

**IMPLEMENTASI SISTEM REVERSE VENDING MACHINE
(RVM) BERBASIS IOT MENGGUNAKAN MOBILENET SSD
UNTUK DETEKSI OBJEK DAN MEKANISME INSENTIF
POIN PADA PLATFORM SAMPAHMAS
LAPORAN AKHIR**



Disusun Oleh:

N a m a : Gilang Aryo Pamungkas
NIM : 21523122

**PROGRAM STUDI INFORMATIKA – PROGRAM SARJANA
FAKULTAS TEKNOLOGI INDUSTRI
UNIVERSITAS ISLAM INDONESIA**

2025

HALAMAN PENGESAHAN DOSEN PEMBIMBING

**IMPLEMENTASI SISTEM REVERSE VENDING MACHINE
(RVM) BERBASIS IOT MENGGUNAKAN MOBILENET SSD
UNTUK DETEKSI OBJEK DAN MEKANISME INSENTIF
POIN PADA PLATFORM SAMPAHMAS
TUGAS AKHIR**



Disusun Oleh:

N a m a : Gilang Aryo Pamungkas
NIM : 21523122

Yogyakarta, 16 Januari 2025

Pembimbing,

Andhika Giri Persada, S.Kom, M.Eng.

HALAMAN PENGESAHAN DOSEN PENGUJI

IMPLEMENTASI SISTEM REVERSE VENDING MACHINE (RVM) BERBASIS IOT MENGGUNAKAN MOBILENET SSD UNTUK DETEKSI OBJEK DAN MEKANISME INSENTIF POIN PADA PLATFORM SAMPAHMAS

TUGAS AKHIR

Telah dipertahankan di depan sidang penguji sebagai salah satu syarat untuk memperoleh gelar Sarjana Komputer dari Program Studi Informatika – Program Sarjana di Fakultas Teknologi Industri Universitas Islam Indonesia

Yogyakarta, 16 Januari 2025

Tim Penguji

Andhika Giri Persada, S. Kom., M.Eng.

Anggota 1

Kurniawan Dwi Irianto, S.T., M.Sc.

Anggota 2

Dr. Novi Setiani, S.T., M.T.

Mengetahui,

Ketua Program Studi Informatika – Program Sarjana

Fakultas Teknologi Industri

Universitas Islam Indonesia



Dhomas Hatta Hudholi, S.T., M.Eng., Ph.D

HALAMAN PERNYATAAN KEASLIAN TUGAS AKHIR

Yang bertanda tangan di bawah ini:

Nama : Gilang Aryo Pamungkas

NIM : 21523122

Tugas akhir dengan judul:

Implementasi Sistem Reverse Vending Machine (RVM) Berbasis IoT Menggunakan MobileNet SSD untuk Deteksi Objek dan Mekanisme Insentif Poin pada Platform Sampahmas

Menyatakan bahwa seluruh komponen dan isi dalam tugas akhir ini adalah hasil karya saya sendiri. Apabila di kemudian hari terbukti ada beberapa bagian dari karya ini adalah bukan hasil karya sendiri, tugas akhir yang diajukan sebagai hasil karya sendiri ini siap ditarik kembali dan siap menanggung risiko dan konsekuensi apapun.

Demikian surat pernyataan ini dibuat, semoga dapat dipergunakan sebagaimana mestinya.

Yogyakarta, 16 Januari 2025



Gilang Aryo Pamungkas

HALAMAN PERSEMBAHAN

Segala puji dan syukur saya panjatkan kepada Allah Yang Maha Kuasa atas rahmat dan karunia-Nya, yang telah memberikan saya kesempatan untuk menyelesaikan laporan tugas akhir ini. Laporan ini saya dedikasikan secara khusus kepada semua pihak yang telah mendukung saya, baik secara materiil maupun moril.

Ucapan terima kasih yang sebesar-besarnya saya sampaikan kepada kedua orang tua, sahabat, serta teman-teman yang telah menjadi sistem pendukung selama ini sehingga saya mampu menyelesaikan tanggung jawab ini. Secara khusus, saya juga ingin menyampaikan rasa terima kasih kepada sahabat terbaik saya dan teman-teman Informatika yang selalu memberikan semangat, dukungan, serta kebersamaan yang begitu berarti sepanjang perjalanan ini.

Saya juga ingin menyampaikan rasa terima kasih kepada Universitas Islam Indonesia yang telah menerima saya, mendampingi proses pertumbuhan saya, serta memberikan kesempatan untuk belajar di sini.

Terima kasih saya tujukan pula kepada semua pihak yang telah mengajarkan saya untuk senantiasa bersyukur, memperhatikan detail, berpikir kritis, serta memberikan banyak pelajaran berharga lainnya. Terakhir, laporan ini juga saya persembahkan kepada para pembaca yang ingin memanfaatkannya sebagai referensi dalam penulisan laporan tugas akhir.

HALAMAN MOTO

“With great power comes great responsibility”

Spider-Man

KATA PENGANTAR

Tiada kata yang lebih pantas saya ucapkan selain rasa syukur kepada Allah SWT, atas rahmat dan karunia-Nya sehingga saya dapat menyelesaikan tugas akhir ini yang berjudul "Implementasi Sistem *Reverse Vending Machine* (RVM) Berbasis IoT Menggunakan MobileNet SSD untuk Deteksi Objek dan Mekanisme Insentif Poin pada Platform Sampahmas". Tugas akhir ini disusun sebagai salah satu syarat untuk menyelesaikan studi di Program Studi Informatika, Universitas Islam Indonesia.

Penelitian dan pengembangan bisnis ini memiliki makna yang sangat pribadi bagi saya karena berawal dari keinginan untuk menciptakan solusi nyata bagi kebutuhan masyarakat. Dengan ini, saya berharap dapat memberikan kontribusi, tidak hanya dalam dunia bisnis tetapi juga dalam ilmu pengetahuan yang relevan dengan bidang studi saya. Untuk itu saya menyampaikan banyak terima kasih kepada semua pihak yang telah berkontribusi dalam pembuatan makalah ini.

Saya menyadari bahwa tugas akhir ini masih jauh dari sempurna. Oleh karena itu, kritik dan saran yang membangun dari pembaca sangat saya harapkan untuk penyempurnaan karya ini di masa mendatang. Semoga tugas akhir ini dapat bermanfaat bagi siapa pun yang membacanya dan menjadi inspirasi bagi inovasi selanjutnya.

Yogyakarta, 16 Januari 2025



(Gilang Aryo Pamungkas)

SARI

Pengelolaan sampah secara efisien menjadi tantangan serius terutama dalam hal pengumpulan dan pemilahan sampah. Sampah plastik menjadi salah satu penyumbang terbesar dalam polusi lingkungan. Sebagai langkah untuk mengurangi dampak tersebut, penelitian ini mengembangkan sistem *Reverse Vending Machine* (RVM) berbasis Internet of Things sebagai solusi otomatisasi pengelolaan sampah botol plastik. Sistem dikembangkan dengan mekanisme insentif berupa reward berbasis poin. Pengguna yang memasukkan botol plastik melalui sistem RVM akan memperoleh poin yang dapat dimanfaatkan untuk berbagai keuntungan.

Sistem mengintegrasikan metode autentikasi menggunakan kartu dan QR Code untuk identifikasi pengguna, serta ESP32-CAM dan OpenCV untuk deteksi botol secara *real-time*. Sistem dirancang menggunakan pendekatan Agile yang adaptif terhadap kebutuhan yang muncul selama proses pengembangan. Proses pengembangan dilakukan secara iteratif dengan tiga fase utama yaitu pengembangan fungsionalitas inti, integrasi sistem, dan optimasi performa. Hasil dari penelitian adalah mengimplementasikan mekanisme reward berbasis poin untuk mendorong partisipasi masyarakat dalam pengelolaan sampah terutama botol plastik. Sistem reward diharapkan dapat menjadi insentif yang efektif untuk meningkatkan kesadaran dan partisipasi aktif masyarakat dalam program daur ulang sampah plastik.

Kata kunci: *Reverse Vending Machine*, IoT, MobileNet SSD, Deteksi Objek, ESP32-CAM

GLOSARIUM

Glosarium memuat daftar kata tertentu yang digunakan dalam laporan dan membutuhkan penjelasan, misalnya kata serapan yang belum lazim digunakan. Urutkan sesuai abjad. Contoh penulisannya seperti di bawah ini:

API	sekumpulan protokol untuk membangun aplikasi perangkat lunak yang memungkinkan komunikasi antar program
ESP32	mikrokontroler yang dilengkapi dengan fitur WiFi dan Bluetooth untuk aplikasi Internet of Things (IoT)
ESP32-CAM	varian ESP32 yang dilengkapi dengan modul kamera untuk aplikasi pengambilan gambar
Firebase	platform pengembangan aplikasi mobile dan web yang menyediakan berbagai layanan <i>backend</i> seperti database realtime, autentikasi, dan hosting
Internet of Things	konsep yang menghubungkan perangkat fisik dengan internet untuk memungkinkan pengumpulan dan pertukaran data
MobileNet SSD	model <i>deep learning</i> yang dioptimalkan untuk deteksi objek pada perangkat mobile dengan efisiensi komputasi tinggi
MQTT	protokol messaging ringan untuk komunikasi mesin-ke-mesin (M2M) dalam konteks IoT
OpenCV	library open source untuk computer vision dan machine learning.
RFID	teknologi yang menggunakan gelombang radio untuk mengidentifikasi objek secara otomatis
REST API	arsitektur perangkat lunak untuk sistem terdistribusi yang menggunakan protokol HTTP untuk komunikasi data
Reverse Vending Machine	mesin yang menerima sampah daur ulang dan memberikan insentif kepada pengguna
WebSocket	protokol komunikasi yang menyediakan saluran komunikasi dua arah secara <i>real-time</i> antara client dan server
Computer Vision	bidang ilmu yang mempelajari bagaimana komputer dapat memahami konten visual dari gambar atau video digital
QR Code	kode matriks dua dimensi yang dapat menyimpan informasi dan dibaca menggunakan kamera digital

Node.js	<i>platform runtime</i> Javascript yang memungkinkan eksekusi kode Javascript di sisi server
Flask	framework web mikro untuk Python yang digunakan untuk membangun aplikasi web
Black Box Testing	metode pengujian perangkat lunak yang menguji fungsionalitas tanpa melihat struktur kode internal
Real-time Database	database yang memungkinkan sinkronisasi data secara langsung antara server dan client
GPIO	pin pada mikrokontroler yang dapat diprogram untuk input atau output digital
JWT	standar untuk membuat token akses yang dapat digunakan untuk autentikasi dan pertukaran informasi
Webhook	mekanisme komunikasi otomatis antar aplikasi yang dipicu oleh event tertentu
Express.js	framework web untuk Node.js yang menyediakan fitur-fitur untuk membangun aplikasi web dan API
Middleware	suatu fungsi yang berperan sebagai jembatan antara sistem operasi atau database dan aplikasi
Agile	metodologi pengembangan perangkat lunak yang menekankan pada fleksibilitas dan kolaborasi tim

DAFTAR ISI

HALAMAN JUDUL	i
HALAMAN PENGESAHAN DOSEN PEMBIMBING	ii
HALAMAN PENGESAHAN DOSEN PENGUJI	iii
HALAMAN PERSEMBAHAN	v
HALAMAN MOTO	vi
KATA PENGANTAR	vii
SARI	viii
GLOSARIUM	ix
DAFTAR ISI	xi
DAFTAR TABEL	xiii
DAFTAR GAMBAR	xiv
BAB I PENDAHULUAN	1
1.1 Latar Belakang	1
1.2 Rumusan Masalah	2
1.3 Batasan Masalah	3
1.4 Tujuan	3
1.5 Manfaat	3
1.6 Metodologi Perintisan Bisnis	4
1.7 Sistematika Penulisan	5
BAB II LANDASAN TEORI	6
2.1 Sampahmas	6
2.2 <i>Reserve Vending Machine</i>	6
2.3 Internet of Things	6
2.4 RESTful API	7
2.5 Firebase Realtime Database	7
2.6 Sequence Diagram	8
2.7 ESP-31 CAM	8
2.8 ESP-32	8
2.9 RFID	9
2.10 QR Code	9
2.11 MQTT	9
2.12 WebSocket	9
2.13 Node.js	9
2.14 Express.js	10
2.15 Flask Phyton	10
2.16 Computer Vision	11
2.17 OpenCV	11
2.18 MobileNet SSD	11
2.19 Blackbox Testing	12
BAB III METODOLOGI	13
3.1 Aktivitas Tim Perintisan Bisnis	13
3.2 Pendekatan Lean Startup	15
3.3 Iterasi 1 – Problen – Solution Fit	15
3.3.1 Requirement	17
3.3.2 Design	17
3.3.3 Development	18
3.3.4 Testing	18

3.3.5	Deployment.....	18
3.3.6	Review	18
3.4	Iterasi ke-2-Product-Market Fit	18
3.4.1	Requirement	19
3.4.2	Design	19
3.4.3	Development	20
3.4.4	Testing.....	20
3.4.5	Deployment.....	20
3.4.6	Review	20
3.5	Iterasi ke-3-Scaling & Optimization	20
3.5.1	Requirement	21
3.5.2	Design	22
3.5.3	Development	22
3.5.4	Testing.....	22
3.5.5	Deployment.....	23
3.5.6	Review	23
BAB IV HASIL DAN PEMBAHASAN.....		24
4.1	Iterasi 1.....	24
4.1.1	Requirement	24
4.1.2	Design	24
4.1.3	Development	25
4.1.4	Testing.....	27
4.1.5	Deployment.....	27
4.1.6	Review	27
4.2	Iterasi 2.....	28
4.2.1	Requirement	28
4.2.2	Design	29
4.2.3	Development	41
4.2.4	Testing.....	72
4.2.5	Deployment.....	83
4.2.6	Review	83
4.3	Iterasi 3.....	84
4.3.1	Requirement	84
4.3.2	Design	85
4.3.3	Development	86
4.3.4	Testing.....	90
4.3.5	Deployment.....	91
4.3.6	Review	91
4.4	Hasil Akhir.....	91
4.4.1	Reverse Vending Machine (RVM) Sampahmas	92
4.4.2	Mockup Aplikasi Mobile Sampahmas	93
4.4.3	Hasil Pencapaian & Penghargaan Ide Bisnis Sampahmas.....	93
BAB V KESIMPULAN DAN SARAN		95
5.1	Kesimpulan	95
5.2	Saran.....	96
DAFTAR PUSTAKA.....		97

DAFTAR TABEL

Tabel 3.1 Aktivitas Tim Perintisan Bisnis	13
Tabel 4.1 Hasil pengujian deteksi objek	81
Tabel 4.2 Hasil pengujian deteksi objek	82

DAFTAR GAMBAR

Gambar 3.1 Diagram <i>Lean Startup Methodology</i>	15
Gambar 3.2 Metode Agile.....	16
Gambar 4.1 Desain alat deteksi botol	24
Gambar 4.2 Prototyping alat deteksi.....	25
Gambar 4.3 Desain antarmuka <i>Reverse Vending Machine</i>	28
Gambar 4.4 Desain arsitektur sistem SampahMas	29
Gambar 4.5 Desain sirkuit diagram	30
Gambar 4.6 Sequence diagram autentikasi dengan kartu	31
Gambar 4.7 Sequence diagram autentikasi dengan QR Code	32
Gambar 4.8 Sequence diagram deteksi botol.....	33
Gambar 4.9 Class Diagram Sistem Sampahmas.....	37
Gambar 4.10 Prototyping RVM.....	44
Gambar 4.11 API Daftar Pengguna	73
Gambar 4.12 API Login.....	73
Gambar 4.13 API Logout.....	74
Gambar 4.14 API Profile User.....	75
Gambar 4.15 API Get Product	76
Gambar 4.16 API Get Product by id.....	76
Gambar 4.17 API Get lokasi vending.....	77
Gambar 4.18 Topic Auth	78
Gambar 4.19 Topic Point.....	79
Gambar 4.20 Upload Gambar Deteksi objek botol.....	80
Gambar 4.21 Hasil <i>Reverse Vending Machine</i>	83
Gambar 4.22 Perubahan fungsi dan tampilan RVM	84
Gambar 4.23 Sequence diagram register kartu	85
Gambar 4.24 Pengujian registrasi kartu melalui websocket.....	90
Gambar 4.25 Spesifikasi <i>Reverse Vending Machine</i>	92
Gambar 4.26 Purwarupa Alpha Reverse Vending Machine	92
Gambar 4.27 Mockup Aplikasi Sampahmas	93

BAB I

PENDAHULUAN

1.1 Latar Belakang

Permasalahan lingkungan pengelolaan sampah plastik telah menjadi permasalahan serius di Indonesia. Dengan populasi yang terus meningkat, volume sampah terutama sampah plastik semakin tak terkendali. Menurut (Law et al., 2020), Indonesia merupakan salah satu negara dengan jumlah produksi sampah plastik terbesar di dunia yang sebagian besarnya berakhir dengan mencemari lingkungan khususnya kawasan pesisir dan wilayah perairan.

Keadaan tersebut juga dapat dilihat pada tingkat lokal, sebagaimana yang terjadi di Yogyakarta ketika Tempat Pembuangan Akhir (TPA) Piyungan ditutup pada Juli 2023. Penutupan ini disebabkan oleh kapasitas yang terlampaui batas, dimana TPA yang seharusnya hanya mampu menampung 600 ton sampah, terpaksa mengelola hingga 1.000 ton. Kondisi ini terjadi akibat dari kurangnya efisiensi dalam pengelolaan sampah dan bertambahnya limbah yang tidak tertangani dengan baik (Hasanudin, 2023). Data (Kehutanan, 2023) menyebutkan volume sampah harian di Daerah Istimewa Yogyakarta mencapai 1.506 ton per hari, dengan komposisi terbesar berasal dari sisa makanan (50,56%) dan sampah plastik sebagai penyumbang kedua terbesar (27,94%). Sampah plastik yang tidak dikelola dengan baik dapat merusak ekosistem laut, mengancam kehidupan biota air, dan pada akhirnya berdampak buruk pada kesehatan manusia.

