BAB II
TINJAUAN PUSTAKA

2.1 Penelitian Terdahulu

Bahasa isyarat digunakan dalam komunikasi antara tunarungu dan tunawicara dengan masyarakat umum menjadi permasalah sosial yang sering terjadi (Sihabudin, 2022:2). Minimnya pembelajaran bahasa isyarat bagi masyarakat umum membuat komunikasi dan interaksi yang dilakukan dengan penyandang disabilitas, sehingga membuat komunikasi dan interaksi yang dilakukan dengan penyandang disabilitas, menghalangi mereka untuk berpartisipasi penuh dalam kehidupan masyarakat.

Melalui teknologi baru pengenalan gerakan tangan (hand tracking) maka media pembelajaran bahasa isyarat yang selama ini melalui aplikasi dapat digantikan (Nababan, dan Burdiarso, 2023: 1). Dengan menggunakan Deep Learning kita dapat memanfaatkan teknologi dengan menggunakan library dari Tensorflow dan akan menggunakan metode Convolutional Neural Network (CNN) dan menggunakan dataset untuk mendeteksi pergerakan kedua lengan, dan jari. CNN akan menggunakan arsitektur MobileNetV3 untuk optimalisasi kinerja model yang akan di kembangkan pada perangkat mobile.

Dasarnya deteksi gerakan akan mendeteksi citra video yang merupakan kumpulan citra-citra yang direkam secara frame by frame yang membentuk pergerakan dinamis (Putri, Fadlisyah, Fuadi, 2022:664-665). Untuk model dapat mengambil gerakan video diperlukan proses yang panjang untuk model mempelajari video yang diterima. Untuk memproses video tersebut juga akan dilakukan proses Confusion Matrix. Confusion matrix berisi semua informasi mentah tentang prediksi yang dilakukan oleh sebuah model klasifikasi pada kumpulan data tertentu (Abidin, 2023:12).

Dari melalui tahapan proses pelatihan, model dievaluasi untuk mengukur kinerja yang dihasilkan oleh data testing, kemudian hasil dari proses training dan testing akan memiliki matriks evaluasi (Widjaya & Wasito, 2024:1487). Akurasi yang diberikan dari model

tergantung dari bagaimana video atau gambar itu diolah dalam pemrosesan citra.

Tabel 2.1 Tabel Penelitian Terdahulu

Penulis	Judul	Metode	Gap
Yunus, M. & Anwar, Y. (2022)	Aplikasi Penerjemahan Bahasa Isyarat Indonesia ke dalam Huruf Abjad	Mengembangkan aplikasi penerjemah bahasa isyarat BISINDO ke dalam huruf abjad menggunakan model CNN agar dapat digunakan oleh masyarakat luas	Menggunakan model Convolutional Neural Network (CNN) dengan arsitektur MobileNetV2. Model dikonversi ke TensorFlow Lite untuk digunakan di Android Studio
Hikmatia, N. & Zul, I. M. (2021)	Aplikasi Penerjemah Bahasa Isyarat Indonesia menjadi Suara berbasis *Android* menggunakan *Tensorflow*	Mengembangkan aplikasi yang menerjemahkan bahasa isyarat BISINDO menjadi suara untuk mempermudah komunikasi dengan masyarakat umum	Menggunakan metode *Convolutional Neural Network* (*CNN*) dengan arsitektur *MobileNetV2*. Model dikonversi ke *TensorFlow Lite* dan diintegrasikan dengan *text-to-speech*
Pramono, I. M., Niswati, Z., & Agustina, A. (2024)	Model Penerjemah Bahasa Isyarat Indonesia dengan Metode *Convolutional*	Mengembangkan sistem penerjemah bahasa isyarat Indonesia berbasis komputer dengan *Python Hand*	Menggunakan metode *Convolutional Neural Network* (*CNN*) untuk mendeteksi gerakan

Penulis	Judul	Metode	Gap
	Neural Network (*CNN*)	*Gesture Recognition* menggunakan metode *CNN* pada video *realtime*	tangan dalam video *real-time*. Model dikembangkan dengan *Python Hand Gesture Recognition* dan diuji dengan data video langsung
Gustiar, D., Sitorus, S. H., & Midyanti, D. M. (2020)	Penerjemahan Bahasa Isyarat Menggunakan Metode *Generalized Learning Vector Quantization* (*GLVQ*)	Mengembangkan sistem penerjemah bahasa isyarat SIBI dengan metode *GLVQ* untuk mengenali pola huruf secara otomatis	Menggunakan model *Generalized Learning Vector Quantization* (*GLVQ*) berbasis jaringan saraf tiruan.

