia.summary(texts,
auto_suggest=False, sentences=2)
                    result1_wolframalpha = next(rawres_wolfram).text
                    engine.say(result1_wolframalpha)
                    sg.PopupNonBlocking("Wolfram Result:
"+result1_wolframalpha,"Wikipedia Result: "+result2_wikipedia)
                except wikipedia.exceptions.DisambiguationError:
                    try :
                        result1_wolframalpha = next(rawres_wolfram).text
                        engine.say(result1_wolframalpha)
                        sg.PopupNonBlocking(result1_wolframalpha)
                    except:
                        message = "Result Not found ! Try Another Words";
                        engine.say(message);
                        sg.PopupNonBlocking(message);
                except wikipedia.exceptions.PageError:
                    try :
                        result1_wolframalpha = next(rawres_wolfram).text
                        engine.say(result1_wolframalpha)
                        sg.PopupNonBlocking(result1_wolframalpha)
                    except:
                        message = "Result Not found ! Try Another Words";
                        engine.say(message);
                        sg.PopupNonBlocking(message);
                except:
                    try :
                        result2_wikipedia = wikipedia.summary(texts,
auto_suggest=False, sentences=2)
                        engine.say(result2_wikipedia)
                        sg.PopupNonBlocking(result2_wikipedia)
                    except:
                        message = "Result Not found ! Try Another Words";
                        engine.say(message);
                        sg.PopupNonBlocking(message);
        except :
```

```
            message = "unclear voice, try again !"
            engine.say(message)


    if event == 'Search' :
        texts = ""

        if ("open " in values[1].lower()) :
            if ("google" in values[1].lower()) or ("chrome" in values[1].lower()) or
("google chrome" in values[1].lower()) or ("browser" in values[1].lower()) :
                engine.say("opening");
                engine.say("Google Chrome");
                engine.runAndWait();
                os.startfile("chrome");
            else:
                erroropening = "Program unavailable, Make sure your windows
recognized the program."
                engine.say(erroropening)
                sg.PopupNonBlocking(erroropening)

        elif ("search" in values[1].lower()):
            query = values[1].lower().replace('search', ' ')
            url = "https://www.google.com.tr/search?q={}".format(query)
            webbrowser.open_new_tab(url)
            engine.say('searching')
            engine.say(query)

        elif ("where is" in values[1].lower()):
            query = values[1].lower().replace('where is', ' ')
            url = "https://www.google.com/maps/place/{}".format(query)
            webbrowser.open_new_tab(url)
            engine.say('searching')
            engine.say(query)

        elif ("screenshot" in values[1].lower()):
            save_path = glob.glob(os.path.join('screenshot/', "*.*"))
            save_path_len = len(save_path)
            Screenshot = pyautogui.screenshot()
            Screenshot.save('screenshot/'+str(save_path_len+1)+'.png')
            engine.say('succesfull')

        elif ("introduce yourself" in values[1].lower()):
            engine.say("Hello")
            engine.say("i am your personal assistant")
            engine.say("python V A")

        elif("play" in values[1].lower()):
            video = values[1].lower().replace('play',' ')
```

```
            engine.say('playing')
            engine.say(video)
            pywhatkit.playonyt(video)

        elif ("who" in values[1].lower()) and ("me" in values[1].lower()):
            cap = cv2.VideoCapture(0)
            stop_time = 0 #count for camera

            while True:
                ret, frame = cap.read()

                # Detect Faces
                face_locations, face_names = sfrec.detect_known_faces(frame)
                for face_loc, name in zip(face_locations, face_names):
                    y1, x2, y2, x1 = face_loc[0], face_loc[1], face_loc[2], face_loc[3]

                    cv2.putText(frame, name,(x1, y1 - 10), cv2.FONT_HERSHEY_DUPLEX, 1, (0, 0, 200), 2)
                    cv2.rectangle(frame, (x1, y1), (x2, y2), (0, 0, 200), 4)

                stop_time+=1
                cv2.imshow("face recognition", frame)

                key = cv2.waitKey(1)
                if stop_time == 50:
                    cap.release()
                    cv2.destroyAllWindows()
                    break

            if name != "Unknown" :
                engine.say("You Are ")
                engine.say(name)
                engine.say("Hello")
                engine.say(name)
                engine.say("how can I help You ?")
                sg.PopupNonBlocking("You Are "+ name, "Hello "+name, "How can I help you ?")
            else :
                engine.say("I dont recognize you")
                engine.say("your face is not in my dataset, but")
                engine.say("how can I help you ?")
                sg.PopupNonBlocking("I dont recognize you", "Your face is not in my dataset", "But, how can I help you ?")
        elif("exit" in values[1].lower()):
            engine.say('exiting')
            break
        else :
            try:
```