Salah satu tantangan utama dalam pengelolaan sampah plastik adalah kurangnya kesadaran masyarakat dalam memilah dan mendaur ulang sampah. Inisiatif seperti bank sampah telah diusulkan untuk membantu pengelolaan sampah. Akan tetapi partisipasi masyarakat masih rendah, seperti yang terlihat di Kabupaten Sleman, Yogyakarta. Partisipasi pengelolaan sampah di Kabupaten Sleman hanya sekitar 1,65% hingga 1,72% (R. D. Astuti et al., 2023). Situasi diperburuk oleh penutupan tempat pembuangan akhir (TPA) Piyungan pada Juli 2023 yang menandakan adanya ketidaksesuaian antara kebijakan pengelolaan sampah dengan pelaksanaannya di lapangan (Tasya, 2023). Masalah ini menunjukkan bahwa pengelolaan sampah plastik memerlukan pendekatan yang inovatif dan berorientasi pada hasil nyata untuk meningkatkan efektivitas.

Rendahnya partisipasi masyarakat menunjukkan adanya peluang untuk menciptakan model bisnis yang dapat mendorong peningkatan keterlibatan publik dalam daur ulang sampah plastik. Model bisnis tidak hanya fokus pada pengelolaan sampah, tetapi juga pada

pengembangan ekonomi yang menguntungkan berbagai pihak. Untuk mewujudkan model bisnis tersebut, teknologi memainkan peran penting dalam menciptakan solusi yang efektif dan efisien. Salah satu konsep yang muncul dari pemanfaatan teknologi adalah Sampahmas, sebuah inovasi yang dirancang untuk mengatasi masalah pengelolaan sampah anorganik.

Sampahmas menawarkan solusi yaitu aplikasi e-commerce dan Reverse Vending Machine (RVM). Masyarakat dapat menukarkan sampah anorganik yang mereka kumpulkan dan mendapatkan poin sebagai insentif dari aplikasi tersebut. Menurut Struk (2019), sistem insentif ini mencakup mekanisme yang memberikan manfaat langsung kepada individu atau masyarakat, baik dalam bentuk pengurangan biaya maupun insentif finansial. Mekanisme insentif seperti ini terbukti efektif dalam meningkatkan partisipasi masyarakat dalam pemilahan sampah, sekaligus menciptakan sistem pengelolaan sampah yang lebih adil.

Inovasi teknologi tidak hanya mendukung pengelolaan sampah plastik, tetapi juga berfungsi sebagai pendorong utama dalam mengimplementasikan model bisnis yang berfokus pada pengembangan ekonomi sirkular termasuk masyarakat, bisnis, dan pemerintah.

Pengembangan aplikasi Sampahmas didasarkan pada rancangan bisnis oleh hustler dan rancangan pengalaman pengguna oleh hipster. Dalam konteks ini, hustler bertanggung jawab dalam menyusun strategi bisnis yang berfokus pada peningkatan partisipasi masyarakat dalam pengelolaan sampah, sementara hipster merancang bagaimana sistem akan memberikan pengalaman pengguna yang intuitif dan mudah digunakan.

Dalam pengembangan Sampahmas, peran Hacker adalah merancang dan mengembangkan sistem deteksi sampah anorganik khususnya botol plastik yang efektif dan efisien. Sistem terintegrasi dengan Internet of Things (IoT) sehingga sistem dapat mengirim dan menerima data secara jarak jauh secara real-time serta terhubung langsung dengan aplikasi. Menurut (Suarna & Sopyan, 2023), teknologi IoT memiliki peran krusial dalam era saat ini karena kemampuannya untuk menghubungkan berbagai perangkat dan mendukung proses komputasi serta komunikasi secara otomatis. Teknologi ini mengintegrasikan teknologi operasi dan teknologi informasi sebagai jembatan antara dunia fisik dan digital. Selain itu, IoT digunakan pada perangkat untuk mengirimkan dan menerima data melalui jaringan internet secara otomatis tanpa memerlukan keterlibatan langsung dari pengguna, baik dalam interaksi sesama manusia maupun dengan (Suarna & Sopyan, 2023).

1.2 Rumusan Masalah

Berdasarkan pemaparan di latar belakang, diperoleh rumusan masalah yang diangkat pada tugas akhir ini sebagai berikut:

- a. Bagaimana mengembangkan sistem backend untuk aplikasi Sampahmas sesuai dengan rancangan desain yang dihasilkan oleh hipster dan proses bisnis yang dirancang oleh hustler?
- b. Bagaimana mengintegrasikan Sampahmas dengan sistem IoT Reverse Vending Machine untuk mendukung pengelolaan sampah botol plastik secara otomatis?
- c. Bagaimana mengimplementasikan sistem deteksi objek dan validasi botol plastik pada Reverse Vending Machine?

1.3 Batasan Masalah

Untuk memperjelas fokus penulisan, batasan masalah perlu ditetapkan. Berikut merupakan batasan-batasan masalah dalam pengembangan sistem deteksi botol untuk aplikasi Sampahmas:

- a. Bagaimana mengembangkan sistem backend untuk aplikasi Sampahmas sesuai dengan rancangan desain yang dihasilkan oleh hipster dan proses bisnis yang dirancang oleh hustler?
- b. Bagaimana mengintegrasikan Sampahmas dengan sistem IoT Reverse Vending Machine untuk mendukung pengelolaan sampah botol plastik secara otomatis?
- c. Bagaimana mengimplementasikan sistem deteksi objek dan validasi botol plastik pada Reverse Vending Machine?
- d. *Reverse Vending Machine* yang dikembangkan hanya bertujuan untuk mengumpulkan dan mengelola informasi dari sensor dan identifikasi pengguna tanpa mengembangkan fungsi tambahan seperti penyortiran sampah otomatis atau daur ulang.

1.4 Tujuan

Tujuan dari perintisan bisnis Sampahmas ini adalah pengembangan aplikasi yang terintegrasi dengan *Reverse Vending Machine*. Pada laporan ini fokus utama adalah pengembangan sistem deteksi pada *Reverse Vending Machine*, pencatatan dan pengiriman data botol secara *real-time*.

1.5 Manfaat

Manfaat Untuk memperjelas fokus penulisan, batasan masalah perlu ditetapkan. Berikut merupakan batasan-batasan masalah dalam pengembangan sistem deteksi botol untuk aplikasi Sampahmas:

- a. Pengguna mendapatkan insentif berupa poin setiap kali melakukan penukaran botol plastik. Poin yang terkumpul dapat digunakan untuk berbagai keuntungan seperti potongan harga pada produk-produk yang tersedia di aplikasi. Sistem pencatatan poin dilakukan

secara *real-time* sehingga pengguna dapat langsung melihat perubahan jumlah poin mereka

- b. Membuka kesempatan bagi mitra bisnis seperti minimarket, supermarket, atau pusat perbelanjaan untuk mengembangkan program pengelolaan sampah yang terintegrasi dengan teknologi. Dengan menempatkan *Reverse Vending Machine* di lokasi mereka, mitra dapat meningkatkan traffic pengunjung dan berpartisipasi dalam program ramah lingkungan yang memberikan dampak positif bagi lingkungan sekaligus meningkatkan citra bisnis mereka dalam aspek keberlanjutan lingkungan

1.6 Metodologi Perintisan Bisnis

Dalam pengembangan sistem deteksi botol Sampahmas yang terintegrasi dengan *Reverse Vending Machine* pendekatan yang digunakan adalah metode *Lean Startup*. Pendekatan ini berfokus pada pengembangan produk yang berkelanjutan melalui siklus *Build-Measure-Learn* untuk validasi asumsi bisnis secara cepat:

a. Build

Pada tahap ini fokus pada pembuatan *Minimum Viable Product* (MVP) dari sistem Sampahmas. MVP yang dikembangkan berupa prototipe dasar *Reverse Vending Machine* dengan fitur esensial yaitu deteksi botol sederhana dan sistem poin basic. *Prototipe* ini dibuat secepat mungkin untuk bisa segera diuji ke pengguna dengan investasi minimal.

b. Measure

Setelah MVP selesai dibuat, pada tahap *Measure* dilakukan pengukuran utama untuk menilai penerimaan pengguna terhadap solusi yang ditawarkan.

c. Learn

Pada tahap *Learn*, data yang terkumpul dari tahap pengukuran dianalisis untuk mendapatkan insight penting tentang perilaku pengguna dan performa sistem. Pembelajaran ini digunakan untuk validasi atau asumsi bisnis awal, perbaikan akurasi sistem deteksi botol, penyempurnaan tampilan pengguna, dan penambahan fitur berdasarkan kebutuhan nyata pengguna.

Proses ini dilakukan secara berulang dimana setiap siklus menghasilkan versi produk yang lebih baik berdasarkan feedback nyata dari pengguna. Metode ini memfasilitasi pengembangan produk dengan cara yang lebih efisien, menurunkan kemungkinan terjadinya kegagalan karena setiap fitur yang dirancang telah melalui proses validasi langsung dari pengguna

1.7 Sistematika Penulisan

Sistematika penulisan bertujuan untuk memudahkan pemahaman isi dan tujuan dari laporan tugas akhir ini. Berikut adalah sistematika penulisan yang terdiri dari lima bab, yaitu:

BAB 1 Pendahuluan

Bab ini membahas tentang latar belakang pengembangan sistem deteksi botol, rumusan masalah, batasan masalah, tujuan pengembangan, manfaat pengembangan, dan sistematika penulisan yang terkait dengan pengembangan sistem deteksi botol untuk aplikasi Sampahmas yang terintegrasi dengan *Reverse Vending Machine*.

BAB II Landasan Teori

Bab ini membahas teori-teori yang relevan dengan pengembangan sistem deteksi botol berbasis sensor dan *RFID*, serta teori-teori yang mendukung integrasi sistem dengan *Reverse Vending Machine* dan pengiriman data *real-time*.

BAB III Metode Pengembangan

Bab ini menjelaskan hasil pengembangan sistem deteksi botol yang terintegrasi dengan *Reverse Vending Machine*, hasil pengujian *API* dan sistem deteksi serta refleksi atas kegiatan pengembangan yang telah dilakukan.

BAB IV Hasil dan Pembahasan

Bab ini menjelaskan hasil pengembangan sistem deteksi botol yang terintegrasi dengan *Reverse Vending Machine*, hasil pengujian *API* dan sistem deteksi serta refleksi atas kegiatan pengembangan yang telah dilakukan.

BAB V Kesimpulan dan Saran

Bab ini memberikan kesimpulan dari seluruh kegiatan pengembangan sistem deteksi botol pada aplikasi Sampahmas, serta memberikan saran untuk pengembangan lebih lanjut atau perbaikan sistem di masa mendatang.

BAB II

LANDASAN TEORI

2.1 Sampahmas

Sampahmas merupakan sebuah platform *e-commerce* yang terintegrasi dengan *Reverse Vending Machine* dimana pengguna dapat menukarkan sampah anorganiknya menjadi insentif berupa poin yang dapat digunakan untuk potongan harga saat berbelanja *online*. Tujuan adanya Sampahmas adalah untuk meningkatkan kesadaran masyarakat terhadap lingkungan dengan upaya mengubah sudut pandang masyarakat mengenai sampah menjadi sesuatu yang berharga.

2.2 Reserve Vending Machine

Reverse Vending Machine (RVM) merupakan sebuah mesin yang membalikkan prinsip kerja mesin vending pada umumnya. Jika mesin vending konvensional bekerja dengan prinsip konsumen memasukkan uang untuk mendapatkan produk, RVM justru sebaliknya dimana konsumen memasukkan produk tertentu dan mesin akan memberikan imbalan berupa uang atau poin (Handoko et al., 2018). RVM dikembangkan sebagai salah satu solusi untuk mengurangi limbah dengan memberikan insentif kepada masyarakat yang berpartisipasi dalam program daur ulang.

Sistem pada RVM umumnya terdiri dari beberapa komponen utama seperti sensor untuk deteksi objek, mikrokontroler sebagai pemroses utama, dan mekanisme pemberian reward. RVM sering dilengkapi dengan kemampuan untuk mengidentifikasi berbagai jenis dan ukuran botol, serta dapat memberikan reward yang berbeda sesuai dengan karakteristik objek yang dimasukkan. Pengembangan RVM merupakan bagian dari upaya pemanfaatan teknologi untuk mendukung program pengelolaan sampah dan mendorong partisipasi masyarakat dalam kegiatan daur ulang.

2.3 Internet of Things

Internet of Things (IoT) adalah sebuah konsep yang bertujuan untuk memperluas manfaat dari konektivitas internet yang selalu aktif dan terhubung secara terus-menerus (Efendi, 2018). Teknologi ini mendukung integrasi antara dunia fisik dan digital, sehingga setiap perangkat dapat dikendalikan dan dipantau secara *real-time* dari jarak jauh sekaligus. Hal ini tidak hanya meningkatkan efisiensi dan kontrol, tetapi juga memungkinkan analisis data yang lebih mendalam untuk pengambilan keputusan yang lebih cepat dan akurat.

Penggunaan IoT dalam penelitian ini didasarkan pada potensi besar teknologi ini untuk meningkatkan efisiensi dan akurasi dalam pengumpulan serta analisis data. IoT memungkinkan

otomatisasi proses data *real-time* dapat dikumpulkan secara terus-menerus, memberikan gambaran yang lebih lengkap dan akurat mengenai kondisi atau proses tertentu. Oleh karena itu, penggunaan IoT dalam penelitian ini bertujuan untuk memaksimalkan pengumpulan data yang akurat dan responsif.

2.4 RESTful API

RESTful API merupakan salah satu tipe dari arsitektur dalam *Application Programming Interface* (API). RESTful API juga dikenal sebagai layanan web RESTful atau REST API. REST, yang artinya *Representational State Transfer* adalah cara arsitektur dan metode komunikasi yang banyak dipakai dalam pembuatan layanan web. RESTful API memberikan kemampuan bagi berbagai sistem yang berbeda untuk berinteraksi secara efektif (Choirudin & Adil, 2019). Sumber daya yang tersedia di dalam database sebuah sistem dapat dijangkau lewat endpoint API yang ada di REST (Choirudin & Adil, 2019).

RESTful API digunakan untuk proyek ini karena arsitektur tersebut mendukung komunikasi yang efisien dan sederhana antara berbagai sistem yang berbeda. REST API memberikan kemampuan bagi aplikasi untuk berinteraksi dengan sistem lain secara langsung melalui HTTP.

2.5 Firebase Realtime Database

Firebase Realtime Database merupakan produk utama dari *Firebase* untuk menyimpan dan mensinkronisasi data secara *real-time* melalui beberapa *client* yang berbeda. *Firebase* sendiri adalah layanan *Backend as a Service* (BaaS) yang dikembangkan oleh *Google* untuk mempermudah proses pengembangan aplikasi web maupun mobile (Khairun Nisa Meiah Ngafidin et al., 2021).

Sebagai basis data NoSQL, *Firebase Realtime Database* menyimpan data dalam format *JavaScript Object Notation* (JSON) yang memungkinkan sinkronisasi data secara *real-time* ke setiap *client* yang terhubung. Hal ini berarti setiap kali data berubah, semua *client* akan menerima *update* data dengan cepat.

Salah satu keunggulan *Firebase Realtime Database* dibandingkan basis data lainnya adalah kemampuannya untuk menyinkronkan data secara *real-time* dengan latensi yang rendah. Selain itu, *Firebase* juga terintegrasi dengan berbagai layanan lain dalam ekosistem *Firebase* seperti *Firebase Authentication* untuk autentikasi pengguna, *Firebase Hosting* untuk hosting aplikasi web, dan *Firebase Storage* untuk penyimpanan file (Khairun Nisa Meiah Ngafidin et al., 2021). Dengan menggunakan *Firebase*, data yang dikirim akan langsung diperbarui di aplikasi setiap kali nilainya berubah (Khairun Nisa Meiah Ngafidin et al., 2021).

2.6 Sequence Diagram

Sequence diagram merupakan jenis diagram interaksi yang menggambarkan bagaimana proses, objek, dan komponen interaksi antar objek dalam suatu skenario use case termasuk objek yang terlibat dan pesan-pesan yang dikirimkan di antara mereka (Raharjana & Justitia, 2015). Diagram ini fokus pada urutan pesan yang dikirim antar objek dan sangat berguna untuk menggambarkan alur kerja, protokol messaging, dan pola interaksi dalam sistem. Dalam konteks pengembangan perangkat lunak, sequence diagram membantu memvisualisasikan interaksi antara berbagai komponen sistem dalam urutan kronologis.

Penggunaan sequence diagram dalam arsitektur sistem memiliki peran penting dalam mendokumentasikan kebutuhan sistem dan memvalidasi logika arsitektur perangkat lunak. Komponen utama sequence diagram terdiri dari *lifeline* yang merepresentasikan peran atau objek dalam interaksi, *messages* yang menunjukkan komunikasi antar objek, *activation boxes* yang menandakan periode aktif suatu objek, dan *combined fragments* yang menggambarkan aliran kontrol bersyarat.

2.7 ESP-31 CAM

ESP32-CAM merupakan modul pengembangan yang mengintegrasikan kamera dengan mikrokontroler ESP32, menjadikannya solusi kompak untuk aplikasi yang membutuhkan kemampuan pengambilan gambar dan konektivitas nirkabel. Dalam implementasinya pada sistem *Reverse Vending Machine* (RVM), ESP32-CAM berperan penting dalam proses validasi visual objek untuk memastikan bahwa benda yang dimasukkan pengguna sesuai dengan kriteria yang ditentukan.

Perangkat ini menggabungkan kamera OV2640 dengan resolusi tinggi serta prosesor *dual-core* yang mampu melaksanakan algoritma pengolahan gambar secara waktu nyata. Keunggulan dari ESP32-CAM terletak pada kemampuannya untuk menangkap gambar dan mengirimkannya melalui jaringan nirkabel yang menjadikannya pilihan optimal untuk aplikasi *Internet of Things* berbasis visual.