Penelitian ini bertujuan untuk mengembangkan aplikasi terjemahan bahasa isyarat dengan menggunakan model *MobileNetV3* dari arsitektur *Convolutional Neural Network* (CNN) untuk mengenali pola tangan dari pengguna, dengan menggunakan dataset yang dikembangkan sendiri. Perbedaan dengan penelitian sebelumnya yang hanya berfokus pada huruf saja tidak dengan pola isyarat yang rumit.

2.2 Landasan Teori

2.2.1 Sistem Isyarat Bahasa Indonesia (SIBI)

Sistem Isyarat Bahasa Indonesia (SIBI) merupakan salah satu jenis aplikasi bahasa isyarat yang umum digunakan oleh penyandang disabilitas tunarungu dan tunawicara yang juga telah dibakukan sebagai media komunikasi nasional bagi penyandang disabilitas. Terhitung 211.889 penduduk indonesia merupakan penyandang disabilitas yang terdiri dari 6.5% (13.802) adalah tunarungu dan 2.6%

(5.580) adalah tunawicara (Sindarto, Ratnawati, dan Arwani, 2022:2130).

SIBI memiliki sistem yang berbasis pada tata bahasa Bahasa Indonesia yang terstruktur, sehingga dalam penyampaian gerakan kalimat yang sama dengan bahasa tulis dan lisan. Isi dari SIBI mencakup:

1) Huruf abjad, dimana dalam sibi menggunakan sistem pengejaan jari untuk mengeja nama atau kata-kata tertentu yang belum memiliki gerakan isyaratnya sendiri, biasanya digunakan dalam pengejaan nama atau nama tempat.
2) Kata umum, berisikan kosa kata yang sudah memiliki gerakan isyaratnya sendiri yang akan digunakan untuk komunikasi sehari-hari. Kata yang ada merupakan kata ganti, kata kerja, dan kata benda.

SIBI sebagai jembatan komunikasi bagi penyandang disabilitas namun tidak dapat dipastikan hal ini dapat menyelesaikan masalah kesenjangan sosial, karena masih banyak orang awam yang tidak mengerti mengenai bahasa isyarat. Jadi untuk membantu mengurangi kesenjangan sosial dari komunikasi antara penyandang disabilitas dengan masyarakat umum dibuatkan sebuah aplikasi yang dapat menerjemahkan bahasa isyarat melalui perangkat kamera menjadi tulisan yang dapat dibaca oleh masyarakat umum.

2.2.2 Aplikasi Penerjemah

Aplikasi penerjemahan biasanya digunakan sebagai alat untuk menerjemahkan antara bahasa berbagai negara yang biasanya akan dilakukan dengan menginputkan kata atau kalimat yang ingin diterjemahkan. Berbeda dengan aplikasi penerjemah yang akan difokuskan untuk menjembatani komunikasi penyandang disabilitas, yang dimana aplikasi perbedaannya terdapat pada hasil terjemahannya. Contoh aplikasi yang serupa adalah "*Hear Me*", sebuah aplikasi startup sosial yang menyediakan teknologi penerjemahan Bahasa Indonesia ke BISINDO yang digunakan untuk penyandang disabilitas (Hear Me, n.d.).

Berbeda untuk aplikasi pada proyek ini yang dimana aplikasi ini akan diperuntukan bagi masyarakat yang ingin melakukan komunikasi dengan penyandang disabilitas melalui akses kamera

dengan mendeteksi gerakan individu yang mengidap disabilitas yang menggunakan SIBI.

2.2.3 *Action Recognition*

Action Recognition merupakan salah satu pekerjaan dari *computer vision*, yaitu akan melibatkan citra dari video atau gambar dalam mengenali tindakan dari manusia (Papers With Code, n.d). *Action Recognition* sudah banyak diterapkan dalam masa saat ini dimana dengan penggunaan dari *Deep Learning* sebagai alat untuk memproses *Action Recognition* tersebut. Kemampuan suatu teknologi dalam pengenalan dalam tindakan seseorang dimanfaatkan dalam berbagai bidang, sebagai contoh dalam keamanan atau pengawasan dengan memanfaatkan perangkat kamera *CCTV* untuk medekteksi seseorang. Batasan dari *Action Reccognition* berupa kompleksnya gerakan dari seseorang sehingga teknologi ini harus dikembangkan setiap saat untuk mendapatkan kemampuan yang bagus.

