Pengembangan Aplikasi Web Pengisian Daya Kendaraan Listrik Berbasis *Open Charge Point Protocol* (OCPP)

1st Hermanus Hasta Wicaksana
Fakultas Ilmu Terapan
Universitas Telkom
Bandung, Indonesia
hermanushasta@student.telkomuniversity.ac.id

2nd Amir Hasanudin Fauzi, S.T., M.T.
Fakultas Ilmu Terapan
Universitas Telkom
Bandung, Indonesia
amirhf@telkomuniversity.ac.id

3rd Rizza Indah Mega Mandasari, S.Kom., M.T.
Fakultas Ilmu Terapan
Universitas Telkom
Bandung, Indonesia
rizzamandasari@telkomuniversity.ac.id

***Abstrak* — Laporan ini menyajikan pengembangan aplikasi web untuk pengisian daya kendaraan listrik berbasis *Open Charge Point Protocol* (OCPP). Modul ini meggunakan protokol OCPP 1.6. Aplikasi ini bertujuan untuk memudahkan dalam mengontrol, memonitoring, dan menganalisa data pengisian daya kendaraan listrik di Universitas Telkom. Pengembangan server lokal diusulkan untuk mengurangi ketergantungan terhadap server eksternal dan meningkatkan keamanan dan stabilitas. Aplikasi ini dirancang untuk memberikan kontrol penuh, pemantauan, dan analisis data pengisian daya kendaraan listrik, dan untuk meminimalkan risiko keamanan yang terkait dengan penggunaan *server* eksternal. Hasil penelitian ini menunjukkan bahwa aplikasi yang dikembangkan dapat secara efektif memfasilitasi pengelolaan data pengisian daya kendaraan listrik dan meningkatkan keamanan dan stabilitas sistem.**

Kata kunci— back-end, charging point management, electric vehicle*, OCPP, SPKLU, *server.

I. PENDAHULUAN

Penggunaan bahan bakar fosil telah menyebabkan ketergantungan energi dan krisis energi global. Di Indonesia, pemerintah mendorong pembangunan Stasiun Pengisian Kendaraan Listrik Umum (SPKLU) serta menetapkan Peraturan Presiden Nomor 5 Tahun 2019 tentang "Percepatan Program Kendaraan Bermotor Listrik Berbasis Baterai (KBLBB)" untuk mendukung mobilitas ramah lingkungan dan mengurangi emisi gas rumah kaca [1]. Pada akhir tahun 2023, jumlah kendaraan listrik di Indonesia mencapai lebih dari 108.000 unit, dan PLN telah membangun 1.124 unit SPKLU di seluruh Indonesia [2]. Pertumbuhan ini diperkirakan akan terus meningkat di tahun-tahun berikutnya.

Institusi, instansi, dan universitas turut berupaya membangun SPKLU, termasuk Telkom University. Namun, SPKLU di Telkom University masih bergantung pada server eksternal untuk kontrol dan monitoring, yang menimbulkan beberapa masalah terkait kontrol yang terbatas dan kurang optimal.

Penelitian ini bertujuan untuk merancang dan mengembangkan server *back-end* berbasis *Open Charge Point Protocol* (OCPP) sebagai solusi untuk mengatasi masalah ini. OCPP adalah protokol terbuka standar untuk komunikasi antara *Charge Point* dan sistem pusat yang dapat mengakomodasi berbagai teknik pengisian daya [3]. Implementasi server lokal berbasis OCPP diharapkan dapat memberikan kontrol penuh atas stasiun pengisian daya, termasuk monitoring, manajemen pengguna, dan analisis data penggunaan. Keamanan dan privasi data pengguna juga akan ditingkatkan melalui langkah-langkah keamanan standar industri. Penelitian ini juga dapat menjadi panduan bagi institusi lain dalam mengembangkan *server* kendaraan listrik yang efisien dan berkelanjutan.

II. KAJIAN TEORI

A. *Electric Vehicle* (EV)

Electric Vehicle (EV) adalah kendaraan yang menggunakan motor listrik sebagai sumber tenaga utama untuk beroperasi, menggantikan mesin pembakaran dalam kendaraan konvensional yang menggunakan bahan bakar fosil seperti bensin atau diesel.