2.8 ESP-32

Mikrokontroler ESP32 merupakan perangkat *System on Chip* (SoC) yang terintegrasi dengan fitur komunikasi nirkabel seperti Wi-Fi 802.11 b/g/n, Bluetooth versi 4.2, dan berbagai periferal lainnya. ESP32 dirancang sebagai chip dengan performa tinggi yang menggabungkan prosesor, penyimpanan, dan akses GPIO (*General Purpose Input Output*). Kemampuan ini menjadikan ESP32 sangat ideal untuk aplikasi *Internet of Things* (IoT) yang membutuhkan koneksi jaringan yang stabil dan efisien. Selain itu, ESP32 dapat digunakan sebagai pengganti

Arduino pada berbagai rangkaian, terutama karena fitur *built-in* Wi-Fi yang memungkinkan perangkat terhubung langsung ke jaringan tanpa tambahan perangkat keras lain (Wagya, 2019).

2.9 RFID

RFID (*Radio Frequency Identification*) adalah teknologi yang memanfaatkan gelombang radio untuk secara otomatis mengidentifikasi dan melacak objek (Yusup, 2022). Sistem bekerja dengan menggunakan tag RFID sebagai identifikasi unik yang akan dipindai oleh pembaca RFID. Ketika ID yang terdaftar cocok maka otorisasi akan diberikan. Dalam system *reverse vending machine*, RFID dapat mengidentifikasi pengguna secara cepat dan aman serta memudahkan pencatatan transaksi secara otomatis tanpa membuka aplikasi.

2.10 QR Code

Quick Response Code (QR Code) adalah kode dua dimensi yang dirancang untuk menyimpan berbagai jenis data, seperti angka, huruf, biner, hingga karakter kanji dalam format yang dapat dibaca mesin (Irawan & Adriantantri, 2018). Teknologi ini memungkinkan pembacaan cepat melalui kamera ponsel dan sering digunakan untuk menyimpan informasi unik seperti tautan atau data pengguna (Irawan & Adriantantri, 2018).

2.11 MQTT

Message Queuing Telemetry Transport (MQTT) adalah protokol messaging ringan yang dirancang khusus untuk komunikasi *machine-to-machine* (M2M) dan *Internet of Things* (Susanto et al., 2018). Protokol ini menggunakan model *publish/subscribe* yang memungkinkan pengiriman data dengan *overhead* minimal dan mendukung komunikasi yang *reliable* bahkan dalam kondisi jaringan yang tidak stabil. MQTT sangat cocok untuk implementasi IoT karena efisiensinya dalam penggunaan *bandwidth* dan kemampuannya untuk menangani sejumlah besar koneksi perangkat secara bersamaan.

2.12 WebSocket

WebSocket adalah protokol komunikasi yang menyediakan saluran komunikasi melalui koneksi TCP/IP (Kojansow et al., 2024). Berbeda dengan HTTP tradisional, WebSocket dapat melakukan komunikasi dua arah secara *real-time* antara client dan server. Protokol ini sangat efisien karena mengurangi *overhead* komunikasi dan latency dibandingkan dengan polling HTTP tradisional.

2.13 Node.js

Node.js merupakan *runtime environment* JavaScript yang dikembangkan untuk menjalankan kode pada sisi server. Teknologi ini menggunakan sistem *event-driven* dan

operasi input-output yang tidak menghalangi proses lain (Iskandar et al., 2024). Node.js tidak menggunakan sistem penguncian dalam mengelola proses bersamaan, sehingga membuat aplikasi berjalan lebih ringan dan efisien. Hal ini menjadikan Node.js sangat cocok untuk aplikasi yang membutuhkan pemrosesan data secara cepat dan *real-time*. Node.js juga mendukung arsitektur mikroservis dan dapat menangani ribuan koneksi bersamaan dengan throughput tinggi yang menjadikannya pilihan populer untuk pengembangan aplikasi *backend* modern.

2.14 Express.js

Express.js adalah sebuah kerangka kerja minimalis dan fleksibel yang berjalan di atas *Node.js*. Dokumentasinya yang komprehensif serta kemudahan penggunaannya memungkinkan pengembang untuk membangun berbagai jenis sistem mulai dari aplikasi web hingga RESTful API.

Dalam penelitian (Mulana et al., 2022), *Express.js* terbukti memiliki kinerja yang sangat baik dengan rata-rata waktu respons 438,64 ms dan penggunaan sumber daya CPU sebesar 1,95% serta memori 2,74%, menunjukkan keunggulan framework ini dalam menangani berbagai kebutuhan aplikasi dengan cepat.

Kerangka kerja ini memiliki sejumlah keunggulan, antara lain dukungan terhadap middleware, berbagai HTTP verb (*POST*, *GET*, *PUT*, *DELETE*), pengaturan *template engine*, pengelolaan file statis, serta fleksibilitas dalam penyesuaian. Semua aspek tersebut menjadikan *Express.js* pilihan menarik bagi para pengembang web Febrianto Widyoutomo et al.,(2021). Kemampuannya untuk menghasilkan RESTful API dengan mudah, dukungan terhadap berbagai metode HTTP, serta kinerja yang efisien membuat *Express.js* menjadi pilihan yang tepat untuk mengembangkan sistem. Fleksibilitas *framework* ini dapat dengan mudah dan cepat mengimplementasikan fitur-fitur kompleks dan mempertahankan kinerja yang optimal sekaligus memudahkan integrasi dengan aplikasi dan database.

2.15 Flask Python

Flask adalah sebuah *microframework* yang bersifat minimalis untuk pengembangan web yang dibangun dengan menggunakan bahasa pemrograman *Python*. *Framework* ini berfungsi sebagai basis struktur bagi pembuatan aplikasi dan antarmuka web sehingga para pengembang dapat lebih leluasa dalam menentukan struktur serta dinamika situs. Salah satu kelebihan utama yang ditawarkan oleh *Flask* adalah kemampuannya untuk memfasilitasi pengembangan dengan efisien dan dapat mengurangi waktu yang diperlukan untuk menciptakan aplikasi tingkat kompleksitas tinggi (Irsyad, 2018).

Keunggulan utama Flask *framework* terletak pada kemampuan integrasinya yang lancar dengan berbagai *library Python* lainnya. Dengan memanfaatkan arsitektur modular yang dimiliki *Flask*, para pengembang diberikan fleksibilitas yang besar dalam merancang aplikasi, baik yang sederhana maupun yang memiliki tingkat kompleksitas tinggi. Di samping itu, *framework* ini dirancang dengan kesederhanaan dan fleksibilitas yang memungkinkan penyesuaian sesuai dengan kebutuhan spesifik para pengguna (Srinath & Associate, 2017).

2.16 Computer Vision

Computer vision adalah sebuah proses pembelajaran dan analisis gambar atau video untuk menghasilkan keluaran yang serupa dengan apa yang dapat dilakukan oleh manusia. Dengan kata lain, *computer vision* bertujuan untuk meniru cara manusia memproses visualisasi (Umam & Negara, 2016).

Computer vision memiliki keterkaitan dengan beberapa bidang lainnya, seperti pengolahan citra (*image processing*) dan visi mesin (*machine vision*). Ada kesamaan signifikan dalam berbagai teknik dan penerapan di ketiga bidang ini yang menunjukkan adanya teknik dasar yang serupa. Secara umum, *computer vision* berhubungan erat dan diterapkan dalam berbagai bidang lain seperti kecerdasan buatan (*artificial intelligence*), robotika, otomasi industri, pengolahan sinyal, optik fisik, neurobiologi, dan sebagainya.

2.17 OpenCV

OpenCV (*Open Source Computer Vision Library*) adalah *library* perangkat lunak yang dirancang untuk aplikasi pengolahan citra dan visi komputer Zebua & Rosyani, (2024). *Library* ini menyediakan lebih dari 2500 algoritma yang dioptimalkan, termasuk fungsi-fungsi pengolahan citra seperti deteksi objek, segmentasi gambar, dan pengenalan pola. Dalam pengembangan sistem pengenalan sampah otomatis, OpenCV berperan dalam memproses dan menganalisis gambar untuk identifikasi jenis objek. Penerapan OpenCV juga mempermudah pembuatan sistem deteksi berbasis komputer karena algoritma yang sudah terintegrasi mampu mengolah data visual seperti pengenalan wajah, bentuk, atau objek lainnya (Zulkhaidi et al., 2020).

2.18 MobileNet SSD

MobileNet-SSD merupakan arsitektur *deep learning* yang menggabungkan dua komponen utama, yaitu MobileNet sebagai jaringan ekstraksi fitur dan *Single Shot Detector (SSD)* sebagai pendeteksi objek. Pendekatan ini dirancang untuk mengoptimalkan kecepatan tanpa mengabaikan tingkat akurasi dalam mendeteksi objek.

Berdasarkan hasil penelitian oleh Islam et al., (2024), MobileNet-SSD terbukti memiliki performa *real-time* terbaik dengan waktu inferensi tercepat dibandingkan algoritma lainnya meskipun memiliki tingkat akurasi yang lebih rendah. Selain itu, metode *Convolutional Neural Network* (CNN) yang digunakan dalam model ini telah banyak diakui efektif untuk menyelesaikan permasalahan terkait deteksi objek dan pengenalan citra karena tingkat akurasi yang tinggi dengan hasil yang signifikan dalam pengenalan citra I. Astuti et al., (2022). Dalam implementasi deteksi objek, MobileNet-SSD dipilih sebagai model utama dalam penelitian ini karena memiliki kecepatan dan akurasi yang baik dalam mendeteksi objek. SSD (*Single Shot Multibox Detector*) dikenal sebagai *one-stage detector* yang mampu melakukan proses klasifikasi dan lokalisasi objek secara bersamaan dalam satu jaringan sehingga lebih efisien dibandingkan model lain seperti Faster R-CNN dan YOLO (I. Astuti et al., 2022).

MobileNet SSD digunakan sebagai alternatif yang lebih mengoptimalkan untuk kecepatan, terutama jika dibandingkan dengan desain *deep learning* konvensional yang umumnya berfokus pada penambahan kedalaman jaringan (Howard et al., 2017). Sebagaimana dijelaskan, arsitektur MobileNet SSD yang ringan memungkinkan model ini untuk mendeteksi objek secara cepat dan akurat serta menjadikan solusi yang ideal untuk perangkat dengan keterbatasan sumber daya (Islam et al., 2024).

2.19 Blackbox Testing

Black box testing adalah metode pengujian perangkat lunak. *Black box testing* digunakan untuk mengidentifikasi cacat dan kekurangan seperti fungsionalitas yang tidak tepat, kesalahan dalam struktur data, serta isu terkait akses ke database dan elemen lainnya Hermawan et al., (2020).

Fokus dari pengujian ini adalah pada input dan output perangkat lunak yang memastikan bahwa aplikasi bekerja sesuai dengan spesifikasi dan persyaratan yang telah ditentukan. Penguji memberikan berbagai input dan memeriksa apakah output yang dihasilkan sesuai dengan yang diharapkan, tanpa mengetahui bagaimana sistem tersebut memproses input di balik layar. Metode ini efektif untuk menguji fungsionalitas sistem secara keseluruhan dan memastikan bahwa setiap fitur bekerja sesuai dengan spesifikasinya.

BAB III

METODOLOGI

3.1 Aktivitas Tim Perintisan Bisnis

Proses perintisan bisnis ini dikerjakan oleh sebuah kelompok bernama "Haystudio," yang terdiri dari tiga mahasiswa Program Studi Informatika, Universitas Islam Indonesia. Anggota kelompok dapat dilihat pada tabel berikut :

Tabel 3.1 Aktivitas Tim Perintisan Bisnis

No	Anggota	Deskripsi Tugas
1	Hasan Rama Sagita	Berperan sebagai <i>Hustler</i> yang memegang peran kunci sebagai penggerak utama dalam perancangan, manajemen, dan strategi pemasaran bisnis. <i>Hustler</i> berperan penting dalam memastikan keberhasilan dan keberlanjutan bisnis.
2	Jamaludin Haikal Al Ghifari	Berperan Sebagai <i>Hipster</i> yang memegang peran penting dalam perancangan dan pembuatan desain pengalaman pengguna atau kerangka desain serta pengembangan purwarupa. <i>Hipster</i> memastikan produk memiliki visual yang menarik dan fungsional sesuai dengan tujuan bisnis.
3	Gilang Aryo Pamungkas	Berperan sebagai <i>Hacker</i> yang bertanggung jawab atas perancangan arsitektur sistem, pengembangan aplikasi, dan mengintegrasikan perangkat IOT <i>Reverse Vending Machine</i> dengan aplikasi.

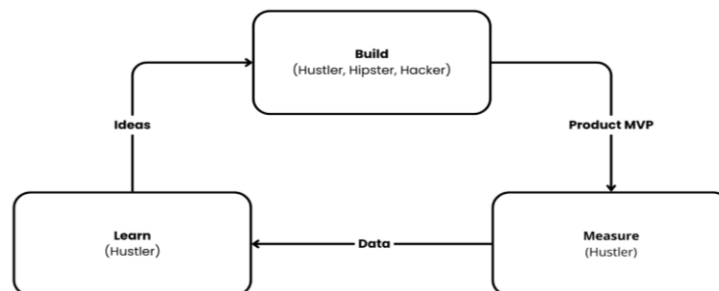
Tim Haystudio memulai proses penggalian ide untuk merumuskan hipotesis yang akan menjadi dasar pengembangan bisnis mereka. Proses ini dimulai dengan sesi brainstorming, di mana setiap anggota tim didorong untuk berkontribusi dengan berbagai ide. Selama sesi brainstorming, terdapat beberapa aturan penting yang diterapkan untuk menjaga kreativitas tetap terbuka. Salah satunya adalah kebebasan bagi setiap anggota tim untuk menyampaikan ide tanpa batasan atau hambatan, serta larangan keras untuk mengkritik ide orang lain pada tahap awal dengan tujuan mengumpulkan sebanyak mungkin ide yang berpotensi.

Setelah semua ide dikumpulkan dan diskusi selesai, tahap ideasi pun dilakukan. Dalam tahap ini, dilakukan seleksi ide-ide yang dianggap paling relevan, mendesak, dan mungkin untuk diwujudkan. Ide-ide yang dipilih ini akan dikembangkan lebih lanjut menjadi hipotesis yang akan diuji pada tahap pengembangan selanjutnya. Di sisi lain, beberapa ide yang dipertimbangkan pada akhirnya tidak terpilih karena alasan tertentu, seperti tingkat kompleksitas teknis, potensi pasar yang rendah, atau regulasi yang sulit dicapai.

Sebagai contoh, beberapa ide yang tidak dipilih antara lain KAKILIMA, sebuah platform berbasis *blockchain* yang menghubungkan pedagang kaki lima dengan pemilik lahan. Meskipun inovatif, ide ini dianggap terlalu kompleks untuk diimplementasikan pada tahap awal dan memerlukan regulasi khusus. Ide lain seperti E-Ring juga tidak dipilih karena tingkat adopsi pasar yang rendah untuk perangkat wearable, serta Turu, sebuah tempat tidur pintar yang dinilai hanya menarik untuk pasar yang sangat terbatas.

Namun, dari sekian banyak ide, tim akhirnya memilih Sampahmas sebagai proyek yang akan dikembangkan lebih lanjut. Ide ini dinilai sangat relevan dengan isu pengelolaan sampah yang menjadi perhatian publik, terutama di daerah perkotaan. Selain itu, teknologi seperti *Reverse Vending Machine* (RVM) yang digunakan dalam Sampahmas telah terbukti sukses di beberapa negara maju. Dibandingkan dengan ide-ide lainnya, Sampahmas dinilai lebih mudah untuk dikembangkan dengan biaya yang lebih rendah dan berpotensi memberikan dampak sosial serta ekonomi yang signifikan.

3.2 Pendekatan Lean Startup



Gambar 3.1 Diagram *Lean Startup Methodology*

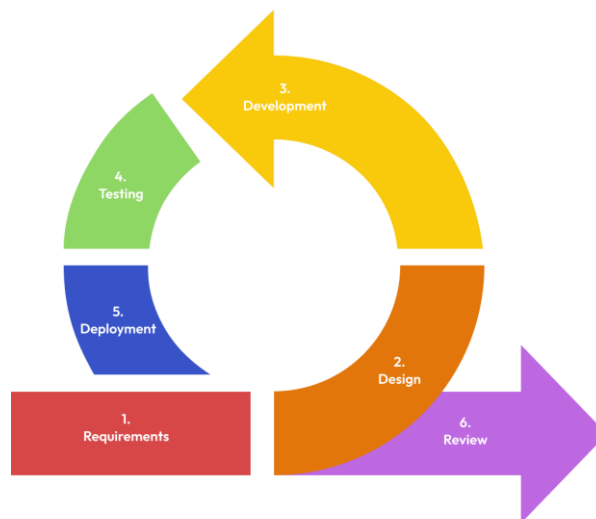
Pengembangan sistem Sampahmas akan mengadopsi metode lean startup, yang terdiri dari tiga tahapan utama: *Build*, *Measure*, dan *Learn*. Setiap iterasi dalam pengembangan ini melibatkan peran dari anggota tim *hacker*, *hipster*, dan *hustler* yang bekerja secara kolaboratif untuk memastikan setiap fitur dan aspek teknis maupun non-teknis dikembangkan dengan cepat dan relevan terhadap kebutuhan pasar. Berikut penjelasan tahapan *Lean Startup* di setiap peran masing-masing anggota tim dalam perintisan bisnis ini:

3.3 Iterasi 1 – Problem – Solution Fit

Pada iterasi pertama, tim memulai dengan menetapkan target pengguna secara jelas. Pada tahap ini, pengembangan *Minimum Viable Product* (MVP) dilakukan sebagai langkah awal untuk menguji hipotesis pasar. MVP ini berfungsi sebagai versi dasar produk yang mencakup fitur-fitur inti dengan tujuan untuk mendapatkan umpan balik awal dari pengguna sebelum melakukan pengembangan lebih lanjut.

Diawali dengan Hustler melakukan build atau menganalisis pasar, menyusun diagram BPMN, dan membuat kuesioner untuk memahami kebutuhan pengguna. Selanjutnya, *measure* atau Hustler menguji MVP pada pengguna untuk mengevaluasi ketertarikan terhadap ide bisnis dan *Learn* atau Hustler mengumpulkan dan menganalisis umpan balik untuk menentukan langkah pengembangan berikutnya.

