Pengembangan *Back end* Antarmuka Pengguna Untuk Sistem *Counter* *Push-up* Dan *Sit-up* Berbasis Sensor

1st Sabila Hayyinun Jannah
Fakultas Teknik Elektro
Universitas Telkom
Bandung, Indonesia
sabilahayyinunj@student.telkomuniversity.id

2nd Inung Wijayanto
Fakultas Teknik Elektro
Universitas Telkom
Bandung, Indonesia
iwijayanto@telkomuniversity.ac.id

3rd Sugondo Hadiyoso
Fakultas Teknik Elektro
Universitas Telkom
Bandung, Indonesia
sugondo@telkomuniversity.ac.id

***Abstrak* — Dalam tes fisik seperti Tes Rekrutmen Polri, dibutuhkan sistem pendeteksi gerakan *push-up* dan *sit-up* yang akurat dan komprehensif. Tujuan dari penelitian ini adalah untuk membuat sistem berbasis sensor IMU yang terintegrasi dengan *front end* yang dibangun menggunakan HTML, CSS, dan JavaScript dan *back end* yang didukung oleh Django. Dua ESP32 *Client* dengan sensor IMU untuk mendeteksi gerakan membentuk sistem, sementara satu ESP32 Server digunakan untuk mengumpulkan data, menilai apakah gerakan tersebut benar, dan meneruskannya ke Django untuk diproses lebih lanjut. Empat responden menjalani pengujian menggunakan skenario pengujian yang sesuai dengan standar Polri. Hasil pengujian menunjukkan bahwa, meskipun terdapat tantangan tertentu berupa *delay* yang disebabkan oleh jarak komunikasi BLE, sistem ini mampu mendeteksi dan menampilkan hasil gerakan secara *real-time* melalui antarmuka pengguna dengan integrasi yang baik antara perangkat, *back end*, dan *front end*. Kesimpulan dari penelitian ini menunjukkan bahwa sistem yang dirancang dapat berfungsi secara efektif, namun disarankan agar metode komunikasi ditingkatkan untuk jangkauan yang lebih luas.**

***Kata kunci*— sensor IMU, django, ESP32, *push-up*, *sit-up*, ble.**

I. PENDAHULUAN

Rekrutmen untuk Kepolisian Republik Indonesia (Polri) sangat ketat. Serangkaian tes kualifikasi fisik, mental, dan intelektual dilakukan selama tes rekrutmen ini. Tahapan penting adalah pemeriksaan kesehatan menyeluruh, yang mencakup pemeriksaan fisik dan psikologis [1]. Tes fisik atau yang bisa disebut sebagai Tes Kesamaptaan Jasmani, seperti tes *push-up* dan *sit-up*, mengukur kekuatan, ketahanan, dan kebugaran calon anggota, yang menunjukkan kemampuan fisik yang diperlukan untuk melakukan tugas militer dan polisi [2]. Polri memilih anggota terbaik melalui proses rekrutmen yang ketat untuk memastikan bahwa calon anggota memiliki keahlian, keberanian, dan komitmen untuk menjaga kedaulatan negara dan melindungi masyarakat dari berbagai ancaman dan tantangan.

Pada tahun 2023, 11.531 orang telah mendaftar sebagai calon anggota Polri [3]. Jika Tes Kesamaptaan Jasmani dilakukan secara manual dan dikombinasikan dengan jumlah calon yang tinggi, menghasilkan proses yang memakan waktu dan sangat melelahkan, baik bagi petugas yang mengawasi maupun peserta yang menjalani tes. Diperlukan pengembangan alat untuk mempermudah tes *push-up* dan *sit-up* yang tidak hanya mempercepat prosedur tes, tetapi juga memberikan hasil yang lebih akurat. Oleh karena itu, proses pemilihan calon anggota Polri lebih efektif dan menguntungkan bagi semua pihak.

Kompleksitas masalah ini mencakup penggunaan sensor yang tepat, kemampuan untuk mengenali gerakan *push-up* dan *sit-up* dengan benar, algoritma yang diperlukan untuk sensor tersebut, kalibrasi, akurasi, kebutuhan *real-time*, dan integrasi sensor dengan perangkat lunak. Proses validasi dan penyesuaian dengan standar Polri juga penting.