Terdapat empat jenis EV, yaitu *Plug-in Hybrid Electric Vehicle* (PHEV), *Hybrid Electric Vehicle* (HEV), *Fuel Cell Electric Vehicle* (FCEV), dan KBLBB atau *Battery Electric Vehicle* (BEV) [6]. Berikut penjelasan masing-masing jenis Electric Vehicle.

B. *Hypertext Transfer Protocol* (HTPP)

Hypertext Transfer Protocol (HTTP) adalah protokol antara *client* dan *server* bisa saling berkomunikasi dengan melakukan sistem *request-response*. Biasanya HTTP digunakan untuk mengirimkan dokumen yang berbentuk *hypermedia* seperti video, gambar, audio, HTML. Protokol ini dirancang untuk melakukan komunikasi antara web *browser* dengan web *server,* selain itu juga dapat digunakan untuk tujuan lain [9].

HTTP mengikuti model klien-*server* klasik, dengan klien membuka koneksi untuk membuat permintaan, kemudian menunggu sampai menerima *respons*. HTTP adalah protokol tanpa status, yang berarti bahwa *server* tidak menyimpan data (status) apapun di antara dua permintaan [9].

C. *Websocket*

Socket memungkinkan komunikasi antar komputer, berguna dalam pemrograman berbasis *client*-server. API *WebSocket* memungkinkan komunikasi dua arah antara klien dan server melalui web. Antarmuka *WebSocket* menentukan metode dan interaksi klien dengan jaringan, dimulai dengan pemanggilan konstruktor *WebSocket* yang mengembalikan *instance* objek *WebSocket*, mengatur kapan koneksi dibuka, pesan tiba, koneksi ditutup, dan pemberitahuan kesalahan [10].

D. *Open Charge Point Protocol (*OCPP)

Open Charge Point Protocol (OCPP) adalah protokol terbuka standar untuk komunikasi antara *charge point* dan sistem pusat, dirancang untuk mengakomodasi berbagai teknik pengisian daya.

Beberapa fitur inti OCPP 1.6 mencakup otentikasi pengguna (*Authorize*), notifikasi *boot* (*BootNotification*), perubahan ketersediaan (*ChangeAvailability*), perubahan konfigurasi (*ChangeConfiguration*), penghapusan *cache* (*ClearCache*), transfer data tambahan (*DataTransfer*), pengambilan konfigurasi (*GetConfiguration*), notifikasi status (Heartbeat), pengiriman nilai meter (MeterValues), memulai dan menghentikan pengisian daya jarak jauh (*RemoteStartTransaction* dan *RemoteStopTransaction*), inisialisasi ulang (Reset), memulai dan menghentikan pengisian daya *(StartTransaction* dan *StopTransaction*), notifikasi status (*StatusNotification*), dan membuka pengunci konektor (*UnlockConnector*) [3].

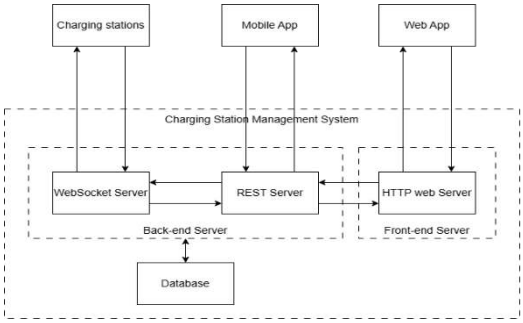
III. METODE

Tahap persiapan dimulai dengan memberikan pemahaman kepada peserta magang tentang arsitektur Stasiun Pengisian Kendaraan Listrik Umum (SPKLU) dan protokol *Open Charge Point Protocol* (OCPP), termasuk fitur-fitur intinya. Peserta akan mengikuti pelatihan dan melakukan penelitian terkait implementasi OCPP dalam sistem pengisian daya kendaraan listrik.