Kemudian, *Hipster* memilih fitur utama MVP (sistem insentif poin), mendesain elemen branding (logo dan palet warna) serta menyiapkan kuesioner berbasis *Google Form*. *Hipster* fokus pada pengembangan visual tanpa terlibat dalam tahap pengukuran dan evaluasi.



Gambar 3.2 Metode Agile

Hacker menggunakan pendekatan Agile dalam pengembangan perangkat lunak untuk memastikan proses pengembangan dapat merespons kebutuhan yang berubah-ubah secara efektif dan adaptif. Tahap pengembangan sistem dijalankan sesuai dengan tugas-tugas yang telah direncanakan dalam sprint. Menurut (Santoso et al., 2022), *sprint* adalah periode waktu yang ditentukan untuk menyelesaikan bagian tertentu dari produk yang sedang dikembangkan dengan durasi yang terbatas. Proses pengembangan pada bagian *Hacker* dimulai dengan tahap analisis kebutuhan sistem. Tahap ini bertujuan untuk mengidentifikasi kebutuhan utama sistem dan memastikan bahwa fitur-fitur yang dikembangkan sesuai dengan tujuan aplikasi. *Hacker* fokus pada pengumpulan informasi terkait kebutuhan teknis, alat yang akan digunakan, serta fitur utama yang harus disiapkan. Informasi ini dikumpulkan melalui diskusi tim dan konsultasi dengan pemangku kepentingan.

Selanjutnya, *Hacker* memasuki fase perancangan sistem. Pada tahap ini, desain arsitektur sistem dibuat, yang mencakup desain database yang digunakan untuk mengelola data aplikasi. Selain itu, desain antarmuka pemrograman aplikasi (API) dibuat untuk menghubungkan komponen *frontend* dan *backend*. Desain ini

memastikan struktur sistem yang jelas sehingga dapat dilanjutkan pengembangan. Tahap berikutnya adalah menerapkan sistem. Hasil desain mulai diterjemahkan oleh *hacker*. Pengembangan logika sistem, integrasi komponen, dan pengujian awal secara internal adalah semua bagian dari proses ini. Setiap fitur diuji secara menyeluruh untuk memastikan bahwa tidak ada bug yang signifikan dan bahwa semua komponen bekerja dengan benar. Setelah pengujian selesai, sistem kemudian diterapkan ke lingkungan produksi melalui proses *deployment*. Pada tahap ini, sistem dipindahkan dari lingkungan pengembangan ke lingkungan yang siap diujikan kepada pengguna. Untuk memastikan bahwa sistem dapat berjalan dengan lancar tanpa masalah, deployment dilakukan dengan memastikan bahwa semua konfigurasi telah sesuai.

Tahap terakhir adalah evaluasi dan revisi. Ini dilakukan untuk memastikan sistem berfungsi secara optimal dan memenuhi kebutuhan pengguna. Jika ada kesalahan, perbaikan atau peningkatan dilakukan untuk menjaga kualitas dan relevansi sistem.

3.3.1 Requirement

Pada tahap ini, dimulai dengan Tahap requirement, *hacker* mengidentifikasi kebutuhan utama sistem yaitu implementasi mekanisme konversi sampah menjadi poin berdasarkan spesifikasi yang telah dirancang oleh *Hipster*. Pada tahap ini, fokus utamanya adalah memahami bagaimana sistem akan mengkonversi botol yang dibuang menjadi poin reward dan bagaimana poin tersebut dapat diintegrasikan dengan fitur potongan harga.

3.3.2 Design

Setelah persyaratan dipahami dengan baik, *Hacker* bergerak ke tahap design. Tahap ini membuat rancangan arsitektur sistem secara keseluruhan. Tahap ini mencakup pembuatan skema rangkaian untuk komponen hardware, pengembangan diagram sequence untuk model MVP dan yang terpenting adalah perancangan struktur database yang akan digunakan untuk menyimpan dan mengelola data sistem. Semua desain ini dibuat dengan mempertimbangkan alur sistem untuk deteksi botol dan akumulasi poin.

3.3.3 Development

Pada tahap development, implementasi dibagi menjadi dua fokus utama. Pertama adalah implementasi hardware yang meliputi integrasi sensor infrared untuk mendeteksi botol dan yang kedua adalah pengembangan software.

3.3.4 Testing

Tahap testing dilakukan dengan metode blackbox testing yang fokus pada pengujian fungsionalitas penambahan poin. *Hacker* memastikan bahwa setiap kali botol terdeteksi, sistem secara akurat menambahkan poin sesuai dengan yang telah ditentukan.

3.3.5 Deployment

Pada tahap development, implementasi dibagi menjadi dua fokus utama. Pertama adalah implementasi hardware yang meliputi integrasi sensor infrared untuk mendeteksi botol dan yang kedua adalah pengembangan software.

3.3.6 Review

Terakhir adalah tahap review yang dilakukan untuk evaluasi produk oleh *hustler*. Pada tahap ini, *hacker* menerima umpan balik tentang kinerja sistem, mengidentifikasi area yang membutuhkan perbaikan dan merencanakan pengembangan fitur selanjutnya berdasarkan masukan yang diterima. Seluruh proses pengembangan ini dilakukan dengan tujuan untuk menciptakan sistem yang dapat berfungsi optimal dan siap untuk dikembangkan lebih lanjut sesuai kebutuhan pengguna.

3.4 Iterasi ke-2-Product-Market Fit

Pada iterasi kedua, tim Sampahmas fokus pada pencapaian *Product Market Fit* untuk memastikan produk sesuai dengan kebutuhan pasar. Pada fase ini, dilakukan penyesuaian fitur berdasarkan umpan balik dari pengguna awal serta pengujian lebih lanjut terhadap MVP yang telah diperbarui.

Pada tahap *Build*, *Hustler* melakukan analisis persona pengguna berdasarkan umpan balik sebelumnya, mengolah data hasil pengujian awal, dan melakukan riset pasar tambahan. Selain itu, *Hustler* mengembangkan wireframe awal aplikasi Sampahmas yang akan digunakan oleh *Hipster* untuk desain UI/UX. Kemudian, Pada tahap *Measure*, *Hustler* menguji MVP yang telah diperbarui kepada pengguna

untuk mengevaluasi apakah fitur-fitur baru sudah memenuhi kebutuhan. *Hustler* juga menggunakan *System Usability Scale* (SUS) yang dirancang oleh *Hipster* untuk mengukur pengalaman pengguna. Tahap terakhir yakni tahap *Learn*, *Hustler* mengumpulkan umpan balik melalui wawancara singkat untuk memahami pengalaman pengguna dalam menggunakan aplikasi. Analisis ini digunakan untuk mengevaluasi apakah perubahan pada MVP telah memperbaiki kelemahan sebelumnya. sebelumnya.

Kemudian, Pada iterasi ini, *Hipster* melanjutkan perancangan fitur berdasarkan hasil wawancara pengguna. *Hipster* juga memperdalam analisis persona pengguna untuk merancang desain UI/UX yang lebih intuitif yakni membuat tampilan awal antarmuka aplikasi. *Hipster* tidak terlibat dalam tahap *Measure* dan *Learn* karena fokusnya pada pengembangan visual dan UI/UX.

Pada tahap ini, seorang hacker mengambil langkah lebih lanjut dengan mengimplementasikan berbagai fitur tambahan berdasarkan hasil diskusi dan masukan yang diberikan oleh *hustler* dan *hipster*.

3.4.1 Requirement

Pada tahap ini, *hacker* memulai dengan mengidentifikasi kebutuhan utama sistem untuk iterasi kedua. Fokusnya adalah memperbaiki dan mengelola kebutuhan database aplikasi agar mampu menyimpan dan mengolah data dari interaksi pengguna dan perangkat secara efisien. *Hacker* juga memastikan database dirancang fleksibel, skalabilitas, dan dapat menangani peningkatan jumlah data tanpa mengganggu performa sistem. Selain itu, *hacker* merancang kebutuhan *API Gateway* yang akan digunakan untuk komunikasi antara aplikasi mobile dan perangkat *reverse vending machine*, memastikan aliran data berjalan lancar.

3.4.2 Design

Setelah persyaratan teridentifikasi, *hacker* masuk ke tahap desain. Pada tahap ini, *hacker* merancang struktur database yang lebih optimal untuk mendukung kebutuhan aplikasi Sampahmas. Selain itu, dirancang endpoint-endpoint pada *API Gateway* yang berfungsi untuk mengelola data dari aplikasi mobile dan vending machine. *Hacker* juga mendesain integrasi antara vending machine dengan *API* untuk memastikan komunikasi data *real-time*. Di sisi lain, implementasi *WebSocket*

dirancang untuk menangani data poin agar perubahan atau penambahan poin langsung diperbarui di antarmuka pengguna tanpa perlu refresh manual.

3.4.3 Development

Pada tahap pengembangan, *hacker* mulai mengimplementasikan database yang mampu mengelola data dengan skala yang lebih besar. *Hacker* juga membangun *API Gateway* berdasarkan rancangan endpoint yang telah dibuat, menghubungkan aplikasi mobile dan perangkat vending machine. Integrasi perangkat vending machine dilakukan untuk memastikan sistem dapat menerima dan mengirim data secara *real-time*. Selain itu, *hacker* mengembangkan *WebSocket* agar pembaruan data poin pada aplikasi lebih responsif dan fleksibel.

3.4.4 Testing

Tahap pengujian dilakukan menggunakan metode *blackbox testing* untuk memastikan fungsionalitas *API Gateway* dan komunikasi *real-time* berjalan dengan baik. *Hacker* juga menguji integrasi antara *vending machine* dan sistem *backend* untuk memastikan data dapat dikirim dan diterima tanpa kendala.

3.4.5 Deployment

Pada tahap ini, *hacker* merilis sistem *API Gateway* dan database yang telah diuji kepada tim lain, termasuk *hustler* dan *hipster*. Sistem ini siap digunakan untuk mendukung pengujian lebih lanjut serta mendapatkan umpan balik dari pengguna awal.

3.4.6 Review

Tahap akhir dari iterasi ini adalah review, di mana *hacker* menerima umpan balik dari *hustler* dan pengguna terkait performa sistem. Umpan balik ini digunakan untuk mengevaluasi area yang perlu diperbaiki dan merencanakan pengembangan fitur tambahan untuk iterasi berikutnya. Dengan pendekatan ini, sistem diharapkan dapat berfungsi lebih optimal dan mendukung kebutuhan pengguna secara menyeluruh.

3.5 Iterasi ke-3-Scaling & Optimization

Pada iterasi ketiga, tim berfokus pada pengoptimalan produk serta peningkatan skalabilitas aplikasi. Umpan balik dan hasil pengujian dari iterasi

sebelumnya dimanfaatkan untuk memperbaiki produk. Pada fase ini, peran *Hacker* berfokus pada optimasi sistem aplikasi secara menyeluruh.

Hustler pada iterasi ini menyusun *Lean Canvas* dan mengikuti lomba bisnis untuk memvalidasi hipotesis serta mencari peluang investasi (*Build*). Kemudian, untuk memvalidasi hipotesis bisnis dan mencari potensi pendanaan, *Hustler* mengikuti lomba bisnis dan menguji model bisnis Sampahmas, memperoleh umpan balik dari juri, dan mengevaluasi daya tarik platform bagi investor (*Measure*). Dengan berpartisipasi dalam lomba bisnis dan pengujian hipotesis sementara menjadi dasar bagi *Hustler* untuk memperbaiki dan memperkuat model bisnis Sampahmas. Berdasarkan umpan balik juri dan investor, *Hustler* memperbaiki hipotesis dan strategi bisnis agar lebih sesuai dengan kebutuhan pasar (*Learn*).

Hipster pada tahap ini akan berfokus memperbaiki desain UI/UX berdasarkan umpan balik iterasi sebelumnya, menyempurnakan elemen visual, serta membuat tampilan aplikasi lebih matang dan user-friendly.

Sedangkan *Hacker* pada tahap ini akan mengambil langkah lebih lanjut dengan mengimplementasikan berbagai fitur tambahan berdasarkan hasil diskusi dan masukan yang diberikan oleh *hustler* dan *hipster*.

3.5.1 Requirement

Pada tahap ini, *Hacker* memulai dengan menganalisis kebutuhan sistem berdasarkan hasil *review* yang dilakukan oleh *Hustler*. Analisis ini mencakup evaluasi terhadap umpan balik pengguna untuk memastikan bahwa sistem yang dikembangkan mampu menjawab kebutuhan dan permasalahan pengguna dengan lebih baik. Fokus utama dalam iterasi ini adalah penambahan fitur daftar kartu yang memungkinkan pengguna mendaftarkan kartu mereka secara langsung melalui *reverse vending machine*. Selain itu, *Hacker* juga memastikan bahwa sistem mendukung pengelolaan data pengguna yang terintegrasi dengan perangkat *Reverse Vending Machine* (RVM). Analisis kebutuhan ini menjadi dasar untuk mengembangkan fitur baru yang relevan dan meningkatkan performa sistem secara keseluruhan.

3.5.2 Design

Setelah kebutuhan sistem teridentifikasi, *Hacker* masuk ke tahap desain untuk menyusun arsitektur sistem yang mendukung fitur baru. Pada tahap ini, *sequence diagram* dibuat untuk menggambarkan alur komunikasi data antara aplikasi mobile, perangkat RVM, dan *backend*. Diagram ini berfungsi sebagai panduan teknis bagi tim untuk memastikan bahwa setiap komponen dalam sistem bekerja secara sinergis. Desain ini juga mencakup penyesuaian endpoint API agar mampu menangani data pendaftaran kartu RFID sekaligus mendukung integrasi dengan fitur-fitur lain yang sudah ada.

3.5.3 Development

Pada tahap pengembangan, *Hacker* mulai mengimplementasikan fitur pendaftaran kartu NFC pada *backend*. Fitur ini dirancang untuk memungkinkan pengguna mendaftarkan kartu mereka dengan mudah melalui aplikasi, sementara data tersebut langsung disinkronkan dengan perangkat RVM. Selain itu, *Hacker* melakukan integrasi antara API Gateway dan perangkat RVM untuk memastikan bahwa sistem dapat berkomunikasi secara *real-time* tanpa hambatan. Proses pengembangan juga mencakup optimalisasi sistem untuk mendukung penggunaan kartu NFC dalam skala besar, sehingga sistem tetap responsif meskipun jumlah pengguna meningkat. Dengan pendekatan yang terstruktur, *Hacker* memastikan setiap fitur baru berfungsi dengan baik dalam keseluruhan ekosistem aplikasi.

3.5.4 Testing

Setelah pengembangan selesai, tahap pengujian dilakukan untuk memastikan bahwa setiap fitur berjalan sesuai spesifikasi yang telah dirancang. *Hacker* menggunakan metode *blackbox testing* untuk menguji fungsionalitas pendaftaran kartu NFC, termasuk alur pendaftaran, pengolahan data pengguna, dan integrasi dengan perangkat RVM. Pengujian ini juga difokuskan pada validasi endpoint API untuk memastikan bahwa sistem dapat menangani permintaan data tanpa error atau keterlambatan. Selain itu, *Hacker* menguji stabilitas sistem untuk memastikan bahwa integrasi fitur baru tidak mengganggu fungsi-fungsi yang sudah ada. Dengan pengujian yang mendalam, *Hacker* memastikan bahwa sistem siap untuk diimplementasikan.

3.5.5 Deployment

Setelah tahap pengujian selesai dan fitur dinyatakan memenuhi kriteria, *Hacker* melanjutkan ke tahap deployment. Pada tahap ini, sistem yang telah diperbarui diterapkan pada prototipe RVM untuk mendukung pengujian lebih lanjut di lingkungan nyata. Deployment ini mencakup instalasi *backend* yang telah diperbarui, termasuk integrasi fitur kartu NFC, API Gateway, dan sinkronisasi data dengan perangkat RVM. Dengan deployment ini, *Hacker* memastikan bahwa sistem siap digunakan oleh pengguna awal untuk mendapatkan umpan balik yang lebih komprehensif.

3.5.6 Review

Tahap terakhir dalam iterasi ini adalah review, di mana *Hacker* menerima umpan balik dari *Hustler* terkait performa produk yang telah diperbarui. Umpan balik ini mencakup evaluasi terhadap fungsi pendaftaran kartu NFC, stabilitas integrasi dengan perangkat RVM, dan pengalaman pengguna secara keseluruhan. Dengan masukan dari *Hustler*, *Hacker* dapat mengidentifikasi area yang masih memerlukan perbaikan dan merencanakan pengembangan lebih lanjut untuk iterasi berikutnya. Review ini memastikan bahwa sistem terus berkembang dan mampu memenuhi kebutuhan pengguna serta ekspektasi pasar dengan lebih baik.

BAB IV

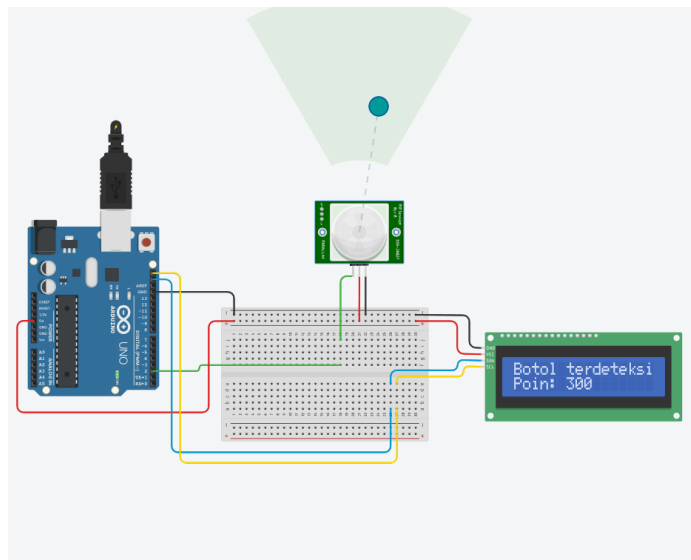
HASIL DAN PEMBAHASAN

4.1 Iterasi 1

4.1.1 Requirement

Pada iterasi pertama, kebutuhan sistem diidentifikasi untuk memastikan bahwa prototipe Reverse Vending Machine (RVM) mampu mendeteksi botol secara otomatis dan mengkonversinya menjadi poin reward. Proses ini dilakukan untuk menjawab kebutuhan pengguna yang ingin mendapatkan insentif melalui penukaran sampah botol plastik dengan mekanisme yang sederhana dan efisien.