Aspek teknis, kesejahteraan, keselamatan, kesesuaian dengan standar, dan keberlangsungan adalah bagian dari masalah ini. Aspek teknis mencakup cara data pengukuran dikirim dan diterima, kecepatan, latensi, dan keamanan data. Aspek kesejahteraan juga terkait dengan mengatasi kelelahan peserta dan petugas karena waktu tes seleksi yang lama. Aspek keselamatan terkait dengan keamanan alat agar aman digunakan. Aspek kesesuaian dengan standar penting karena alat harus dapat mengenali gerakan *push-up* dan *sit-up* yang sesuai dengan standar Polri. Aspek keberlangsungan meliputi proses pengembangan, perbaikan dan ketahanan.

Saat ini, terdapat beberapa solusi yang ada, seperti sistem GARJAS berbasis Kinect Xbox, BlazePose, dan sensor Ultrasonik. Namun, penelitian lebih lanjut diperlukan untuk mengetahui seberapa cocok dan efektif masing-masing metode dalam memilih calon anggota Polri.

II. KAJIAN TEORI

A. Django sebagai *Framework Back end*

Back end aplikasi web mengacu pada sisi *server* aplikasi yang bertanggung jawab untuk mengelola data, logika, dan interaksi dengan basis data. Ini mencakup komponen seperti basis data, *server*, dan aplikasi yang bekerja di belakang layar untuk mengirimkan data dan konten ke browser pengguna [4]. Dalam pengembangan sistem ini, Django dipilih untuk membuat *back end*. Django adalah *framework* web yang dibuat dengan bahasa pemrograman Python yang bertujuan untuk menulis situs web [5]. *Framework* adalah sekumpulan *software* yang memudahkan pengembang dan mengatur arsitektur aplikasi. *Framework* dapat disesuaikan untuk berbagai tujuan.

B. Integrasi Alat dengan *Back end*

Bleak adalah library pada Python yang dirancang untuk berinteraksi dengan perangkat *Bluetooth Low Energy* (BLE). Pengembang dapat membaca dan menulis *properties*, *subscribe notifications*, dan mengelola koneksi dengan perangkat BLE menggunakan Bleak [6]. Membuat aplikasi untuk *smart home*, perangkat pemantau kesehatan, dan perangkat *Internet of Things* (IoT) lainnya yang menggunakan BLE untuk komunikasi adalah kasus penggunaan yang umum untuk Bleak.

Mikrokontroler ESP32 memiliki modul BLE yang dapat digunakan untuk terhubung ke peragkat lainnya. Karena itu, Modul Bleak digunakan agar *back end* dapat terhubung ke ESP32 melalui BLE.

C. Komunikasi Antara *Front end* dan *Back end*

Application Programming Interface (API) adalah seperangkat aturan dan protokol yang memungkinkan aplikasi perangkat lunak yang berbeda untuk berkomunikasi satu sama lain. API memungkinkan pengembang untuk mengakses fitur atau informasi tertentu dari program lain tanpa harus memahami cara kerjanya. Aplikasi dapat meminta dan bertukar informasi menggunakan metode dan format data yang ditentukan oleh API. Interaksi ini dapat terjadi melalui jaringan, sehingga API sangat penting untuk aplikasi berbasis web [7].

Salah satu komponen penting dari aplikasi web modern adalah komunikasi *front end* dan *back end* berbasis API. API memudahkan aplikasi *front end* (web atau seluler) untuk berkomunikasi dengan layanan *back end*. Sebagai contoh, aplikasi *front end* dapat mengirim dan menerima data dalam format terstruktur (biasanya JSON) melalui API yang memberdayakan *back end* [8].