Tahap pengembangan melibatkan pembuatan *back-end* manajemen SPKLU dan implementasi fitur-fitur inti OCPP seperti *Authorize, BootNotification, ChangeAvailability, ChangeConfiguration, ClearCache, DataTransfer, GetConfiguration, Heartbeat, MeterValues, RemoteStartTransaction, RemoteStopTransaction, Reset, StartTransaction, StatusNotification*, dan *UnlockConnector*. Peserta juga akan membangun *server* lokal yang mengontrol dan memonitor stasiun pengisian daya sesuai standar OCPP.

Tahap pengujian dan *debugging* dilakukan untuk memastikan sistem berfungsi optimal. Setiap fitur yang diimplementasikan akan diuji, dan *debugging* dilakukan untuk memperbaiki kesalahan. Peserta magang akan melaporkan perkembangan secara berkala kepada pembimbing lapangan, yang akan memberikan masukan untuk perbaikan. Proyek dianggap selesai setelah semua fitur diuji dan sistem berjalan optimal, memberikan kontrol penuh atas stasiun pengisian daya, termasuk monitoring, manajemen pengguna, dan analisis data penggunaan.

A. Gambaran Sistem Saat Ini

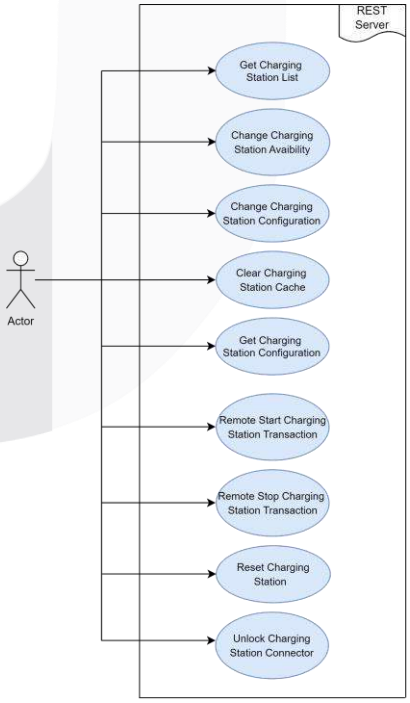


GAMBAR 1
GAMBARAN UMUM SISTEM MANAJEMEN SPKLU

Sistem manajemen stasiun pengisian kendaraan listrik umum memiliki tiga jenis klien. Yaitu, SPKLU (*charging station*), aplikasi *mobile*, dan aplikasi *website*. Aplikasi *mobile* diperuntukkan untuk pengendara kendaraan listrik sehingga pengendara bisa mengakses 12 fasilitas SPKLU. Aplikasi web diperuntukkan untuk operator SPKLU sehingga dapat mengelola dan memantau ataupun mengubah konfigurasi SPKLU.

B. Pengembangan Sistem

Dalam rancangan aplikasi *back-end* sistem manajemen SPKLU, terdapat diagram *use case* yang menggambarkan interaksi antara aplikasi web dengan *server back-end* interaksi ini akan diimplementasikan sebagai fitur REST API sehingga sistem lain dapat mengakses *server back-end*. Diagram *use case* mendeskripsikan fungsionalitas yang ditawarkan oleh sistem digambarkan pada gambar 2.



GAMBAR 2
USE CASE REST API SERVER

IV. HASIL DAN PEMBAHASAN

Telah dilakukan pengujian fungsionalitas pada aplikasi *Postman* dan seluruh fitur aplikasi dipastikan sudah berjalan dengan baik. Didapatkan hasil sebagai berikut.

TABEL 1
PENGUJIAN *AUTHORIZE*

Deskripsi	**Pengujian fungsionalitas *Authorize* OCPP**

Parameter yang dimasukkan	{
	"idTag": "*{{IdTag}}*"
	}

Kriteria Keberhasilan	Server merespon dengan *messageTypeId* 2 serta *payload* "idTagInfo" yang merupakan sebuah objek dengan properti: a. status: string enum (*Accepted*, *Blocked*, *Expired*, *Invalid*, *ConcurrentTx*). b. *expiryDate* (opsional): *date-time* c. *parentIdTag* (opsional): *string*

Hasil Pengujian	[
	"idTagInfo": {
	"Status": "Expired"
	}
	]