2.2.4 *Real-Time Processing*

Real-Time Processing merupakan metode yang memproses data yang dilakukan secara sangat cepat saat data diterima, sehingga proses yang dilakukan hampir terlihat secara instan. Proses instan yang diberikan membutuhkan proses yang langsung dilakukan saat data diterima tanpa terjadinya penundaan. *Latency* dari *Real-Time Processing* harus dalam waktu singkat dengan waktu sekitar kurang dari 1 detik (milidetik) untuk mendapatkan kecepatan yang diharapkan.

Proses cepat yang dilakukan membutuhkan kekuatan komputasi yang tinggi dan optimalisasi dalam *latency* untuk mendapatkan performa yang bagus. Oleh karena itu, sistem *real-time* sering kali menggunakan perangkat keras khusus seperti *Graphics Processing Unit* (GPU) untuk mempercepat pemrosesan.

Dalam implementasinya, sistem ini harus mampu menangani data dalam jumlah besar secara terus-menerus (stream processing) tanpa adanya bottleneck yang dapat memperlambat kinerja.

2.2.5 *Deep Learning*

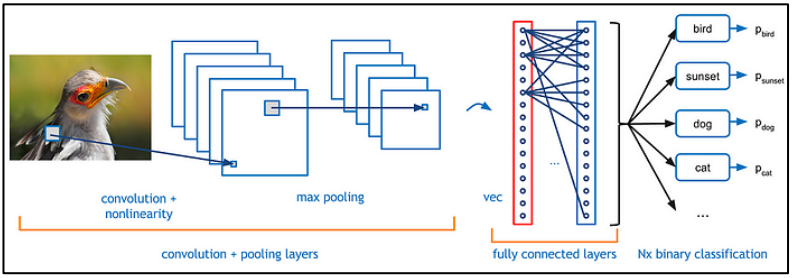
Deep learning merupakan ilmu *Artificial Intelligence* (AI) yang masih satu bidang dengan *Machine Learning*, teknologi ini berfokus pada penggunaan aplikasi *Deep Neural Network* yang digunakan untuk memodelkan dan memahami pola data yang kompleks. *Deep Learning* bermula dari algoritma bernama backpropagation yang dikembangkan oleh Geoffrey Hinton, Yoshua Bengio, dan Yann LeCun pada tahun 2006. Berkembangan *Deep Learning* berlangsung hingga tahun-tahun berikutnya bersamaan dengan perkembangan *Graphical Processing Unit* (GPU), sehingga dapat mempercepat proses dari *neural network* yang kompleks. Model yang melakukan pembelajaran mendalam memiliki kemampuan untuk mengidentifikasi pola rumit dalam berbagai bentuk data, seperti foto, teks, suara, dan lainnya, untuk menghasilkan wawasan dan prediksi yang tepat (Nurhakiki dan Yahfizham, 2024:272), jadi *Deep Learning* memiliki perbedaan mendasar dengan *Machine Learning*, dimana *Deep Learning* memiliki *neural network* yang lebih kompleks sehingga dapat memberikan prediksi yang lebih baik

Proses yang dilakukan oleh *machine learning* akan dibagi dari Input, *Feature Extraction, Classification*, dan *Output*. Proses tersebut masih memerlukan sebuah proses *Feature Extraction* dari data dilakukan secara manual oleh pengembang misalnya menentukan warna, panjang, atau lebar dari sebuah kendaraan untuk dimasukkan ke dalam pola *Machine Learning* untuk mendapatkan hasil sesuai ekspetasi. Berbeda dengan *Deep Learning* dimana proses *Feature Extraction* akan dilakukan bersamaan dengan proses *Classification* dan dilakukan secara otomatis oleh *neural network*, Sebagai contohnya, *Layers* atau lapisan-lapisan pada *neural network* seperti *Convolutional Neural Network* (CNN) secara otomatis mengambil fitur sederhana seperti garis dan sudut dala, dan fitur kompleks seperti warna kaca, warna mobil, dan ukuran roda dalam layer lebih dalam.