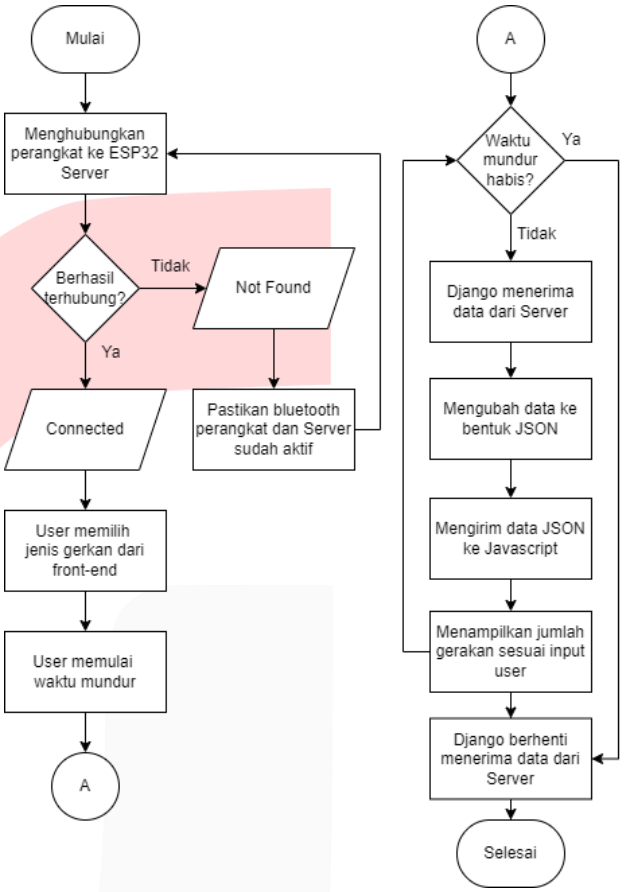
III. METODE

A. Rancangan Sistem

Alat yang dirancang untuk sistem yang dikembangkan dalam proyek ini terdiri dari dua ESP32 *Client* dan satu ESP32 Server. Sensor IMU terpasang pada ESP32 *Client* untuk melacak pergerakan pengguna. Dengan menggunakan koneksi BLE, kedua ESP32 *Client* ini mengomunikasikan data pergerakan ke Server ESP32. Selanjutnya, ESP32 Server memverifikasi apakah gerakan pengguna akurat atau tidak, mengirimkan temuan ke Django, yang berfungsi sebagai *back end*. Data diproses oleh Django dan dikirim ke *front end*, atau antarmuka pengguna, yang dibuat dengan HTML, CSS, dan JavaScript. Hubungan antara ESP32 Server, *back end* dan *front end* digambarkan oleh Gambar 1.

GAMBAR 1.
Diagram hubungan antar komponen

Berdasarkan Gambar 2, pengguna dapat memulai sistem ini dengan membuka halaman beranda aplikasi dan menggunakan pustaka Bleak untuk menghubungkan Django ke ESP32 Server. Setelah koneksi berhasil, pengguna dapat memilih mode latihan, lalu pengguna akan diarahkan ke halaman penghitung di mana perangkat akan mulai menghitung jumlah gerakan yang dilakukan. Pemformatan JSON akan diterapkan pada data yang didapat Django dari ESP32 Server sebelum dikirim ke JavaScript untuk ditampilkan di antarmuka pengguna. Ketika pengguna mengakhiri sesi latihan atau waktu yang dialokasikan telah habis, Django tidak akan lagi mendapatkan data.



GAMBAR 2.
Diagram Alir *Back end*

B. Integrasi Sensor dengan *Back end*

Integrasi antara sensor pada ESP32 dan Django sebagai *back end* dilakukan dengan konfigurasi yang memungkinkan setiap ESP32 *Client* untuk mendeteksi dan mengirimkan data gerakan ke ESP32 Server. Bergantung pada jenis gerakan, pengguna menempatkan ESP32 *Client* pertama pada kaki pada saat *push-up* atau dada pada saat *sit-up*, sedangkan ESP32 *Client* kedua ditempatkan di tangan. Kedua perangkat ini mendeteksi gerakan yang tepat dan memberi tahu ESP32 Server tentang hal itu. Server ESP32 akan memberi tahu Django sebagai '*True*' jika kedua ESP32 *Client* mengidentifikasi gerakan yang tepat. Pemberitahuan ini diterima oleh Django melalui koneksi BLE yang dikelola oleh Bleak, dan dikirim ke antarmuka pengguna sebagai data JSON. Agar Django dapat secara efektif terhubung dan berkomunikasi dengan ESP32 Server dan menerima pemberitahuan yang diperlukan untuk pemrosesan tambahan, perpustakaan Bleak digunakan.

C. Pengolahan Data

Data yang dikirim dari ESP32 ke Django melalui BLE berisi notifikasi sederhana dalam bentuk '*True*' atau '*False*', yang menunjukkan apakah gerakan yang dilakukan oleh pengguna sesuai dengan standar Polri atau tidak. Ketika Django menerima data ini, Django memformatnya ke dalam JSON sehingga JavaScript, yang menangani logika *front end*, dapat menggunakannya. Berdasarkan data yang dikumpulkan, JavaScript kemudian dapat memperbarui antarmuka pengguna untuk menunjukkan jumlah gerakan yang benar dan salah. Pengguna dapat mengamati hasil latihan mereka secara *real-time* berkat metode ini.