TABEL 2
PENGUJIAN *BOOTNOTIFICATION*

Deskripsi	**Pengujian fungsionalitas *BootNotification* OCPP**

Parameter yang dimasukkan	{
	"chargePointVendor": "VendorA",
	"chargePointModel": "SingleSocketCharger"
	}

Kriteria Keberhasilan	Server merespon dengan *messageTypeId* 3 serta *payload* yang memiliki properti: a. *status*: *string enum* (*Accepted*, *Rejected*, *Pending*) b. *currentTime*: *date-time* c. *interval*: *integer*

Hasil Pengujian	{
	"status": "Accepted",
	"messageId": "2024-06-01T06:42:51.810Z",
	"data": 60
	}

TABEL 3
PENGUJIAN *DATATRANSFER*

Deskripsi	**Pengujian fungsionalitas *DataTransfer* OCPP**

Parameter yang dimasukkan	{
	"vendorId": "VendorA",
	"messageId": "customMsg",
	"data": "Some data"
	}

Kriteria Keberhasilan	Server merespon dengan *messageTypeId* 3 serta *payload* yang memiliki properti: status: *string enum* (*Accepted*, *Rejected*, *UnknownMessageId*, *UnknownVendorId*)

Hasil Pengujian	{
	"status": "Accepted"
	}

TABEL 4
PENGUJIAN *HEARTBEAT*

Deskripsi	**Pengujian fungsionalitas *HeartBeat* OCPP**
Parameter yang dimasukkan	Pengujian tidak perlu parameter tambahan sehingga objek kosong atau (`{}`).
Kriteria Keberhasilan	Server merespon dengan *messageTypeId* 3 serta *payload* yang memiliki properti: status: *string enum* (*Accepted*, *Rejected*, *UnknownMessageId*, *UnknownVendorId*)

Hasil Pengujian	{
	"currentTime": "2024-07-17T21:14:38.437Z"
	}

TABEL 5
PENGUJIAN *METERVALUES*

Deskripsi	**Pengujian fungsionalitas *MeterValues* OCPP**

Parameter yang dimasukkan	{
	"connectorId": 1,
	"meterValue": [
	{
	"timestamp": "2024-03-20T03:45:46.918Z",
	"sampledValue":[
	{
	"value": "ABCDEFGHIJKLMNOPQRSTUVWX",
	"context": "Transaction.Begin",
	"format": "Raw",
	"measurand": "Power.Active.Export",
	"phase": "L2",
	"location": "Body",
	"unit": "Percent"
	}
	]
	}
	]
	}

Kriteria Keberhasilan	Server merespon dengan *messageTypeId* 3 serta *payload* yang kosong.
Hasil Pengujian	Pengujian berhasil dan *response* objek kosong "{}"

TABEL 6
PENGUJIAN *STARTTRANSACTION*

Deskripsi	**Pengujian fungsionalitas *StartTransaction* OCPP**
Parameter yang dimasukkan	{ "connectorId": 1,

	"idTag": "{{IdTag}}", "meterStart": 0, "timestamp":"{{$isoTimestamp}}" }
Kriteria Keberhasilan	Server merespon dengan *messageTypeId* 3 serta *payload* yang memiliki properti: 1. *transactionId*: *integer* 2. *idTagInfo*: *object* a. status: *string enum* (*Accepted*, *Blocked*, *Expired*, *Invalid*, *ConcurrentTx*). b. *expiryDate* (opsional): *date-time* c. *parentIdTag* (opsional): *string*

Hasil Pengujian	{
	"transactionId": 3,
	"idTagInfo": {
	"status": "Accepted"
	}
	}

TABEL 7
PENGUJIAN *STATUSNOTIFICATION*

Deskripsi	**Pengujian fungsionalitas *StatusNotification* OCPP**

Parameter yang dimasukkan	{
	"connectorId": 1,
	"errorCode": "NoError",
	"status": "Available"
	}

Kriteria Keberhasilan	Server merespon dengan *messageTypeId* 3 serta *payload* yang kosong.
Hasil Pengujian	Pengujian berhasil dan *response* objek kosong "{}"