Teknologi ini dapat diterapkan untuk melakukan membangun aplikasi yang membutuhkan kemampuan analisis data tingkat lanjut. Misalnya dalam kasus proyek ini dimanfaatkan untuk mengenali pola gerakan jari dan lengan untuk menerjemahkan bahasa isyarat. Dengan bantuan dari adanya *library Tensorflow* yang membantu pengembangan *neural network* yang lebih dalam, sehingga dapat membangun model *Deep Learning* yang memperlajari data yang rumit dan di prediksi dengan akurat jika dibuat dengan benar.

2.2.6 *Convolutional Neural Network*

Convolutional Neural Network (CNN) adalah algoritma yang digunakan dalam pembuatan *Deep Learning*. Jenis *neural network* ini pertama kali dikenalkan oleh Yann LeCun dkk. pada tahun 1998 dalam makalah buatan mereka berjudul "*Gradient-Based Learning Applied to Document Recognition*". Versi awal dari CNN bernama LeNet yang diambil dari nama LeCun yang dalam pengujiannya berhasil mengenali tulisan tangan. Model ini digunakan untuk memproses data citra baik itu gambar atau video yang diambil frame per frame. CNN bekerja dengan menggunakan prinsip aljabar linear dan operasi convolution untuk mengektraksi fitur dari data citra, sehingga memungkinkan model untuk mengenali pola-pola visual yang kompleks secara otomatis.



Gambar 2.1 CNN layers (Sumber: Oktavian & Santoso, 2021)

Gambar 2.1 merupakan contoh dari process CNN. Dimulai dari proses ekstraksi fitur dari data yang diinputkan dari proses *Convolution Layer*, filter akan mendekteksi fitur spesifik dari gambar yang diinputkan, seperti pola tepi gambar, sudut, dan pola lainnya. Secara kompleks ekstraksi juga akan mengambil filter dari setiap fitur spesifik seperti warna, bentuk mata dan lainnya.

Proses *Convolution Layer* akan berjalan bersamaan dengan *Pooling Layer*, yaitu *layer* yang akan mengurangi dimensi data tanpa menghilangkan informasi fitur yang penting.

Setelah itu fitur akan di transfer menggunakan fitur *Vector*, yaitu representasi numerik dari fitur gambar menuju *Fully Connected Layers* dimana layer ini merupakan lapisan terakhir untuk menghasilkan probabilitas classification. Vector akan dicocokkan dengan probabilitas *classification* dan akan diambil probabilitas tertinggi dari model.

2.2.7 *Long Short-Term Memory*

Convolutional Neural Network (CNN) adalah algoritma yang digunakan dalam pembuatan *Deep Learning*. Jenis *neural network* ini pertama kali dikenalkan oleh Yann LeCun dkk. pada tahun 1998 dalam makalah buatan mereka berjudul "*Gradient-Based Learning Applied to Document Recognition*". Versi awal dari CNN bernama LeNet yang diambil dari nama LeCun yang dalam pengujiannya berhasil mengenali tulisan tangan. Model ini digunakan untuk memproses data citra baik itu gambar atau video yang diambil frame per frame. CNN bekerja dengan menggunakan prinsip aljabar linear dan operasi convolution untuk mengektraksi fitur dari data citra, sehingga memungkinkan model untuk mengenali pola-pola visual yang kompleks secara otomatis.

2.2.8 *MobileNetV3*

MobileNet merupakan sebuah arsitektur CNN yang dibuat untuk meningkatkan kecepatan dan akurasi model. Arsitektur ini dikembangkan oleh *Google* untuk pengembangan yang *neural network* untuk perangkat *mobile*, sehingga klasifikasi gambar, objek, atau lainnya dapat dilakukan dengan lebih optimal, tetap ringan, dan cepat dengan tetap mempertahankan akurasi prediksi pada perangkat *mobile*.

MobileNetV3 merupakan peningkatan terbaru setelah *MobileNetV2* dengan peningkatan berupa *Neural Architecture Search* (NAS) dan *NetAdapt*. Arsitektur ini digunakan karena dapat secara drastis mengurangi jumlah komputasi yang dilakukan dibandingan dengan *convolution* tradisional. Berbeda dengan pendahulunya, *MobileNetV3* menggunakan *Activation Function* bernama *Swish*, dimana ini merupakan perkembangan yang lebih baik dari pada *ReLU* yang digunakan oleh *MobileNetV2* dalam beberapa kasus. *Swish* menggunakan *sigmoid* atau *neural network* yang menghasilkan output dalam rentang 0 hingga 1, berikut ini merupakan rumus dari *sigmoid*:

$$\sigma(x) = \frac{1}{1 + e^{-}}$$

dimana x merupakan inputan dari *neuron* dan e merupakan bilangan euler. Sehingga dapat dibuat rumus dari *Swish* sebagai berikut:

$$f(x) = x \times \sigma(x)$$

Dalam Arsitektur *MobileNetV3*, Swish akan digunakan sebagai *Activation Function* utama yang bekerja di *convolution layer* untuk meningkatkan efisiensi model. *Swish* atau sering disingkat *h-swish* dalam *MobileNetV3* merupakan versi lebih optimal yang *Swish*

standar karena memiliki kemampuan lebih efisien secara komputasi dengan menggantikan *sigmoid* dengan *piecewise linear* dengan rumus sebagai berikut:

$$H - Swish(x) = x \times \frac{ReLU6(x+3)}{6}$$

dengan menggunakan *ReLU6* atau *Activation Function ReLU* yang telah diberikan batasan dengan rumus sebagai berikut:

$$ReLU6(x) = \min(\max(0, x), 6)$$

Dari rumus *ReLU6* ini memberikan memiliki output berupa:

1) Jika x<0, output tetap 0.
2) Jika 0 ⩽x ⩽6, output tetap xxx.
3) Jika x>6, output dikunci pada 6.

2.2.9 *Tensorflow*

Tensorflow merupakan *framework Deep Learning* dan juga *library* untuk mengelola data yang dapat digunakan secara gratis yang dikembangkan oleh *Google*. Dalam proyek ini *Tensorflow* dimanfaatkan untuk melakukan *object detection* yang akan digunakan pada deteksi gerakan jari dan lengan untuk bahasa isyarat, API tersebut merupakan suatu alat yang dapat digunakan untuk mempermudah proses *constructing*, *training* dan *development* pada model *machine learning*.

Framework Tensorflow object detection API menyediakan *pretrained object detection* model bagi user, namun memungkinkan jika user ingin menggunakan pretrained object detection model yang lain, seperti *Faster R-CNN, SSD, Retinanet, Resnet50* dan masih banyak lagi (Manajang, Sompie, dan Jacobus, 2020:173). Dengan menggunakan framework ini yang dikombinasikan dengan model yang dibuat dari CNN, diharapkan dapat mempercepat pengembangan pembelajaran model aplikasi.

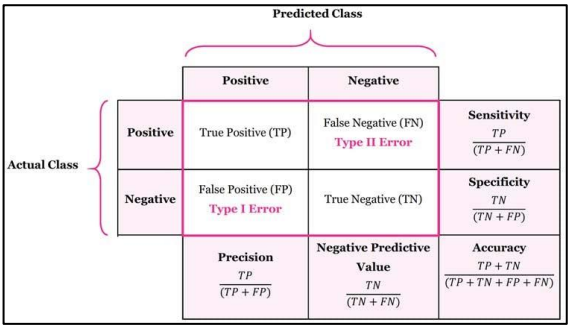
2.2.10 *OpenCV*

Open Source Computer Vision (OpenCV) merupakan *library* yang digunakan untuk pengolahan gambar dan pengenalan pola. *Library* ini diluncurkan pada tahun 1999 oleh *Inter Research* sebagai kelanjutan dari proyek yang menghasilkan aplikasi intensif berbasis *CPU*, *real-time ray tracing* dan tembok penampilan 3 Dimensi. Library ini disediakan secara gratis untuk digunakan dalam pengembangan yang menggunakan pengolahan citra.

OpenCV menyediakan berbagai fungsi untuk pengolahan citra, menjadikannya pilihan utama dalam pengembangan aplikasi yang memanfaatkan teknologi ini. Dengan dukungan berbagai bahasa pemrograman seperti *C++, Python, Java, dan MATLAB*, serta kompatibilitas dengan sistem operasi *Windows, Linux, Android*, dan *MacOS*. *OpenCV* memungkinkan pengolahan gambar atau video sesuai kebutuhan, termasuk integrasi dengan kamera. Sehingga proyek ini dapat menggunakan library ini untuk mempermudah pengembangan.

2.2.11 Confusion Matrix

Confusion Matrix adalah alat evaluasi yang digunakan dalam *machine learning* untuk menganalisis kinerja model klasifikasi. Matriks ini menyajikan informasi tentang jumlah prediksi yang benar (*True Positive* dan *True Negative*) serta jumlah kesalahan (*False Positive* dan *False Negative*) yang dilakukan oleh model. Struktur *Confusion Matrix* berbentuk tabel dengan dimensi , di mana adalah jumlah kelas dalam data.