```
            rawres_wolfram = client.query(values[1]).results
            result2_wikipedia = wikipedia.summary(values[1],
auto_suggest=False, sentences=2)
            result1_wolframalpha = next(rawres_wolfram).text
            engine.say(result1_wolframalpha)
            sg.PopupNonBlocking("Wolfram Result:
"+result1_wolframalpha,"Wikipedia Result: "+result2_wikipedia)
         except wikipedia.exceptions.DisambiguationError:
            try :
               result1_wolframalpha = next(rawres_wolfram).text
               engine.say(result1_wolframalpha)
               sg.PopupNonBlocking(result1_wolframalpha)
            except:
               message = "Result Not found ! Try Another Words";
               engine.say(message);
               sg.PopupNonBlocking(message);
         except wikipedia.exceptions.PageError:
            try :
               result1_wolframalpha = next(rawres_wolfram).text
               engine.say(result1_wolframalpha)
               sg.PopupNonBlocking(result1_wolframalpha)
            except:
               message = "Result Not found ! Try Another Words";
               engine.say(message);
               sg.PopupNonBlocking(message);
         except:
            try :
               result2_wikipedia = wikipedia.summary(values[1],
auto_suggest=False, sentences=2)
               engine.say(result2_wikipedia)
               sg.PopupNonBlocking(result2_wikipedia)
            except:
               message = "Result Not found ! Try Another Words";
               engine.say(message);
               sg.PopupNonBlocking(message);
   engine.runAndWait()
   print (values[1]+" "+texts)
engine.runAndWait()
window.close()
```

b. simple_facerec.py

```
import face_recognition
import cv2
import os
import glob
import numpy as np

class SimpleFacerecognition:
    def __init__(self):
        self.known_face_encodings = []
        self.known_face_names = []

        # Resize frame for a faster speed
        self.frame_resizing = 0.25

    def load_encoding_images(self, images_path):
        """
        Load encoding images from path
        :param images_path:
        :return:
        """
        # Load Images
        images_path = glob.glob(os.path.join(images_path, "*.*"))

        print("{} encode gambar ditemukan.".format(len(images_path)))

        # Store image encoding and names
        for img_path in images_path:
            img = cv2.imread(img_path)
            rgb_img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

            # Get the filename only from the initial file path.
            basename = os.path.basename(img_path)
            (filename, ext) = os.path.splitext(basename)
            # Get encoding
            img_encoding = face_recognition.face_encodings(rgb_img)[0]

            # Store file name and file encoding
            self.known_face_encodings.append(img_encoding)
            self.known_face_names.append(filename)
        print("Encode gambar telah dimuat")

    def detect_known_faces(self, frame):
        small_frame = cv2.resize(frame, (0, 0), fx=self.frame_resizing,
fy=self.frame_resizing)
        # Find all the faces and face encodings in the current frame of video
```

```
        # Convert the image from BGR color (which OpenCV uses) to RGB color (which face_recognition uses)
        rgb_small_frame = cv2.cvtColor(small_frame, cv2.COLOR_BGR2RGB)
        face_locations = face_recognition.face_locations(rgb_small_frame)
        face_encodings = face_recognition.face_encodings(rgb_small_frame, face_locations)

        face_names = []
        for face_encoding in face_encodings:
            # See if the face is a match for the known face(s)
            matches = face_recognition.compare_faces(self.known_face_encodings, face_encoding)
            name = "Unknown"

            # # If a match was found in known_face_encodings, just use the first one.
            # if True in matches:
            #     first_match_index = matches.index(True)
            #     name = known_face_names[first_match_index]

            # Or instead, use the known face with the smallest distance to the new face
            face_distances = face_recognition.face_distance(self.known_face_encodings, face_encoding)
            best_match_index = np.argmin(face_distances)
            if matches[best_match_index]:
                name = self.known_face_names[best_match_index]
            face_names.append(name)

        # Convert to numpy array to adjust coordinates with frame resizing quickly
        face_locations = np.array(face_locations)
        face_locations = face_locations / self.frame_resizing
        print(face_locations.astype(int))
        return face_locations.astype(int), face_names
```

LAMPIRAN-02

ISTN	**LEMBAR KONSULTASI BIMBINGAN SKRIPSI**
	TEKNIK INFORMATIKA **INSTITUT SAINS DAN TEKNOLOGI NASIONAL**

Nama Mahasiswa : Dwiki Aldani
NPM : 18360028
Dosen Pembimbing : Siti Nurmiati, S.Kom., M.Kom.
Judul Skripsi : Pengembangan Program Virtual Assistant Dengan Menggunakan Python

No	Tanggal Bimbingan	Materi Bimbingan	Paraf Pembimbing
1.	25 Oktober 2021	Bimbingan Bab 1	
2.	26 Oktober 2021	Bimbingan Bab 2	
3.	03 November 2021	Bimbingan Bab 3	
4.	19 Januari 2022	Pengecheckan program dan penulisan Bab 1-3	
5.	21 Januari 2022	Bimbingan penulisan Bab 3	
6.	25 Januari 2022	Bimbingan penulisan Bab 4	
7.	26 Februari 2022	Bimbingan perubahan pada Bab 1-3	
8.	27 Februari 2022	Bimbingan perubahan pada Bab 4 dan 5	
9.	28 Februari 2022	Bimbingan penulisan Bab 1-5	

10.	01 Maret 2022	Bimbingan final	

Catatan :
Total bimbingan yang dilakukan adalah 10 (sepuluh) kali pertemuan.

- Bimbingan dimulai dari tanggal : 19 November 2022
- Bimbingan diakhiri pada tanggal : 01 Maret 2022

Jakarta, 02 Maret 2022

Dosen Pembimbing

: Siti Nurmiati, S.Kom., M.Kom.

NIDN. 04.021077.03