TABEL 8
PENGUJIAN *STOPTRANSACTION*

Deskripsi	**Pengujian fungsionalitas *StopTransaction* OCPP**

Parameter yang dimasukkan	{
	"idTag": "{{IdTag}}",
	"timestamp": "{{$isoTimestamp}}",
	"meterStop": 25,
	"transactionId": {{transactionId}}
	}

Kriteria Keberhasilan	Server merespon dengan *messageTypeId* 3 serta *payload* yang memiliki properti: 1. *transactionId*: *integer* 2. *idTagInfo*: *object* a. status: *string enum* (*Accepted*, *Blocked*, *Expired*, *Invalid*, *ConcurrentTx*). b. *expiryDate* (opsional): *date-time* c. *parentIdTag* (opsional): *string*.

Hasil Pengujian	{
	"idTagInfo": {
	"status": "Accepted"
	}
	}

TABEL 9
PENGUJIAN *REST API GET CHARGING STATION CLIENTS*

Deskripsi	**Pengujian fungsionalitas REST API *get charging station clients***
Parameter yang dimasukkan	URL http://localhost:3000/api/v1/central-system/clients dengan *method* GET.
Kriteria Keberhasilan	Server merespon dengan status "200 Ok" dan membawa data *json* berupa daftar ID *charging station* yang sedang terhubung dengan server.
Hasil Pengujian	["123"]

TABEL 10
PENGUJIAN *REST API CHANGE AVAIBILITY*

Deskripsi	**Pengujian fungsionalitas REST API *change availability***

Parameter yang dimasukkan	URL http://localhost:3000/api/v1/central-system/clients/123/change availability menggunakan *method* POST dengan *body*:
	{
	"connectorId": 1,
	"type": "Inoperative"
	}

Kriteria Keberhasilan	Server merespon dengan status "200 Ok" dan membawa data *json* dengan properti status keberhasilan operasi (*Accepted*, *Rejected*, *Scheduled*).

Hasil Pengujian	{
	"status": "Accepted"
	}

TABEL 11
PENGUJIAN *REST API CHANGE CONFIGURATION*

Deskripsi	**Pengujian fungsionalitas REST API *change configuration***

Parameter yang dimasukkan	URL http://localhost:3000/api/v1/central-system/clients/123/change configuration menggunakan *method* POST dengan *body*:
	{
	"key": "HertbeatInterval",
	"value": "300",
	}

Kriteria Keberhasilan	Server merespon dengan status "200 Ok" dan membawa data *json* dengan properti status keberhasilan operasi (*Accepted*, *Rejected*, *Scheduled*).

Hasil Pengujian	{
	"Status": "Accepted"
	}

TABEL 12
PENGUJIAN *REST API CLEAR CACHE*

Deskripsi	**Pengujian fungsionalitas REST API *clear cache***
Parameter yang dimasukkan	URL http://localhost:3000/api/v1/central-system/clients/123/clear cache menggunakan *method* POST.
Kriteria Keberhasilan	Server merespon dengan status "200 Ok" dan membawa data *json* dengan properti status keberhasilan operasi (*Accepted*, *Rejected*).
Hasil Pengujian	{ "Status": "Accepted" }

TABEL 13
PENGUJIAN *REST API GET CONFIGURATION*

Deskripsi	**Pengujian fungsionalitas REST API *get configuration***
Parameter yang dimasukkan	URL http://localhost:3000/api/v1/central-system/clients/123/get configuration menggunakan *method* POST dengan *body*: { "key": ["HeartbeatInterval": "Key2"] }
Kriteria Keberhasilan	Server merespon dengan status "200 Ok" dan membawa data *json* dengan properti: 1. *key*: *string* 2. *readonly*: *boolean* 3. *value*: *string*
Hasil Pengujian	{ "configurationKey": [{ "key": "someConfig", "readonly": **true**, "value": "someValue" }] }

TABEL 14
PENGUJIAN *REST API REMOTE START TRANSACTION*

Deskripsi	**Pengujian fungsionalitas REST API *remote start transaction***
Parameter yang dimasukkan	URL http://localhost:3000/api/v1/central-system/clients/123/start transaction menggunakan *method* POST dengan *body*: { "idTag": "12345" }
Kriteria Keberhasilan	Server merespon dengan status "200 Ok" dan membawa data *json* dengan properti status keberhasilan operasi (*Accepted*, *Rejected*).
Hasil Pengujian	{ "Status": "Accepted" }