Gambar 2.2 Confusion Matrix (Sumber: https://rahulmishra-40030.medium.com)

Ada empat kategori utama di dalamnya: *True Positive* (TP), yaitu ketika model berhasil memprediksi data positif dengan benar; *True Negative* (TN), ketika model tepat mengidentifikasi data negatif; *False Positive* (FP) atau *Type I Error*, yaitu saat model salah memprediksi data negatif sebagai positif; dan *False Negative* (FN) atau *Type II Error*, ketika data positif diprediksi sebagai negatif. *Matrix Evaluation* seperti *Specificity* berguna untuk mengukur kemampuan model dalam mendekteksi kelas negatif (*True Negative*) dengan rumus yang digunakan adalah sebagai berikut:

$$\frac{TP}{TN + FP}$$

Precision berguna untuk mengukur proporsi prediksi positif yang benar dengan rumus yang digunakan adalah sebagai berikut:

$$\frac{TP}{TP + FP}$$

Sensitivity berguna untuk mengukur kemampuan model dalam mendeksi kelas positif (*True Positive*) dengan rumus yang digunakan adalah sebagai berikut:

$$\frac{TN}{TP + FN}$$

Negative Predictive Value berguna untuk mengukur proporsi prediksi negatif yang benar dengan rumus yang digunakan adalah sebagai berikut:

$$\frac{TP}{TP + FP}$$

Lalu untuk *Accuracy* berguna untuk mengukur persentase berapa besar model memprediksi dengan benar, baik yang diprediksi itu positif atau negatif dengan rumus yang digunakan adalah sebagai berikut:

$$\frac{TP + TN}{TP + TN + FP + FN}$$

Dengan mereferensikan nilai dari *Confusion Matrix* kita dapat mencari nilai untuk *Evaluation Matrix* yang akan digunakan untuk mengevaluasi performa model secara lebih detail untuk memperkirakan apakah model yang telah mencapai hasil yang diharapkan.

2.2.12 *Blackbox Testing*

Blackbox Testing adalah sebuah metode yang dipakai untuk menguji sebuah software tanpa harus memperhatikan detail software (Baktiar dkk., 2021:143). Pengujian akan dilakukan pada bagian-bagian proses dalam aplikasi informasi yang dinginkan adalah apakah proses yang diuji berhasil atau tidak. Salah satu jenis pengujian *Blackbox Testing* adalah dengan menggunakan teknik *State Transition Testing* (STD) yang digunakan untuk menguji masukan serta membagi masukan ke dalam kelompok-kelompok berdasarkan fungsinya, sehingga didapatkan sebuah *Test Case* yang akurat (Baktiar dkk., 2021:143). Proses pengujian dilakukan berfokus pada setiap tampilan yang ada pada aplikasi apakah output yang diterima sesuai dengan ekspetasi. Jadi pengujian akan berfokus pada input dan output dari proses aplikasi dan pengujian tidak akan berfokus pada bagaimana proses aplikasi itu berjalan.

2.2.13 *Usability Testing*

Usability testing adalah metode evaluasi rancangan aplikasi atau produk digital yang bertujuan untuk mengukur seberapa mudah pengguna dapat berinteraksi dengan aplikasi tersebut. *Usability testing* merupakan metode yang paling sederhana untuk melihat apa yang terjadi saat pengujian, mudah untuk mendapatkan pengetahuan usability secara nyata, dan sangat murah karena hanya perlu melakukan pengujian kepada sejumlah kecil pengguna (Wibawa, Mursityo, dan Rokhmawati, 2019: 10428). Pengujian ini melibatkan pengamatan langsung terhadap pengguna saat mereka menyelesaikan tugas tertentu menggunakan aplikasi atau prototipe yang sedang diuji, dan pengujian ini akan melihat apakah aplikasi yang dibangun telah memiliki *User Experience* yang tidak sulit untuk digunakan, hal ini berguna untuk meminimalisir hilangnya pemakaian oleh pengguna. Sehingga dapat disimpulkan bahwa dengan menggunakan pengujian *Usability Testing* kita dapat memberikan kenyamanan dan pengalaman yang baik bagi pengguna saat menggunakan aplikasi.