4.1.2 Design



Gambar 4.1 Desain alat deteksi botol

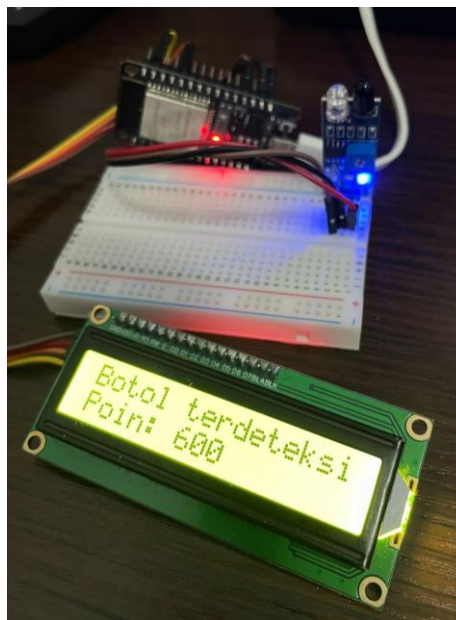
Pada gambar di atas menunjukkan bagaimana sistem *Reverse Vending Machine* sederhana ini bekerja dalam mendeteksi botol yang dimasukkan oleh pengguna. Dengan menggunakan sensor inframerah, sistem mampu mendeteksi adanya gerakan botol yang mendekati sensor. Ketika botol terdeteksi, sistem secara otomatis menambahkan jumlah botol yang berhasil dimasukkan ke dalam variabel penghitung dan mengakumulasi poin yang diperoleh oleh pengguna. Poin ini kemudian ditampilkan pada layar LCD, sehingga pengguna dapat melihat jumlah poin yang telah terkumpul. Pendekatan ini dirancang untuk memberikan pengalaman interaktif dan insentif bagi pengguna, sekaligus mengotomatisasi

proses pengumpulan dan pendataan botol dengan bantuan teknologi mikrokontroler dan sensor sederhana.

Tahap desain mencakup penyusunan diagram alur kerja sistem, rancangan integrasi perangkat keras dan perangkat lunak, serta pembuatan skema pengolahan data. Sistem dirancang dengan memanfaatkan kontroler utama yaitu ESP-32 dan sensor inframerahs untuk mendeteksi botol.

4.1.3 Development

Pada Hasil dari pengembangan sistem *Reverse Vending Machine* pada iterasi pertama adalah sebuah prototipe yang mampu mendeteksi penukaran botol secara otomatis. Sistem ini dibangun menggunakan komponen hardware yaitu ESP32 dan sensor inframerah (IR).



Gambar 4.2 Prototyping alat deteksi

Dalam prototipe ini, sensor inframerah ditempatkan pada jalur masuk botol untuk mendeteksi adanya benda yang melintasi area sensor. Ketika sebuah botol melewati sensor inframerah, sinyal dari sensor akan berubah yang menandakan adanya objek terdeteksi. ESP32 kemudian membaca perubahan sinyal ini dan menganggapnya sebagai satu botol yang masuk. Setiap kali sensor mendeteksi adanya perubahan status, ESP32 mencatat satu botol dan menambah jumlah poin sesuai dengan aturan yang telah ditentukan. Dalam pengembangan ini, satu botol

yang berhasil dideteksi oleh sistem akan menghasilkan 100 poin yang secara otomatis ditambahkan ke total poin.

```
void loop() {  
    int sensorStatus = digitalRead(pirPin);  
    if (sensorStatus == HIGH && !isTriggered) {  
        digitalWrite(ledPin, HIGH);  
        if (millis() - lastTriggered > waitForNextTrigger) {  
            bottleCount++;  
            points += pointsPerBottle;  
            lastTriggered = millis();  
  
            Serial.print("Botol terdeteksi. Jumlah: ");  
            Serial.println(bottleCount);  
            Serial.print("Poin total: ");  
            Serial.println(points);  
            lcd.clear();  
            lcd.setCursor(0, 0);  
            lcd.print("Botol terdeteksi!");  
            lcd.setCursor(0, 1);  
            lcd.print("Poin: ");  
            lcd.print(points);  
        }  
        isTriggered = true;  
    }  
    if (sensorStatus == LOW && isTriggered) {  
        digitalWrite(ledPin, LOW);  
        isTriggered = false;  
    }  
    delay(50);  
}
```

Dalam pengembangan ini, ESP32 menggunakan teknik pemrograman yang sederhana untuk mendeteksi perubahan status dari sensor inframerah yang terbagi dalam dua kondisi, yaitu LOW dan HIGH. Ketika sensor inframerah berada dalam kondisi LOW, sistem menganggap bahwa sebuah botol telah terdeteksi. Setelah itu, ESP32 akan mencatat satu botol yang masuk dan menambahkan sejumlah poin ke total poin pengguna. Setiap kali sensor mendeteksi botol, variabel penghitung di dalam sistem akan bertambah untuk mencatat jumlah botol yang berhasil dimasukkan. aplikasi.

4.1.4 Testing

Sesuai hasil diskusi antara *Hustler* dan *Hipster*, pengembangan prototipe ini memenuhi persyaratan dasar untuk diuji lebih lanjut. Dalam tahap selanjutnya, tim *Hustler* akan melakukan pengujian dan validasi produk ini terhadap stakeholder guna mendapatkan umpan balik dan memperbaiki kelemahan yang ada.

4.1.5 Deployment

Setelah pengujian selesai, prototipe MVP diserahkan kepada tim *Hustler* untuk deployment awal. *Hustler* bertugas untuk menerapkan produk dalam skala kecil di lingkungan nyata dengan tujuan menguji kemampuannya berinteraksi dengan pengguna awal. Deployment mencakup pemasangan perangkat keras seperti ESP32 dan sensor inframerah, serta pengaturan sistem yang mendukung fungsi deteksi botol dan pengelolaan poin. Hasil deployment ini membantu tim mendapatkan wawasan lebih baik tentang performa sistem di luar lingkungan pengembangan.

4.1.6 Review

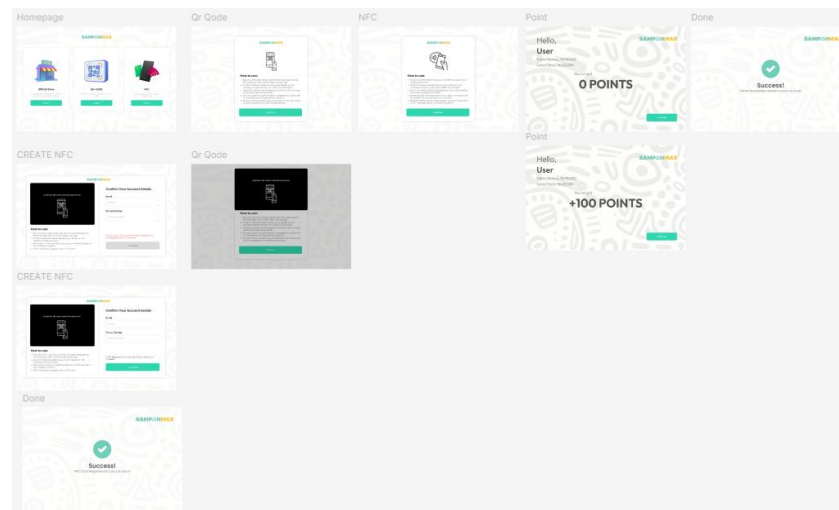
Pada tahap ini, *hacker* menyerahkan tugas kepada *Hustler*. *Hustler* kemudian bertanggung jawab melakukan review dengan stakeholder dan pengguna awal untuk mengevaluasi penerimaan produk. Review ini mencakup penilaian terhadap fungsi utama prototipe seperti kemampuan mendeteksi botol dan mengelola poin, serta pengalaman pengguna selama berinteraksi dengan sistem. Umpan balik yang diperoleh pada tahap ini menjadi dasar bagi *Hacker* untuk menyempurnakan sistem pada iterasi berikutnya yang akan memastikan produk lebih stabil dan siap untuk pengembangan fitur lanjutan.

4.2 Iterasi 2

4.2.1 Requirement

Pada iterasi kedua, pengembangan sistem difokuskan pada implementasi beberapa fitur utama untuk meningkatkan fungsionalitas dan kemudahan penggunaan Reverse Vending Machine (RVM). Sistem autentikasi dikembangkan dengan dua metode yaitu pemindaian barcode untuk identifikasi pengguna melalui aplikasi dan autentikasi menggunakan kartu RFID yang memungkinkan pengguna melakukan transaksi penukaran sampah secara langsung tanpa menggunakan smartphone.

Untuk meningkatkan aksesibilitas layanan, sistem dilengkapi dengan peta interaktif yang memungkinkan pengguna menemukan lokasi RVM terdekat dengan mudah. Inovasi berkelanjutan juga dihadirkan melalui Green Market, sebuah platform yang menyediakan produk-produk hasil daur ulang dari sampah yang telah dikumpulkan.



Gambar 4.3 Desain antarmuka *Reverse Vending Machine*

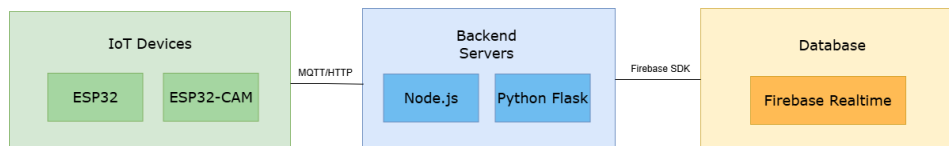
Berdasarkan hasil desain tampilan dari Hipster, kebutuhan antarmuka sistem RVM mencakup sepuluh halaman utama. Halaman awal dirancang untuk menampilkan tiga metode autentikasi yakni QR code, kartu kredit/debit, dan kartu NFC. Dua halaman berikutnya dialokasikan untuk menampilkan panduan penggunaan QR code dan NFC. Sistem juga memerlukan dua halaman khusus

untuk proses pendaftaran kartu NFC yang dilengkapi form input email dan nomor telepon. Untuk pengecekan poin, dibutuhkan dua variasi halaman yang menampilkan informasi saldo poin pengguna. Selain itu, diperlukan halaman pemindai QR code serta halaman konfirmasi sukses untuk menandai keberhasilan transaksi. Setiap halaman harus mengikuti standar desain yang telah ditetapkan oleh tim Hipster.

4.2.2 Design

A. Arsitektur Sistem

Pada tahap desain iterasi kedua, perancangan arsitektur sistem *Reverse Vending Machine* (RVM) mengimplementasikan pendekatan terintegrasi yang terdiri dari tiga komponen utama. Perangkat IoT terdiri dari ESP32 sebagai kontroler utama dan ESP32-CAM untuk akuisisi gambar. ESP32 bertanggung jawab mengelola autentikasi RFID sementara ESP32-CAM berperan dalam pengambilan gambar. Kedua perangkat ini dirancang untuk berkomunikasi dengan server melalui protokol MQTT dan HTTP.



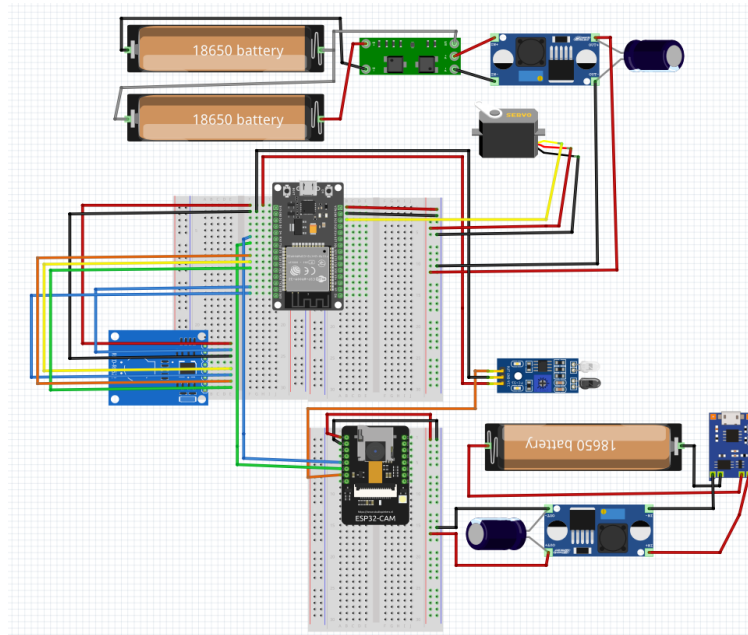
Gambar 4.4 Desain arsitektur sistem SampahMas

Server *backend* diimplementasikan menggunakan dua teknologi utama yakni Node.js untuk API Gateway dan Python Flask untuk pemrosesan gambar. Node.js server dirancang untuk menangani proses autentikasi, manajemen data dan komunikasi *real-time*, sedangkan Flask server berfokus pada validasi gambar menggunakan *computer vision*. Pemilihan teknologi ini didasarkan pada kebutuhan untuk memisahkan tanggung jawab pemrosesan data. Node.js akan menangani operasi umum sementara Flask dikhususkan untuk analisis gambar.

B. Desain sirkuit diagram

Berdasarkan evaluasi *Hipster*, sistem RVM mengalami pengembangan fitur tambahan. Salah satu inovasi terbaru adalah integrasi modul pembaca RFID yang terhubung dengan ESP32 melalui komunikasi SPI (*Serial Peripheral Interface*).

Penambahan ini dilakukan untuk memenuhi kebutuhan autentikasi dan otorisasi pengguna yang diidentifikasi melalui evaluasi sebelumnya.



Gambar 4.5 Desain sirkuit diagram

Gambar 4.5 menunjukkan konfigurasi pin antara ESP32, ESP32-CAM, dan sensor pendukungnya. ESP32 menggunakan pin GPIO21 (SDA) dan GPIO22 (RST) untuk komunikasi dengan modul RFID RC522 melalui protokol SPI dengan pin SCK, MISO, dan MOSI terhubung ke pin default SPI ESP32. Servo motor terhubung ke GPIO13 untuk menggerakkan penutup jalur botol.

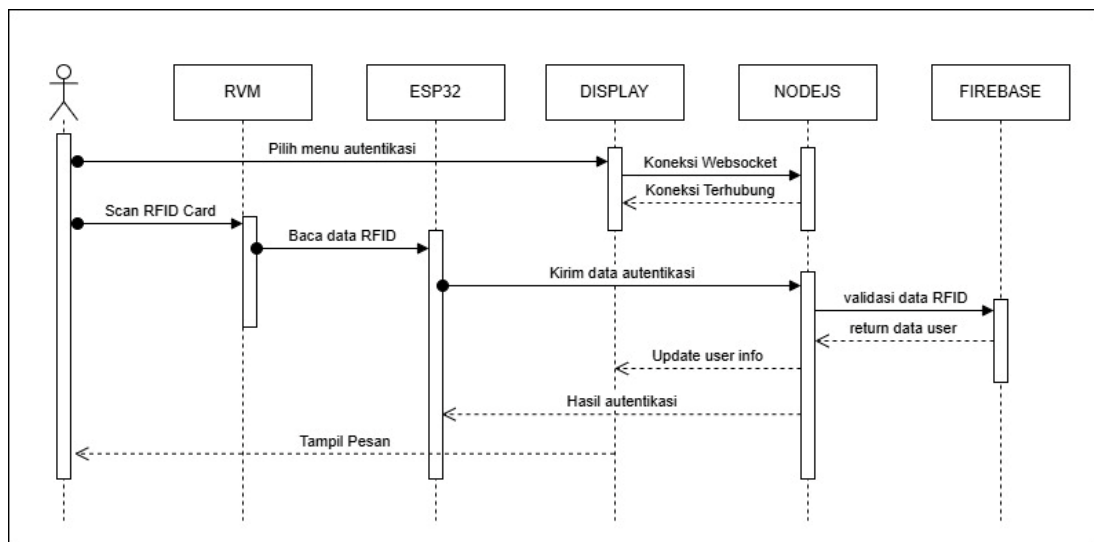
Kemudian untuk komunikasi serial UART antara ESP32 dan ESP32-CAM menggunakan pin GPIO16 (RX) dan GPIO17 (TX) pada ESP32 yang terhubung ke pin IO14R dan IO15T ESP32-CAM. Pada ESP32-CAM, sensor infrared terhubung ke GPIO12 untuk deteksi objek, sementara LED Flash dikontrol melalui GPIO4 secara internal tanpa penyambungan kabel. Semua sensor dan modul berbagi ground yang sama agar referensi tegangan yang seragam.

Untuk power kedua kontroler diperlukan dua kelompok power supply terpisah untuk pengoperasian sistem. Power supply pertama terdiri dari dua baterai 18650 (7.4V) yang dihubungkan dengan BMS 2S untuk proteksi, kemudian diturunkan menjadi 6V melalui step-down converter untuk mensuplai ESP32 dan servo motor. Modul RFID RC522 mendapatkan power secara langsung dari pin ESP32. Power

supply kedua menggunakan satu baterai 18650 (3.7V) yang dinaikkan menjadi 5V melalui step-up converter untuk mensuplai ESP32-CAM dan sistem infrared.