TABEL 15
PENGUJIAN *REST API REMOTE STOP TRANSACTION*

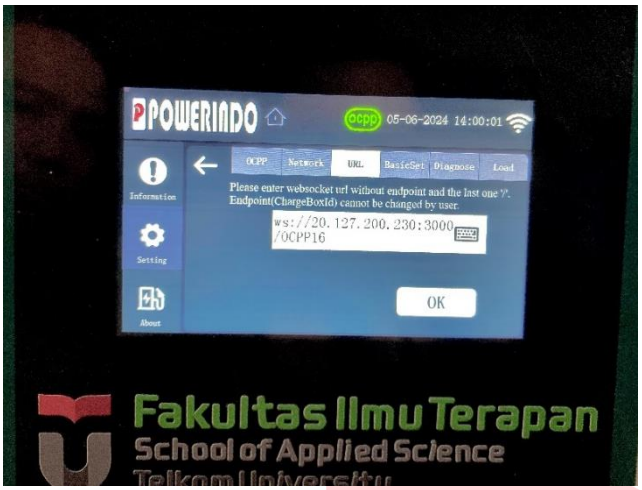
Deskripsi	**Pengujian fungsionalitas REST API *remote stop transcation***
Parameter yang dimasukkan	URL http://localhost:3000/api/v1/central-system/clients/123/stop transaction menggunakan *method* POST dengan *body*: { "transactionId": 1 }
Kriteria Keberhasilan	Server merespon dengan status "200 Ok" dan membawa data *json* dengan properti status keberhasilan operasi (*Accepted*, *Rejected*).
Hasil Pengujian	{ "Status": "Accepted" }

TABEL 16
PENGUJIAN REST API RESET

Deskripsi	**Pengujian fungsionalitas REST API *reset***
Parameter yang dimasukkan	URL http://localhost:3000/api/v1/central-system/clients/123/reset menggunakan *method* POST dengan *body*: { "type": "Soft" }
Kriteria Keberhasilan	Server merespon dengan status "200 Ok" dan membawa data *json* dengan properti status keberhasilan operasi (*Accepted*, *Rejected*).
Hasil Pengujian	{ "Status": "Accepted" }

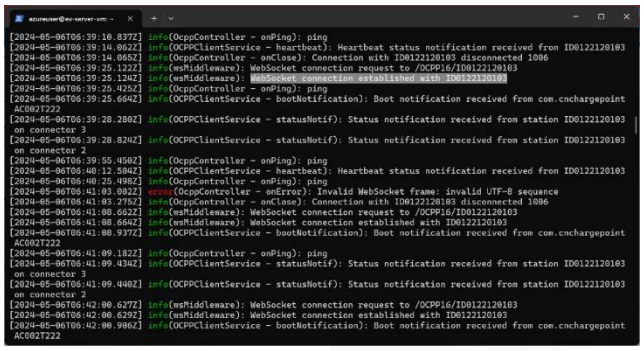
TABEL 17
PENGUJIAN REST *API UNLOCK CONNECTOR*

Deskripsi	**Pengujian fungsionalitas REST API *unlock connector***
Parameter yang dimasukkan	URL http://localhost:3000/api/v1/central-system/clients/123/unlock connector menggunakan *method* POST dengan *body*: { "connectorId": 1 }
Kriteria Keberhasilan	Server merespon dengan status "200 Ok" dan membawa data *json* dengan properti status keberhasilan operasi (*Unlocked*, *UnlockFailed*, *NotSupported*)
Hasil Pengujian	{ "Status": "Unlocked" }



GAMBAR 3
GAMBAR MENU SPKLU

Saat dilakukan pengujian langsung di SPKLU terlihat gambar 3 merupakan menu pengaturan SPKLU yang berada di Telkom University *Landmark Tower.* Ketika melakukan testing dengan menyambungkan program yang sudah di *deploy* ke *Azure Virtual Machine.* Bila program sudah terhubung maka ikon OCPP di layar *monitor* akan berwarna hijau seperti gambar di atas.