IV. HASIL DAN PEMBAHASAN

A. Hasil Pengujian

1. Skenario Pengujian dan Alur Kerja Sistem

Untuk menguji sistem ini, empat orang responden dari UKM Taekwondo Telkom University berpartisipasi. Para responden diminta untuk melakukan teknik *push-up* dan *sit-up* yang benar seperti yang telah ditentukan oleh Polri. Pada skenario pengujian ini, tubuh responden dipasangi perangkat yang terdiri dari dua buah ESP32 *Client* sesuai dengan mode gerakan yang dipilih. Data gerakan yang dihasilkan oleh responden dikirimkan ke *back end* berbasis Django setelah semua perangkat terhubung dengan benar. Laptop yang menjalankan Django, dan ESP32 *Client* yang mendeteksi gerakan terhubung ke Server ESP32 melalui BLE. Data gerakan kemudian diproses di *back end* dan hasilnya ditampilkan pada antarmuka pengguna di *front end*. Ketika responden menekan tombol 'Berhenti' atau waktu mundur tes berakhir, proses ini akan berakhir. Hasil penghitungan gerakan juga dapat disimpan dalam bentuk *file* CSV.

2. Integrasi Sistem

Untuk menjamin stabilitas koneksi, alat, *back end*, dan *front end* diintegrasikan secara bertahap. Django pada awalnya membuat koneksi BLE dengan Server ESP32. Setelah koneksi dibuat, ESP32 *Client* dinyalakan dan dihubungkan ke Server ESP32. *Endpoint* API Django yang berbeda dikembangkan selama pengujian untuk memfasilitasi komunikasi *front end* dan *back end*. *Endpoint* ini dapat digunakan untuk menyimpan dan menampilkan data riwayat gerakan, mendapatkan data dari perangkat, dan menghubungkan perangkat BLE. Dengan koneksi ini, data yang diterima oleh *back end* ditransfer ke *front end* untuk ditampilkan secara *real-time*, sehingga memungkinkan sistem berfungsi sebagai satu kesatuan.

GAMBAR 3.
Tampilan halaman perhitungan *push-up*

Jika sistem dapat terintegrasi dengan baik, maka hasil perhitungan gerakan benar dan salah dapat ditampilkan pada antarmuka pengguna seperti pada Gambar 3 secara *real-time*.

3. Stabilitas Koneksi

Alat ESP32, *back end* Django, dan *front end* semuanya memiliki komunikasi yang baik selama pengujian, yang memungkinkan skor gerakan ditampilkan pada antarmuka pengguna secara *real-time*. Tetapi ketika jarak antara alat dan laptop yang menjalankan *back end* terlalu jauh, ada masalah dengan *delay* yang lebih lama atau kehilangan data. Keterbatasan ini menunjukkan bahwa BLE memiliki batas jangkauan yang mungkin berdampak pada kinerja sistem. Namun demikian, sistem dapat beroperasi dengan benar, menampilkan data dengan tepat, dan menyimpan hasil perhitungan tanpa masalah dalam kondisi normal.

B. Analisis Hasil Pengujian

Integrasi antara alat, *back end*, dan *front end* menunjukkan bahwa sistem dapat berfungsi secara terpadu. Keberhasilan membangun koneksi BLE antara Django dan ESP32 Server memungkinkan penerimaan dan pemrosesan data pergerakan secara *real-time*. Urutan sistem, yang dimulai dengan inisialisasi koneksi dan diakhiri dengan penyimpanan data dalam *file* CSV, menunjukkan betapa pentingnya peran setiap komponen dalam keberhasilan integrasi. Namun demikian, beberapa kesulitan teknis, seperti persyaratan untuk memastikan Django terhubung ke ESP32 Server sebelum menyalakan ESP32 *Client*, menekankan betapa pentingnya melakukan proses integrasi dalam urutan yang benar untuk menghindari kesalahan.