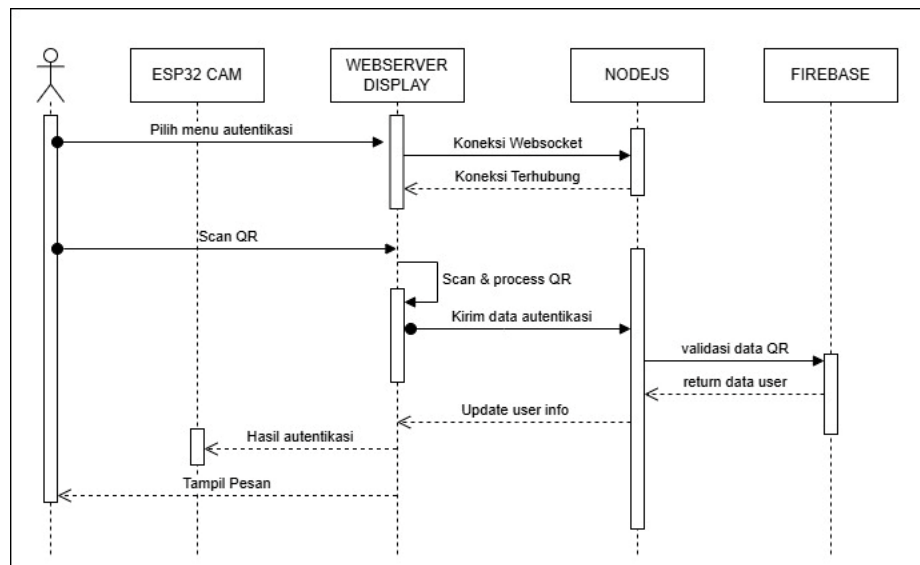
Setiap output regulator tegangan dipasang kapasitor 470 μ F secara paralel untuk menjaga stabilitas tegangan. ESP32 dan ESP32-CAM berkomunikasi melalui serial UART dimana ESP32-CAM akan mengirimkan data hasil deteksi botol menggunakan sensor infrared ke ESP32 utama untuk diproses lebih lanjut ke server.

C. Sequence Diagram



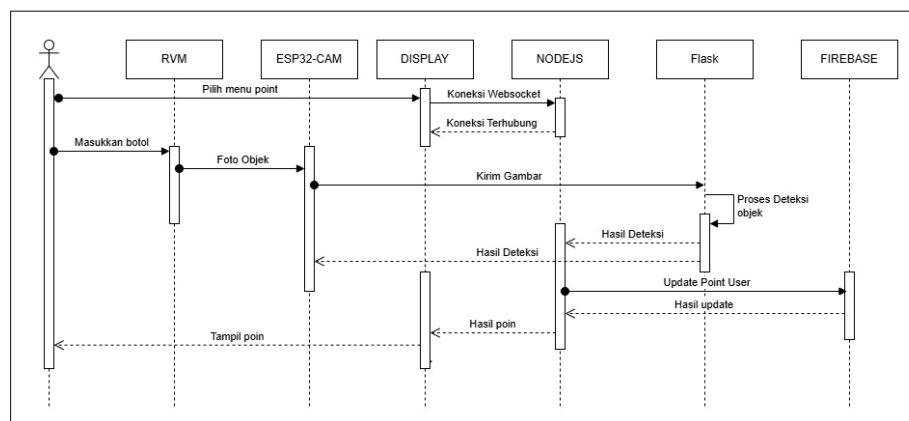
Gambar 4.6 Sequence diagram autentikasi dengan kartu

Gambar 4.6 menunjukkan alur proses autentikasi menggunakan kartu RFID pada sistem RVM. Alur autentikasi dimulai ketika pengguna menempelkan kartu RFID pada reader yang terpasang di RVM. ESP32 akan membaca data RFID tersebut dan mengirimkannya ke Node.js server. Server kemudian melakukan validasi data RFID dengan Firebase untuk memverifikasi identitas pengguna. Setelah proses validasi berhasil, server mengirimkan data pengguna ke ESP32 dan secara bersamaan mengirimkan data untuk memperbarui tampilan pada display.



Gambar 4.7 Sequence diagram autentikasi dengan QR Code

Terlihat gambar 4.7 menunjukkan alur proses autentikasi menggunakan QR Code. Berbeda dengan RFID, proses pemindaian QR Code dilakukan langsung melalui kamera yang terpasang pada tablet display. Ketika pengguna menunjukkan QR Code ke kamera tablet, sistem akan langsung memindai dan memproses QR Code tersebut pada sisi client (tablet). Setelah QR Code berhasil dibaca, data yang terenkripsi dalam QR Code dikirimkan ke Node.js server untuk proses validasi dengan Firebase. Jika validasi berhasil menunjukkan QR Code valid, server akan mengirimkan sinyal konfirmasi ke ESP32 untuk mengaktifkan sistem RVM dan secara bersamaan memperbarui tampilan pada display untuk menampilkan informasi pengguna.



Gambar 4.8 Sequence diagram deteksi botol

Gambar 4.8 memperlihatkan alur proses deteksi botol yang terjadi setelah pengguna berhasil melakukan autentikasi, baik menggunakan RFID maupun QR Code. Proses dimulai ketika pengguna memasukkan botol ke dalam RVM yang kemudian akan diambil gambar dari botol tersebut oleh ESP32-CAM. Gambar yang diambil kemudian dikirimkan ke Flask server untuk diproses deteksi. Flask server melakukan analisis gambar untuk memverifikasi apakah objek yang terdeteksi adalah botol atau bukan. Hasil analisis ini kemudian diteruskan ke Node.js server. Jika objek terdeteksi sebagai botol, Node.js server akan memperbarui data poin pengguna di Firebase dan mengirimkan update secara *real-time* ke display melalui koneksi WebSocket untuk menampilkan perubahan poin pengguna.

D. Database

Pada iterasi kedua pengembangan sistem Sampahmas, *hacker* merancang dan membangun skema database berdasarkan hasil dari *Hipster*. Database yang digunakan adalah Firebase Realtime Database sebagai solusi penyimpanan data karena kemampuannya untuk menyediakan sinkronisasi data secara *real-time* dan skalabilitas. Data yang disimpan meliputi informasi pengguna, transaksi penukaran, serta jumlah poin yang diperoleh oleh pengguna.

Pada sistem SampahMas, skema database dirancang untuk mengelola entitas yang berkaitan dengan pengguna, produk, pesanan, transaksi penukaran, dan lokasi mesin vending. Setiap entitas disusun dalam bentuk koleksi yang memudahkan

penyimpanan dan akses data, sesuai dengan kebutuhan aplikasi yang telah dievaluasi oleh tim *Hipster* dan *Hustler*.

```
"users": {
  "userId": {
    "alamat": "String",
    "balance": "Number",
    "botolTerkumpul": "Number",
    "createdAt": "Timestamp",
    "email": "String",
    "imageProfile": "String",
    "name": "String",
    "phone": "String",
    "points": "Number",
    "rfid": "String",
    "role": "String",
    "userId": "String",
    "updatedAt": "Timestamp"
  }
}
```

Koleksi *users* dirancang untuk menyimpan informasi lengkap pengguna termasuk data profil seperti nama, email, alamat, dan nomor telepon. Koleksi ini juga mencatat data terkait aktivitas pengguna seperti jumlah poin, saldo, jumlah botol yang terkumpul, serta timestamp untuk pencatatan waktu pembuatan dan pembaruan akun. Penambahan field *role* memungkinkan pengelolaan hak akses pengguna, sementara field *rfid* mendukung fitur autentikasi menggunakan kartu NFC.

```
"product": {
  "productId": {
    "description": "String",
    "productId": "String",
    "imageUrl": "String",
    "price": "Number",
    "title": "String"
  }
}
```

```
}
}
```

Koleksi *product* menyimpan informasi detail setiap item seperti judul, deskripsi, harga, dan URL gambar.

```
"wishlist": {
  "userId": {
    "productId": {
      "quantity": "Number"
    }
  }
}
```

Koleksi *wishlist* berfungsi menyimpan daftar keinginan pengguna terhadap produk yang tersedia dalam aplikasi.

```
"location": {
  "deviceId": {
    "description": "String",
    "geohash": "String",
    "latitude": "Number",
    "longitude": "Number",
    "name": "String"
  }
}
```

Koleksi *location* menyimpan data lokasi mesin vending termasuk koordinat geografis yang dilengkapi dengan geohash untuk optimasi query berbasis lokasi.

```
"transactionVending": {
  "transactionId": {
    "bottles": "Number",
    "createdAt": "Number",
    "idUser": "String",
    "method": "String",
    "pointsAdd": "Number",
    "status": "String",
```

```

        "transactionId": "String"
    }
}

```

Koleksi *transaction_vending* bertugas mengelola transaksi penukaran botol yang dilakukan pengguna di mesin vending. Setiap transaksi diidentifikasi dengan *transactionId* sebagai *primary key* dan data yang disimpan meliputi *vendingId* yang merujuk ke mesin vending tempat transaksi dilakukan, *userId* yang menunjukkan pengguna yang melakukan penukaran, jumlah poin yang diperoleh dari penukaran (*pointsEarned*), jumlah botol yang ditukarkan (*bottles*), serta waktu transaksi terjadi (*created_at*).

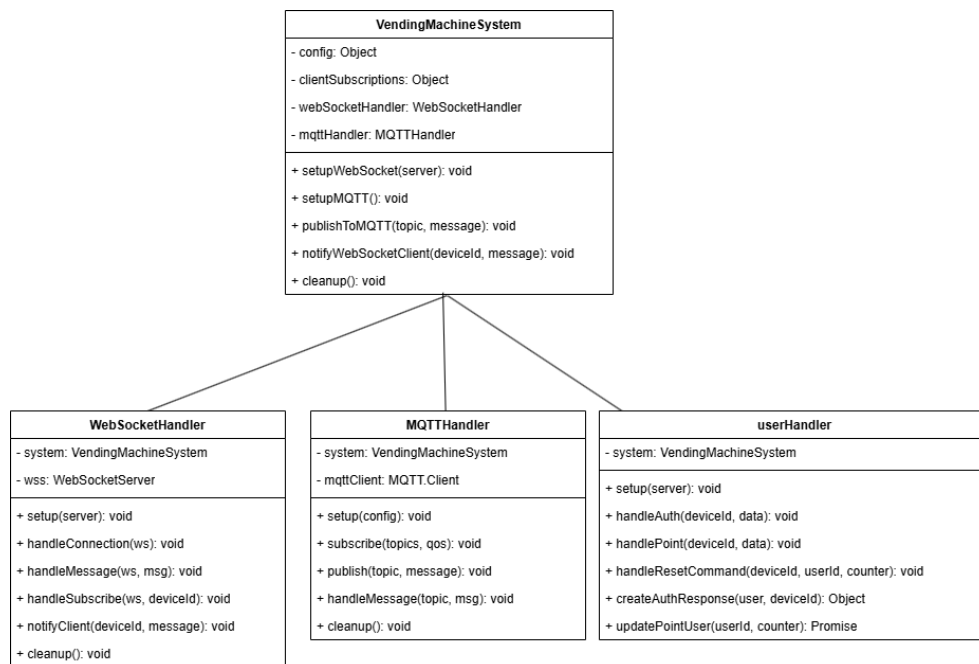
```

"transactions": {
  "orderId": {
    "amount": "Number",
    "orderId": "String",
    "status": "String",
    "transactionToken": "String"
  }
}

```

Transaksi dalam sistem dicatat melalui dua koleksi utama yaitu *transactionVending* untuk mencatat aktivitas penukaran botol di mesin vending, dan *transactions* untuk mencatat transaksi pembayaran. Koleksi *transactionVending* menyimpan detail seperti jumlah botol, metode autentikasi, dan poin yang ditambahkan, sedangkan koleksi *transactions* mencatat informasi pembayaran termasuk token transaksi untuk integrasi dengan sistem pembayaran.

E. Class Diagram



Gambar 4.9 Class Diagram Sistem Sampahmas

Kelas utama dalam sistem ini yaitu *VendingMachineSystem* berfungsi sebagai inti dari pengelolaan seluruh proses komunikasi dan pengaturan sistem vending machine. Kelas ini dirancang untuk menyimpan konfigurasi sistem melalui atribut *config* yang mencakup pengaturan koneksi dan parameter penting lainnya. Selain itu, atribut *clientSubscriptions* digunakan untuk mendaftarkan klien vending yang berlangganan pada topik tertentu agar sistem dapat mengelola komunikasi dengan lebih terstruktur. Kelas ini juga memiliki dua handler utama yaitu *webSocketHandler* dan *mqttHandler* yang masing-masing bertanggung jawab untuk menangani komunikasi menggunakan protokol *WebSocket* dan *MQTT*.

Dalam operasinya, kelas ini menyediakan beberapa metode yang mendukung fungsionalitas inti sistem. Metode *setupWebSocket* bertugas menginisialisasi server *WebSocket* agar sistem dapat berkomunikasi secara *real-time* dengan pengguna. Metode *setupMQTT* digunakan untuk mengatur koneksi dan konfigurasi protokol *MQTT* yang digunakan untuk komunikasi dengan perangkat berbasis *Internet of Things (IoT)*. Untuk mengirim pesan, kelas ini menggunakan metode *publishToMQTT*. Dengan metode ini, server dapat mengirim pesan ke topik tertentu dalam sistem *MQTT*. Selain itu, terdapat metode *notifyWebSocketClient* yang

digunakan untuk memberikan notifikasi kepada klien tertentu berdasarkan identitas perangkatnya atau *deviceId* yang sebelumnya sudah terdaftar di *clientSubscriptions*

Kelas *WebSocketHandler* bertanggung jawab untuk menangani semua komunikasi melalui *WebSocket*. Kelas ini memiliki atribut *wss* sebagai *server WebSocket* yang aktif dan metode seperti *setup* untuk mengatur *server WebSocket*, *handleConnection* untuk menangani koneksi baru, *handleMessage* untuk memproses pesan dari klien, serta *notifyClient* untuk mengirim notifikasi khusus ke klien tertentu. Selain itu, metode *cleanup* digunakan untuk membersihkan resource terkait *WebSocket*.

WebSocketHandler berfungsi dalam mengatasi keterbatasan teknis perangkat ESP32 yang tidak dapat berkomunikasi langsung dengan layar LCD. Handler ini memfasilitasi transmisi data secara instan ke antarmuka pengguna melalui protokol *WebSocket*, sehingga informasi dapat dengan lancar terupdate secara *real-time* pada layar vending.

Kelas *MQTTHandler* dirancang untuk komunikasi berbasis protokol MQTT. Dengan atribut *mqttClient*, kelas ini dapat mengelola koneksi, berlangganan topik tertentu melalui metode *subscribe*, mengirim pesan dengan *publish*, menangani pesan masuk melalui *handleMessage* dan membersihkan *resource* dengan *cleanup*.

Kelas *userHandler* berfokus pada pengelolaan data pengguna. Kelas ini mencakup metode untuk menangani autentikasi (*handleAuth*), mengelola poin pengguna (*handlePoint*), dan menangani perintah reset (*handleResetCommand*). Selain itu, terdapat metode *createAuthResponse* untuk membuat respons autentikasi dan *updatePointUser* untuk memperbarui poin pengguna berdasarkan parameter yang diberikan.

F. Desain REST API

Implementasi desain sistem SAMPAHMAS mengadopsi arsitektur REST API dengan NodeJS sebagai *server side* yang membagi endpoint ke dalam beberapa kelompok fungsional utama. Struktur routing API dibagi menjadi lima kategori utama yaitu autentikasi (*/api/auth*), manajemen pengguna (*/api/user*), pengelolaan item (*/api/item*), kontrol vending machine (*/api/vending*), dan pengelolaan transaksi

(/api/transaction). Setiap kategori memiliki fungsi spesifik yang menangani berbagai operasi sistem sesuai dengan URI masing-masing.

Endpoint	Method
/api/auth/register	POST
/api/auth/login	POST
/api/auth/logout	POST
/api/auth/refresh-token	POST

Endpoint register menangani pendaftaran pengguna baru dengan validasi data dan enkripsi password. Endpoint login memverifikasi kredensial dan menghasilkan JWT (*JSON Web Token*) untuk akses sistem yang di *generate* oleh firebase. Endpoint logout menangani proses pembersihan sesi, sementara refresh-token memperbarui token akses yang sudah mendekati masa habis.

Endpoint	Method
/api/user	GET
/api/user/:userId	PUT
/api/user/:userId	DELETE

Endpoint pengguna (/api/user) menangani operasi-operasi terkait data pengguna seperti pembuatan dan pembaruan profil serta pengambilan informasi pengguna.

Endpoint	Method
/api/product	GET

/api/product/:itemId	GET
----------------------	-----

Method GET pada endpoint (/api/item) digunakan untuk menampilkan keseluruhan daftar produk yang tersedia dalam marketplace. Untuk melihat detail spesifik produk, sistem menggunakan endpoint (/api/item/:itemId) dengan method GET dengan :itemId merupakan identifier unik dari produk yang ingin ditampilkan. Penambahan produk baru dilakukan melalui method POST pada endpoint /api/item.

Endpoint	Method
/api/vending/location/radius	GET
/api/vending/location	GET
/api/vending/location/:locationId	GET

Implementasi endpoint pada modul lokasi vending machine SAMPAHMAS mencakup beberapa fungsi utama yang dioperasikan melalui endpoint /api/vending/location. *Method* GET pada endpoint /api/vending/location/radius digunakan untuk mencari lokasi mesin vending terdekat.

Untuk akses data lokasi secara keseluruhan, sistem menggunakan endpoint /api/vending/location dengan method GET yang menampilkan seluruh daftar lokasi mesin, sementara penambahan lokasi baru dilakukan melalui method POST pada endpoint yang sama dengan menyertakan data seperti alamat dan koordinat GPS.

Untuk mendapatkan lokasi spesifik dilakukan melalui endpoint /api/vending/location/:locationId yang mendukung method GET untuk melihat detail lokasi. Setiap endpoint ini dilengkapi dengan validasi akses untuk memastikan keamanan data.

Endpoint	Method
----------	--------

/api/transaction/create-payment	POST
/api/transaction/notification	POST

Implementasi sistem pembayaran pada aplikasi menggunakan dua endpoint utama dalam transaksi. Endpoint (/api/transaction/create-payment) dengan metode POST menangani inisiasi pembayaran. Kemudian menghasilkan token transaksi melalui integrasi dengan *Midtrans*. Sementara itu, endpoint (/api/transaction/notification) dengan metode POST yang berfungsi menerima dan memproses notifikasi status pembayaran dari *Midtrans* atau sering dikenal *webhook*. Endpoint untuk berguna untuk memastikan pembaruan status transaksi secara *real-time* dalam sistem.