GAMBAR 4
LOG SERVER

Tampilan *log server* ketika program sudah terhubung maka akan terlihat "ID0122120103" yang merupakan ID dari SPKLU di Gedung TULT. Terdapat juga *heartbeat status notification* dan juga *Bootnotification* sudah muncul di *log server* dan sudah tersimpan ke *database*.

V. KESIMPULAN

Berdasarkan hasil penelitian dan pengembangan yang telah dilakukan, beberapa kesimpulan dapat diambil. Modul OCPP yang menggunakan Node.js dan *framework* Express.js berhasil dibuat sesuai dengan rencana awal. Modul ini berhasil diuji di Stasiun Pengisian Kendaraan Listrik Umum (SPKLU) di Telkom University *Landmark Tower,* dan dapat berjalan dengan baik. Modul dapat terhubung ke SPKLU dan data dapat tersimpan ke dalam *database* dengan baik. Pengujian menggunakan *Postman* untuk simulasi berhasil, sehingga modul dapat terhubung dengan baik saat diuji langsung di SPKLU.

Selama kegiatan magang di Torsi EV, penulis mendapatkan pengalaman baru yang berharga, termasuk wawasan mengenai cara kerja SPKLU dan protokol OCPP. Penulis juga berhasil mempelajari dan mengembangkan modul berbasis OCPP, mempelajari TypeScript, Node.js, dan *framework* Express.js. Kesimpulan ini menunjukkan bahwa tujuan penelitian telah tercapai. Penulis berhasil membantu dalam pengembangan aplikasi *website* berbasis OCPP yang dapat memfasilitasi pemantauan penggunaan, manajemen pengguna, dan analisis data penggunaan SPKLU. Selain itu, penulis turut membantu dalam mengembangkan struktur server lokal atau internal untuk meningkatkan kontrol atas operasi stasiun pengisian daya. Pengembangan modul ini juga membantu meningkatkan kemandirian dan keamanan sistem pengisian daya kendaraan listrik di Telkom University.

REFERENSI

[1] Presiden Republik Indonesia, "Peraturan Presiden Republik Indonesia Nomor 55 Tahun 2019 Tentang Percepatan Program Kendaraan Bermotor Listrik Berbasis Baterai (Battery Electric Vehicle) Untuk Transportasi Jalan," Jakarta, Aug. 2019.

[2] Gregorius Adi Trianto, "Terus Tingkatkan Jumlah SPKLU Selama 2023, PLN Berhasil Penuhi Kebutuhan Pengguna Kendaraan Listrik di Indonesia," PLN, 24 Februari 2024. [Online]. Available: https://web.pln.co.id/media/siaran-pers/2024/01/terus-tingkatkan-jumlah-spklu-selama-2023-pln-berhasil-penuhi-kebutuhan-pengguna-kendaraan-listrik-di-indonesia

[3] Open Charge Alliance, "Open Charge Point Protocol JSON 1.6, OCPP-J 1.6 Specification," 2015.

[4] Reif, K. (2014). Fundamentals of Automotive and Engine Technology. Germany: Springer.

[5] Crisostomi, E., Shorten, R., Stüdli, S., & Wirth, F. (2018). Electric and Plug-in Hybrid Vehicle Networks: Optimization and Control. Boca Raton, FL: CRC Press.

[6] Autocrpyt. (2021). Your Basic Guide to Electric Vehicle Technology and Trends. Seoul: Autocrpyt.

[7] Kumara, N. S., & Sukerayasa, I. W. (2009). Tinjauan Perkembangan Kendaraan Listrik Dunia Hingga Sekarang. Teknologi Elektro, 8(1), 74–82.

[8] Vindyanandan, K.V. (2018). Overview of Electric and Hybrid Vehicles. A House Journal Of Corporate Planning, 32, 7-14.

[9] Mozilla, "The WebSocket API (WebSockets)," MDN web docs. Accessed: Jan. 13, 2024. [Online]. Available:https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API

[10] Vanessa. Wang, Frank. Salim, and Peter. Moskovits, The definitive guide to HTML5 WebSocket. Apress, 2013.