Meskipun ada masalah dengan cakupan BLE, secara keseluruhan stabilitas koneksi perangkat, *back end*, dan *front end* dapat berjalan dengan baik Hal ini memengaruhi daya tanggap sistem, terutama ketika ada jarak yang terlalu jauh antara laptop yang menjalankan Django dan Server ESP32, yang menyebabkan *delay* atau kehilangan data. Namun, sistem ini menunjukkan akurasi yang memadai dalam menampilkan hasil temuan gerakan secara *real-time* saat beroperasi dalam kondisi ideal. Analisis ini menunjukkan bahwa BLE memiliki batasan jarak yang dapat memengaruhi kinerja. Metode komunikasi yang lebih andal diperlukan untuk meningkatkan jangkauan dan ketergantungan sistem.

Salah satu saran utama yang dibuat berdasarkan hasil pengujian adalah untuk memikirkan tentang pemanfaatan protokol komunikasi alternatif, seperti Wi-Fi atau Bluetooth Serial, yang dapat memberikan cakupan dan stabilitas yang lebih baik daripada BLE. Diharapkan bahwa peningkatan ini akan mengurangi masalah *delay* dan meningkatkan kinerja sistem secara keseluruhan.

V. KESIMPULAN

Penelitian ini berhasil mengembangkan sistem berbasis sensor IMU yang mampu mendeteksi gerakan *push-up* dan *sit-up* dengan akurasi memadai. Hasil perhitungan *real-time* ditampilkan melalui antarmuka yang menghubungkan alat, *back end*, dan *front end*. Sistem ini bekerja secara efektif dalam mengidentifikasi gerakan benar dan menampilkan hasil secara *real-time*, berdasarkan uji coba yang dilakukan terhadap empat orang responden dari UKM Taekwondo Telkom University. Namun, ada keterbatasan berupa *delay* yang disebabkan oleh jarak antara laptop dan perangkat BLE. Integrasi antara ESP32 *Client* dan ESP32 Server melalui koneksi BLE berjalan stabil, dengan Django sebagai *back end* yang memungkinkan pengelolaan data dan komunikasi yang

efisien antara alat dan antarmuka pengguna. Fungsi-fungsi penting seperti menghubungkan perangkat BLE, mendapatkan data, dan menyimpan hasil perhitungan ke dalam *file* CSV didukung oleh API *endpoint* yang dibuat. Terlepas dari kinerja sistem yang mengesankan dan kemampuannya untuk menghitung gerakan *push-up* dan *sit-up*, disarankan untuk mempertimbangkan metode komunikasi yang lebih andal dan lebih luas, seperti Wi-Fi atau Bluetooth Serial, untuk mengatasi keterbatasan terkait konektivitas dan jarak. Diharapkan bahwa keseluruhan temuan dari penelitian ini akan membentuk fondasi untuk menciptakan sistem yang lebih baik dan lebih efektif di masa depan yang serupa.

REFERENSI

[1] A. Laksitarin, "Tahapan Tes Bintara Polri 2023." Diakses: 16 Oktober 2023. [Daring]. Tersedia pada: https://casispolri.id/tahapan-tes-bintara-polri-2023/

[2] A. Laksitarin, "Penilaian Tes Kesamaptaan Jasmani POLRI," casispolri.id. Diakses: 16 Oktober 2023. [Daring]. Tersedia pada: https://casispolri.id/penilaian-tes-kesamaptaan-jasmani-polri/

[3] "PENERIMAAN BINTARA POLRI GELOMBANG II TAHUN ANGGARAN 2023," Apr 2023.

[4] D. Rander, P. Dani, D. Panjwani, dan D. Ingle, "BackGen—Backend Generator," 2023, hlm. 373–380. doi: 10.1007/978-981-99-6568-7_34.

[5] S. Dauzon, A. Bendoraitis, dan A. Ravindran, Django: Web Development with Python. Birmingham: Packt Publishing Ltd, 2016.

[6] G. S. Singh dan L. Acerbi, "PyBADS: Fast and robust black-box optimization in Python," Jun 2023, [Daring]. Tersedia pada: http://arxiv.org/abs/2306.15576

[7] O. O. Efuntade dan A. O. Efuntade, "Application Programming Interface (API) And Management of Web-Based Accounting Information System (AIS): Security of Transaction Processing System, General Ledger and Financial Reporting System," Journal of Accounting and Financial Management, vol. 9, no. 6, hlm. 1–18, Sep 2023, doi: 10.56201/jafm.v9.no6.2023.pg1.18.

[8] A. Lilleaas, "Building API-Based Back Ends," dalam Pro Kotlin Web Apps from Scratch, Berkeley, CA: Apress, 2023, hlm. 197–212. doi: 10.1007/978-1-4842-9057-6_10.