4.2.3 Development

A. Pengembangan Fitur pada *Reverse Vending Machine*

Pada iterasi kedua dilakukan pengembangan *Reverse Vending Machine* (RVM). Hal ini melibatkan serangkaian tahapan mulai dari fabrikasi untuk menghasilkan prototipe fisik yang nanti diujikan sesuai dengan hasil rancangan gambar *Hipster*. Setelah konstruksi fisik selesai, pengujian awal dilakukan untuk memastikan seluruh komponen termasuk layar interaktif dan sensor yang berfungsi secara optimal sebelum memasuki tahap integrasi dengan sistem perangkat lunak.

Pengembangan sistem *Reverse Vending Machine* (RVM) adalah penambahan integrasi komunikasi antar perangkat dan juga penyimpanan database melalui server *backend* serta deteksi dan pengiriman gambar dengan ESP32-CAM dan deteksi menggunakan OpenCV.

B. Integrasi ESP32

Sistem ini menggunakan perangkat ESP32 dengan koneksi Wi-Fi untuk menghubungkan dan berkomunikasi dengan server. Setelah koneksi Wi-Fi berhasil, perangkat terhubung ke broker MQTT dengan menggunakan *library PubSubClient*. Dengan library ini perangkat ESP32 dapat melakukan subscribe dan publish ke topik tertentu dengan komunikasi MQTT. Berikut adalah kode untuk menghubungkan perangkat ke broker MQTT:

```

void connectMQTT() {
    client.setServer(mqtt_broker, mqtt_port);
    client.setCallback(mqttCallback);
    BearSSL::X509List serverTrustedCA(ca_cert);
    espClient.setTrustAnchors(&serverTrustedCA);

    while (!client.connected()) {
        String      client_id      =      "ESP32-client-"      +
String(WiFi.macAddress());

        if      (client.connect(client_id.c_str(),      mqtt_username,
mqtt_password)) {
            Serial.println("Connected to MQTT broker");
            subscribeToAuth();
            client.subscribe("vending/001/reset");
        } else {
            char err_buf[128];
            espClient.getLastSSLError(err_buf, sizeof(err_buf));
            delay(5000);
        }
    }
}

```

```

void authenticateRFID(String rfidTag) {
    if (WiFi.status() == WL_CONNECTED) {
        String payload = "{\"rfidTag\": \"" + rfidTag + "\"}";
        client.publish("vending/001/auth", payload.c_str());
        waitingForResponse = true;
        lastScanTime = millis();
    }
}

if (rfid.PICC_IsNewCardPresent() && rfid.PICC_ReadCardSerial() &&
(millis() - lastScanTime > DEBOUNCE_DELAY)) {

    String rfidTag = "";
    for (byte i = 0; i < rfid.uid.size; i++) {
        rfidTag += String(rfid.uid.uidByte[i], HEX);
    }
}

```

```

    }
    Serial.println("[RFID] Card detected: " + rfidTag);
    lastScanTime = millis();

    if (isAuthenticating) {
        authenticateRFID(rfidTag);
    }

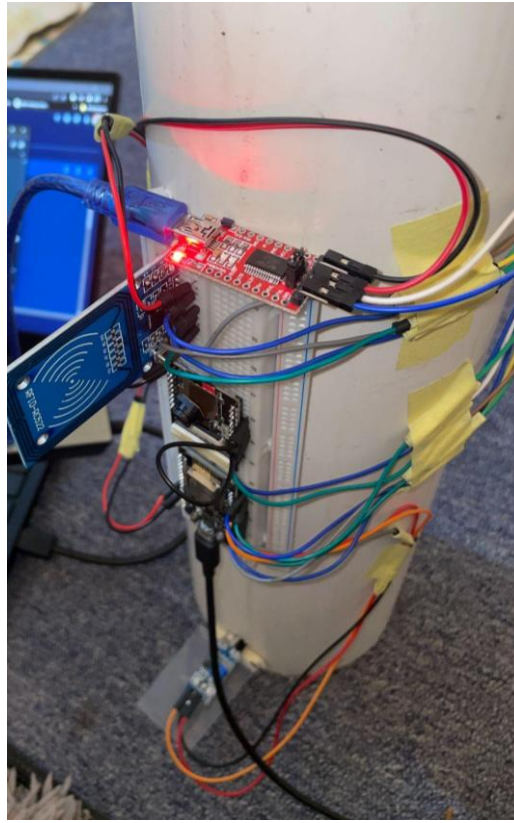
    rfid.PICC_HaltA();
    rfid.PCD_StopCrypto1();
}

if (waitingForResponse && (millis() - lastScanTime >
AUTH_TIMEOUT)) {
    Serial.println("[AUTH] Response timeout, resetting...");
    waitingForResponse = false;
    isAuthenticating = true;
    connectMQTT();
}

```

Fungsi *authenticateRFID()* bertugas mendeteksi dan mengautentikasi kartu RFID dengan menggunakan protokol komunikasi SPI (*Serial Peripheral Interface*) melalui pin SDA (21) dan RST (22) pada ESP32. Saat kartu RFID terdeteksi, data berupa Unique Identifier (UID) diambil, dikonversi ke format heksadesimal, dan disimpan dalam variabel *rfidTag*. Sistem kemudian memverifikasi koneksi WiFi untuk memastikan data dapat dikirim ke broker MQTT.

Proses autentikasi dilakukan melalui arsitektur publish-subscribe menggunakan protokol MQTT. Ketika kartu RFID terbaca, sistem mengirimkan data dalam format JSON yang berisi *rfidTag* ke topik "*vending/001/auth*". Respons autentikasi diterima melalui topik "*vending/001/authResult*" dan hasilnya diproses dalam callback MQTT. Jika autentikasi berhasil, sistem akan menyimpan ID pengguna dan mengaktifkan fungsi capture pada ESP32-CAM melalui komunikasi serial.



Gambar 4.10 Prototyping RVM

Prototipe ini menggunakan modul *RFID RC522* yang dikombinasikan dengan ESP32 untuk mendeteksi kartu. Modul *RFID RC522* digunakan untuk membaca identitas unik (UID) dari kartu *RFID* yang didekatkan ke area sensor. Ketika sebuah kartu *RFID* didekatkan, sinyal dari modul akan mengirimkan data UID ke ESP32 untuk diproses lebih lanjut.

C. Integrasi ESP32-CAM

Dalam iterasi ini, ESP32-CAM digunakan dalam proses validasi visual objek terutama untuk memastikan bahwa objek yang dimasukkan oleh pengguna sesuai dengan kriteria. ESP32-CAM dikonfigurasi untuk mengambil gambar setelah menerima sinyal dari ESP32 yang bertugas mendeteksi keberadaan objek. Gambar yang dihasilkan ESP32-CAM kemudian dikirim ke server *backend* menggunakan koneksi WiFi dan protokol HTTP untuk dilakukan proses identifikasi.

Proses pengiriman gambar ke server dilakukan dengan mengambil gambar menggunakan modul kamera dan kemudian mengirimkannya menggunakan metode HTTP POST. Sistem ESP32-CAM memanfaatkan koneksi WiFi dan SSL

untuk menjamin keamanan data yang dikirim. Setelah gambar dikirim, server *backend* melakukan proses identifikasi. Jika objek tidak memenuhi kriteria, sistem akan menolak transaksi.

D. Pengembangan fungsi dari tampilan antarmuka vending

Pada fase ini, implementasi antarmuka vending machine dilaksanakan sesuai dengan desain yang telah ditetapkan sebelumnya oleh *Hipster*. Dalam pengembangan halaman utama, terdapat dua aspek utama yang diperhatikan, yakni aspek visual dan fungsionalitas JavaScript.

```
<script>
    function savedeviceId() {
        const deviceId = "001"
        if (deviceId) {
            localStorage.setItem('deviceId', deviceId);
            console.log('deviceId disimpan:', deviceId);
        } else {
            console.error('deviceId tidak ditemukan di URL!');
        }
    }

    window.onload = function () {
        savedeviceId();
    };
</script>
```

Kode JavaScript yang ditambahkan bertujuan untuk menangani pengelolaan *deviceId*. Implementasi fungsi *savedeviceId()* dirancang untuk menyimpan identifikasi unik perangkat ke dalam *localStorage* pada browser. Langkah ini sangat penting dalam rangka pelacakan dan identifikasi mesin vending. Fungsi tersebut dipanggil secara otomatis pada saat pemuatan halaman melalui event *window.onload*.

```
function domReady(fn) {
```

```

        if (document.readyState === "complete" ||
document.readyState === "interactive") {
            setTimeout(fn, 1000);
        } else {
            document.addEventListener("DOMContentLoaded",
fn);
        }
    }

    domReady(() => connectWebSocket());

    socket.onopen = () => {
        startQRCodeScanner();
    };

    socket.onerror = (error) => {
        console.error('WebSocket error:', error);
        document.getElementById('my').innerText =
'WebSocket connection error.';
    };

    socket.onclose = () => {
        console.log('WebSocket connection closed.
Reconnecting...');
        setTimeout(connectWebSocket, 5000);
    };

```

Pengembangan fitur pada halaman pemindai QR code melibatkan berbagai komponen utama yang terintegrasi. Proses dimulai dengan fungsi `domReady`, yang memastikan bahwa DOM telah sepenuhnya dimuat sebelum memulai koneksi *WebSocket* dengan server. Fungsi ini disertai penundaan selama 1,5 detik untuk stabilitas sistem.

Setelah halaman siap, koneksi *WebSocket* ke server diinisiasi dan dimulai dengan pengambilan *deviceId* dari penyimpanan lokal sebagai identitas unik *Reverse Vending Machine*. Koneksi *WebSocket* ini dirancang untuk menangani berbagai peristiwa, seperti pembukaan koneksi, penerimaan pesan, penanganan

error, dan pemutusan koneksi. Jika koneksi terputus, sistem akan otomatis mencoba menyambung kembali setiap 5 detik guna menjaga layanan tetap berjalan.

```
function startQRCodeScanner() {
    const qrCodeScanner = new Html5Qrcode("my-qr-reader");

    qrCodeScanner.start(
        { facingMode: "user" },
        {
            fps: 8,
            qrbox: 250
        },
        (decodedText) => {
            console.log(`Scanned QR code: ${decodedText}`);
            if (socket && socket.readyState ===
WebSocket.OPEN) {
                socket.send(JSON.stringify({
                    action: 'auth',
                    uid: decodedText.trim()
                }));
                document.getElementById("info").innerText =
'QR code scanned, awaiting server response.';
            } else {
                console.error("WebSocket is not
connected.");
            }
        }
    ).catch((err) => console.error("QR scanner error:",
err));
}
```

Fitur pemindaian QR code menggunakan *library Html5Qrcode* yang dikonfigurasi secara optimal untuk vending machine. Pemindai difungsikan dengan menggunakan kamera depan dengan kecepatan 8 fps dan area pemindaian berukuran 250x250 piksel. Ketika QR code terdeteksi, hasil pemindaian langsung dikirim ke server melalui koneksi *WebSocket* yang sudah terhubung.

```

socket.onmessage = (event) => {
    const message = JSON.parse(event.data);

    if (message.action === 'auth') {
        if (message.result === 'Authorized' &&
message.status) {
            localStorage.setItem('userName',
message.name);
            localStorage.setItem('userId',
message.idUser);
            localStorage.setItem('accessToken',
message.accessToken);
            localStorage.setItem('totalPoints',
message.points);

            document.getElementById("info").innerText
= 'Authorized! Redirecting...';
            setTimeout(() => {
                window.location.href = "pointws.html";
            }, 1000);
        } else {
            document.getElementById("info").innerText
= 'Authorization failed.';
        }
    } else {
        console.warn('Unhandled WebSocket message:',
message);
    }
};

```

Autentikasi dilakukan dengan pertukaran pesan berbasis JSON dengan server. Jika berhasil, sistem akan menyimpan informasi pengguna seperti nama, ID, token akses, dan jumlah poin ke penyimpanan lokal. Setelah data disimpan, pengguna diarahkan ke halaman pointws.html untuk melanjutkan transaksi.

```

function connectWebSocket() {
    const socket = new WebSocket('');
    const deviceId = getDeviceId();

    if (!deviceId) {
        console.error('No deviceId found, unable to connect to
WebSocket.');
```

WebSocket.');

```

        return;
    }

    socket.onopen = function () {
        socket.send(JSON.stringify({
            deviceId: deviceId,
            action: 'subscribe'
        }));
    };

    socket.onmessage = function (event) {
        try {
            const message = JSON.parse(event.data);
            if (message && message.result === 'Authorized' &&
message.status === true && message.deviceId === deviceId) {

                const result = message.result;
                const status = message.status;
                const accessToken = message.accessToken;
                const idUser = message.idUser;
                const name = message.name;
                const points = message.points;

                localStorage.setItem('userName', name);
                localStorage.setItem('userId', idUser);
                localStorage.setItem('accessToken',
accessToken);

                localStorage.setItem('totalPoints', points);

                if (status) {
                    setTimeout(() => {
                        window.location.href = "pointws.html";
                    }, 1000);
                }
            }
        }
    };
}

```

```

    }
};

```

Implementasi utama berpusat pada fungsi *connectWebSocket* dan bertugas menginisiasi koneksi ke server. Setelah koneksi berhasil dibangun, sistem segera mengirimkan pesan subscribe dengan *deviceId* untuk mendaftarkan *vending machine* ke server. Koneksi ini dirancang untuk menangani berbagai jenis peristiwa komunikasi dengan server secara *real-time*.

Pada proses penerimaan pesan (onmessage), sistem menjalankan validasi dan pengolahan data yang diterima. Pada halaman ini, autentikasi menunggu pesan dari server yang dikirim oleh perangkat IoT setelah memindai kartu NFC RFID pada reader. Ketika pesan autentikasi diterima, sistem memeriksa kecocokan *deviceId* dan status otorisasi yang dikirim oleh server. Apabila autentikasi berhasil, data penting pengguna seperti nama, ID, token akses, dan jumlah poin disimpan ke dalam localStorage. Setelah data tersimpan, pengguna akan diarahkan ke halaman *pointws.html* secara otomatis.

```

function connectWebSocket() {
    if (socket && socket.readyState !== WebSocket.CLOSED) {
        return;
    }

    socket.onopen = function () {
        const deviceId = localStorage.getItem('deviceId')
|| "unknown";

        const userId = localStorage.getItem('userId') ||
"unknown";

        socket.send(JSON.stringify({ deviceId: deviceId,
action: 'subscribe', userId: userId}));

        pingInterval = setInterval(() => {
            if (isSocketConnected()) {
                socket.send(JSON.stringify({ action: 'ping'
})));

```

```

        }
        }, 30000);
    };

    socket.onmessage = function (event) {
        try {
            const data = JSON.parse(event.data);
            const name = data.name ||
localStorage.getItem('userName') || "Unknown User";
            const counter = parseInt(data.counter ||
localStorage.getItem('counter') || 0);
            const totalPoints = parseInt(data.points ||
localStorage.getItem('totalPoints') || 0);

            localStorage.setItem('userName', name);
            localStorage.setItem('counter', counter);
            localStorage.setItem('totalPoints',
totalPoints);

            document.getElementById("nama").innerText =
name;
            document.getElementById("counter").innerText =
`+ ${counter}`;
            document.getElementById("points").innerText =
`Current Points : ${totalPoints}`;

        } catch (error) {
            console.error("Error parsing WebSocket
message:", error);
        }
    };

    socket.onclose = function () {
        if (pingInterval) {
            clearInterval(pingInterval);
        }

        setTimeout(function () {
            connectWebSocket();
        }, 1000);
    };

```

```

        socket.onerror = function (error) {
            console.error('WebSocket error:', error);
        };
    }

```

Fungsi `connectWebSocket` pada halaman `pointws.html` berperan sebagai komponen utama yang menangani konektivitas dengan server. Fungsi ini melakukan pengecekan status koneksi yang ada sebelum membuat koneksi baru ke server WebSocket. Setelah koneksi terbentuk, sistem mengirimkan pesan `subscribe` yang berisi `deviceId` dan `userId`. Untuk menjaga koneksi tetap aktif, sistem mengirimkan ping ke server setiap 30 detik.

Dalam penanganan pesan (*onmessage*), sistem menerima dan memproses data dari server yang mencakup nama pengguna, jumlah counter (botol) dan total poin. Data ini kemudian disimpan ke `localStorage` dan ditampilkan di antarmuka pengguna melalui pembaruan elemen-elemen DOM yang sesuai.

```

function resetAndGoHome() {
    const deviceId = localStorage.getItem('deviceId');
    const userId = localStorage.getItem('userId');
    const counter =
parseInt(localStorage.getItem('counter')) || 0;

    if (socket && socket.readyState === WebSocket.OPEN)
    {
        socket.send(JSON.stringify({
            deviceId: deviceId,
            userId: userId,
            counter: counter,
            action: 'reset'
        }));
    } else {
        console.error('WebSocket is not open for reset
action');
    }
}

```

```

        localStorage.clear();
        if (deviceId) {
            localStorage.setItem('deviceId', deviceId);
        }
        setTimeout(() => {
            window.location.href = "success.html";
        }, 1000);
    }

```

Fungsi `resetAndGoHome` digunakan untuk menangani proses selesai berfungsi sebagai pesan reset ke server dengan informasi `deviceId`, `userId` dan `counter`. Setelah pengiriman berhasil, sistem akan membersihkan `localStorage` dan mengarahkan pengguna ke halaman sukses setelah jeda satu detik.

E. Pengembangan server

Pada iterasi ini, API Gateway dirancang untuk mengelola komunikasi antar berbagai komponen sistem aplikasi. Pengembangan API Gateway ini menggunakan `Express.js` sebagai kerangka kerja utama dalam membangun server berbasis `Node.js`. API Gateway bertindak sebagai pusat pengelolaan dan distribusi data yang mencakup autentikasi pengguna, manajemen insentif, serta komunikasi `WebSocket` dan `MQTT` untuk mendukung operasi vending machine.

```

    setupRoutes() {
        this.app.use('/api/auth', authRoutes);
        this.app.use('/api/user', authMiddleware, usersRoutes);
        this.app.use('/api/product', authMiddleware,
upload.single("image"), productRoutes);
        this.app.use('/api/vending', authMiddleware,
vendingRoutes);
        this.app.use('/api/transaction', authMiddleware,
transactionRoutes);
        this.app.use('/api/wishlist', authMiddleware,
wishlistRoutes);

        this.app.get('/api', (req, res) => {

```

```

        res.json({
            success: true,
            statusCode: 200,
            message: Selamat datang di SAMPAHMAS API!
        });
    });

    this.app.use((req, res) => {
        res.status(404).json({
            success: false,
            message: 'API rute tidak ditemukan,
        });
    });
}

```

Server API dibangun dengan Express.js untuk mengelola rute dan logika bisnis utama. Rute-rute API ini mencakup autentikasi pengguna, manajemen item, manajemen insentif poin, serta rute vending machine. Setiap rute ditangani oleh controller yang terpisah.

```

import express from 'express';
import authController from '../auth/auth.controller.js';

const router = express.Router();

router.post('/register', authController.register);
router.post('/login', authController.login);
router.post('/logout', authController.logout);
router.post('/refresh-token', authController.refreshToken);

export default router;

```

Pada implementasi server API ini, sistem dirancang untuk menangani autentikasi pengguna dengan memanfaatkan kombinasi antara Firebase Authentication. Terdapat beberapa rute utama yang dikelola dalam AuthController, yaitu rute register, login, dan logout. Masing-masing rute ini dirancang untuk memenuhi berbagai kebutuhan autentikasi dan otorisasi pengguna dalam aplikasi.

```

    async register(email, password, username, phone) {
        const auth = getAuth(firebaseClientApp);
        try {
            const userCredential = await
createUserWithEmailAndPassword(auth, email, password);
            await userRepository.addUser(userCredential.user.uid,
username, email, phone);
            return userCredential.user;
        } catch (error) {
            console.error('Register Error:', error);
            throw new Error(error.message);
        }
    }
}

```

Fungsi pertama adalah register yang berguna untuk pengguna baru yang ingin mendaftarkan diri. Pada rute ini, *authService.registerUser()* dipanggil untuk memproses data yang diterima dari permintaan HTTP. Layanan ini bertugas melakukan validasi data pengguna, menyimpan data ke dalam database dan menghasilkan token akses (JWT) serta refresh token.

Fungsi ini memanfaatkan *Firebase Authentication* dengan menggunakan metode bawaan *createUserWithEmailAndPassword* yang memungkinkan pembuatan akun baru dengan email dan kata sandi. *Firebase Authentication* bertanggung jawab menangani logika validasi data, pembuatan pengguna, serta penyimpanan data autentikasi dasar. Setelah pengguna berhasil didaftarkan, data tambahan seperti username dan phone akan disimpan ke dalam database aplikasi menggunakan *userRepository.addUser()*.

Apabila proses pendaftaran berhasil, server akan mengirimkan respon dengan status 201 dan mengirimkan kembali data pengguna beserta token akses yang dihasilkan. Akan tetapi, jika terjadi kesalahan sistem akan mengembalikan status 400 dengan pesan kesalahan yang menjelaskan penyebab gagalnya proses pendaftaran tersebut.

```

    async login(email, password) {
        const auth = getAuth(firebaseClientApp);

```

```

        try {
            const userCredential = await
signInWithEmailAndPassword(auth, email, password);
            return userCredential.user;
        } catch (error) {
            console.error('Login Error:', error);
            throw new Error(error.message);
        }
    }
}

```

Fungsi kedua adalah login yang berperan sebagai proses verifikasi identitas pengguna. Rute login menerima data email dan kata sandi melalui *body request*. Kemudian, *authService.loginUser()* memverifikasi data tersebut menggunakan layanan *Firebase Authentication* melalui metode *signInWithEmailAndPassword*. Dengan memanfaatkan metode ini, *Firebase* menangani proses autentikasi berdasarkan email dan kata sandi yang telah didaftarkan sebelumnya.

Jika proses autentikasi berhasil, *Firebase* akan mengembalikan objek pengguna beserta token autentikasi (*firebaseAccessToken* dan *firebaseRefreshToken*) yang kemudian dapat digunakan untuk proses otorisasi lebih lanjut dalam aplikasi. Semua token ini digunakan untuk otorisasi dan sesi pengguna tetap aman dan valid. Jika login gagal, server akan mengembalikan status 400 dengan pesan "*Invalid email or password*".

```

async logout() {
    const auth = getAuth(firebaseClientApp);
    try {
        await signOut(auth);
        return true;
    } catch (error) {
        console.error('Logout Error:', error);
        throw new Error(error.message);
    }
}

```

Fungsi ketiga adalah *logout* yang digunakan untuk mengakhiri sesi user. Pada rute ini, *authService.logoutUser()* akan memanggil metode bawaan dari *Firebase*

Authentication yaitu *signOut*. Metode *signOut* ini berfungsi untuk mengakhiri sesi pengguna dan menghapus status autentikasi di sisi klien. *Firebase Authentication* menangani pengelolaan sesi pengguna secara aman dan memastikan bahwa setelah pengguna *logout*, sesi dan token yang terkait menjadi tidak valid.

Fungsi-fungsi autentikasi ini didasarkan pada metode yang telah disediakan oleh *Firebase Authentication* yang memungkinkan untuk menangani pendaftaran, login, dan logout pengguna dengan lebih mudah dan efisien.

```
router.get('/', userController.getUser);
router.put('/:userId', userController.updateUser);
```

Pada rute user, endpoint GET '/' diimplementasikan melalui *userController.getUser* untuk mengambil informasi profil pengguna berdasarkan token autentikasi, sementara endpoint PUT('/:userId' dengan *userController.updateUser* digunakan untuk pembaruan data profil pengguna.

```
router.get('/', productController.getAllProducts);
router.get('/:productId', productController.getProductById);
```

Endpoint GET '/' yang ditangani oleh *productController.getAllProducts* bertugas mengambil seluruh daftar produk yang tersedia dalam sistem marketplace, termasuk informasi detail seperti nama, harga, dan deskripsi. Sedangkan endpoint GET('/:productId' dengan *productController.getProductById* memungkinkan akses ke informasi spesifik suatu produk berdasarkan ID yang diberikan.

```
router.get('/location/radius',
VendingController.getLocationsInRadius);
router.get('/location', VendingController.getAllLocations);
```

Pengelolaan lokasi vending machine diimplementasikan melalui dua endpoint dalam *VendingController*. Endpoint GET '/location/radius' dengan fungsi *getLocationsInRadius* memungkinkan pencarian lokasi mesin berdasarkan radius tertentu dari posisi pengguna dengan memanfaatkan perhitungan geolokasi untuk

memberikan hasil yang optimal. Endpoint GET '/location' dengan *getAllLocations* menyediakan akses ke daftar lokasi seluruh mesin vending yang terdaftar dalam sistem.

```
router.post('/', wishlistController.addToWishlist);  
router.get('/', wishlistController.getWishlistByUser);  
router.delete('/:wishlistId',  
wishlistController.removeFromWishlist);
```

Manajemen wishlist pengguna ditangani melalui tiga endpoint utama. Endpoint POST '/' dengan *wishlistController.addToWishlist* memungkinkan pengguna menambahkan produk ke dalam daftar keinginan pengguna. Endpoint GET '/:userId' yang diimplementasikan melalui *getWishlistByUser* berfungsi mengambil seluruh daftar wishlist untuk pengguna tertentu. Terakhir, endpoint DELETE '/:wishlistId' dengan *removeFromWishlist* memungkinkan pengguna menghapus item dari wishlist mereka.

```
router.get('/', transactionController.getAllTransactions);  
router.post('/create-payment',  
transactionController.createPayment);  
router.post('/notification',  
transactionController.handleNotification);
```

Implementasi endpoint transaksi pada API Gateway mencakup tiga fungsi utama untuk mengelola proses pembayaran dan riwayat transaksi. Endpoint GET '/' yang diimplementasikan melalui *transactionController.getAllTransactions* bertugas mengambil seluruh riwayat transaksi pengguna, termasuk detail seperti jumlah pembayaran, status transaksi, dan waktu pembuatan.

Proses pembayaran diinisiasi melalui endpoint POST '/create-payment' yang ditangani oleh *transactionController.createPayment*. Endpoint ini bertanggung jawab untuk membuat sesi pembayaran baru dengan mengintegrasikan layanan Midtrans. Implementasi mencakup pembuatan token transaksi, penghitungan total pembayaran, dan pengiriman data pembayaran ke sistem Midtrans. Setelah

berhasil, endpoint mengembalikan informasi yang diperlukan untuk melanjutkan proses pembayaran di sisi client.

Endpoint POST '/notification' dengan fungsi `handleNotification` berperan sebagai webhook untuk menerima notifikasi status pembayaran dari Midtrans. Ketika pembayaran berhasil atau gagal, Midtrans mengirimkan notifikasi ke endpoint ini, yang kemudian diproses oleh sistem untuk memperbarui status transaksi di database.

Sistem Reverse Vending Machine merupakan komponen inti dalam arsitektur server backend. Reverse Vending Machine yang dirancang dengan pendekatan berorientasi objek. Pada implementasi Reverse Vending Machine (RVM), sistem server backend mengadopsi beberapa handler yang dirancang untuk mengelola komunikasi dengan perangkat IoT. Arsitektur ini dirancang untuk menangani sistem secara efisien dan terorganisir melalui class `VendingMachineSystem`, yang berperan sebagai pengendali utama dalam mengkoordinasikan interaksi antar komponen pada sistem Reverse Vending Machine.

```

constructor(config) {
    this.config = {
        mqttBrokerUrl: config.mqttBrokerUrl,
        mqttUsername: config.mqttUsername,
        mqttPassword: config.mqttPassword,
        topics: ['vending/+/auth', 'vending/+/point', 'vending/+/reset',
'vending/+/register'],
        qos: 2,
    };
    this.clientSubscriptions = {};
    this.userHandler = new UserHandler(this);
    this.websocketHandler = new WebSocketHandler(this);
    this.mqttHandler = new MQTTHandler(this);
}

```

VendingMachineSystem diimplementasikan sebagai kelas utama yang menginisialisasi dan mengatur interaksi antar handler. Sistem ini dikonfigurasi dengan parameter-parameter yang dibutuhkan.

```

    setupWebSocket(server) {
        this.webSocketHandler.setup(server);
    }

    setupMQTT() {
        this.mqttHandler.setup(this.config);
    }

    publishToMQTT(topic, message) {
        this.mqttHandler.publish(topic, message);
    }

    notifyWebSocketClient(deviceId, message) {
        this.webSocketHandler.notifyClient(deviceId, message);
    }

```

Sistem ini menggunakan beberapa *method* untuk pengaturan dan komunikasi. Metode *setupWebSocket(server)* digunakan untuk menginisialisasi *WebSocketHandler* dengan server HTTP yang diberikan agar interaksi dengan antarmuka pengguna berlangsung secara *real-time*.

Metode *setupMQTT()* bertanggung jawab untuk mengkonfigurasi koneksi MQTT sesuai dengan pengaturan yang telah ditentukan, memastikan komunikasi dengan perangkat IoT berjalan lancar. Fungsi *publishToMQTT(topic, message)* menyediakan mekanisme untuk mengirim pesan ke broker MQTT, sehingga mempermudah pertukaran data dengan perangkat vending machine. Sedangkan *notifyWebSocketClient(deviceId, message)* digunakan untuk pengiriman notifikasi langsung ke klien *WebSocket* tertentu berdasarkan *deviceId*.

```

    async handleAuth(deviceId, data) {

```

```

let user;
if (data.rfidTag) {
  user = await userRepository.getUserByRFID(data.rfidTag);
} else if (data.uid) {
  user = await userRepository.getUserById(data.uid);
}
const authResult = this.createAuthResponse(user, deviceId);
const authResponseTopic = `vending/${deviceId}/authResult`;

this.system.publishToMQTT(authResponseTopic, authResult);
this.system.notifyWebSocketClient(deviceId, authResult);
}

```

Method *handleAuth* dalam *UserHandler* mengimplementasikan mekanisme autentikasi yang fleksibel dengan mendukung dua metode identifikasi yaitu melalui *RFID* atau *user ID* dengan menggunakan *QR Code*. Proses autentikasi dimulai dengan memeriksa kredensial yang diterima dari perangkat RVM. Jika data berisi data *RFID*, sistem akan mencari pengguna menggunakan metode *getUserByRFID* dari *userRepository* dan jika data berisi *userID*, maka sistem akan menggunakan *getUserById*. Setelah proses pencarian pengguna selesai, sistem membuat respons autentikasi menggunakan metode *createAuthResponse*. Hasil autentikasi kemudian dikirimkan melalui dua jalur komunikasi yakni MQTT untuk komunikasi dengan perangkat IoT dan WebSocket untuk pembaruan tampilan pengguna secara *real-time*.

```

async handlePoint(deviceId, data) {
  try {
    if (!data.userId) {
      throw new Error('User ID is required for point action');
    }
    if (typeof data.counter !== 'number' || isNaN(data.counter)) {
      throw new Error('Counter must be a valid number');
    }
  }
}

```

```

        const user = await userRepository.getUserById(data.userId);
        if (!user) {
            throw new Error('User not found');
        }

        const lastCounter = user.lastCounter || 0;
        const newBottles = data.counter - lastCounter;
        const pointsToAdd = newBottles * 100;
        const updatedPoints = user.points + pointsToAdd;

        await Promise.all([
            usersRepository.updateLastCounter(data.userId, data.counter),
        ]);

        const pointUpdateMessage = {
            action: 'point',
            deviceId,
            status: true,
            idUser: data.userId,
            counter: data.counter,
            pointsAdded: pointsToAdd,
            totalPoints: updatedPoints,
        };

        this.system.notifyWebSocketClient(deviceId, pointUpdateMessage);
        this.system.publishToMQTT(`vending/${deviceId}/pointResult`,
pointUpdateMessage);

    } catch (error) {
        console.error(`Error handling points for device ${deviceId}:`,
error.message);
        this.system.notifyWebSocketClient(deviceId, {
            action: 'point',
            result: 'Error',
            status: false,
            message: error.message,
        });
    }
}

```

Pengelolaan poin pengguna ditangani oleh metode *handlePoint* untuk perhitungan dan pembaruan poin berdasarkan jumlah botol yang dimasukkan dan dikumpulkan. Metode ini melewati validasi terhadap data yang diterima. Setelah validasi berhasil, sistem mengambil data pengguna dan melakukan perhitungan penambahan poin. Setiap botol yang terkumpul dikonversi menjadi poin dan total poin diperbarui dalam database. Hasil pembaruan poin juga akan dikembalikan ke sistem melalui WebSocket dan MQTT.

```
createAuthResponse(user, deviceId) {  
  if (user) {  
    return {  
      action: 'auth',  
      deviceId: deviceId,  
      result: 'Authorized',  
      status: true,  
      idUser: user.uid,  
      email: user.email,  
      phone: user.phone,  
      name: user.name,  
      rfid: user.rfid,  
      points: user.points,  
    };  
  }  
  return {  
    action: 'auth',  
    deviceId: deviceId,  
    result: 'Not Authorized',  
    status: false,  
  };  
}
```

Metode *createAuthResponse* digunakan dalam standarisasi format respons autentikasi. Untuk pengguna yang terautentikasi, metode ini menghasilkan objek respons yang kaya informasi, mencakup status autentikasi, deviceId, dan data pengguna seperti email, nomor telepon, nama, dan jumlah poin. Untuk kasus autentikasi yang gagal, respons yang dihasilkan tetap konsisten dalam format

namun hanya berisi informasi minimal yang diperlukan untuk menangani kegagalan autentikasi.

```
async handleResetCommand(deviceId, userId) {
  try {
    const user = await userRepository.getUserById(userId);
    if (!user) throw new Error('User not found');

    const lastCounter = user.lastCounter || 0;
    let botolTerkumpul = user.botolTerkumpul;
    const newBottles = lastCounter;
    botolTerkumpul += newBottles;

    if (newBottles < 0) {
      throw new Error('Invalid counter value ');
    }

    const pointsToAdd = newBottles * 100;
    const updatedPoints = user.points + pointsToAdd;

    await Promise.all([
      userRepository.updateBottle(userId, botolTerkumpul),
      userRepository.updatePoints(userId, updatedPoints),
      userRepository.updateLastCounter(userId, 0),
    ]);

    const resetMessage = {
      action: 'reset',
      deviceId,
      status: true,
      idUser: userId,
      pointsAdded: pointsToAdd,
      totalPoints: updatedPoints,
      message: 'Reset completed successfully',
    };

    this.system.notifyWebSocketClient(deviceId, resetMessage);
    this.system.publishToMQTT(`vending/${deviceId}/reset`,
resetMessage);
```

```

    } catch (error) {
        console.error('Error handling reset command:', error);
        this.system.notifyWebSocketClient(deviceId, {
            action: 'reset',
            result: 'Error',
            status: false,
            message: error.message,
        });
    }
}

```

Fitur reset sistem diimplementasikan dalam metode *handleResetCommand* yang menangani proses reset status perangkat dan pembaruan final data pengguna. Sebelum melakukan reset, sistem melakukan kalkulasi final terhadap poin dan jumlah botol yang terkumpul, memastikan bahwa semua aktivitas sebelum reset tercatat dengan akurat. Implementasi menggunakan *Promise.all* untuk melakukan pembaruan data secara paralel agar meningkatkan efisiensi proses. Hasil reset dikomunikasikan melalui pesan terstruktur yang mencakup status operasi, jumlah poin yang ditambahkan dan total poin terkini.

```

switch (action) {
    case 'auth':
        this.system.userHandler.handleAuth(deviceId, data);
        break;

    case 'point':
        this.system.userHandler.handlePoint(deviceId, data);
        break;

    case 'register':
        this.system.userHandler.handleRegisterMQTT(deviceId, data);
        break;
    default:
        console.warn("MQTT topic tidak match: ", action);
}

```

MQTTHandler mengelola komunikasi antara server dan perangkat IoT menggunakan protokol MQTT. Handler ini mengimplementasikan Quality of Service (QoS) level 2 untuk memastikan pengiriman pesan dalam sistem. Handler ini menangani empat topik utama yakni;

1. "vending/+/auth" untuk autentikasi pengguna
2. "vending/+/point" untuk manajemen poin insentif
3. "vending/+/register" untuk menangani registrasi kartu pengguna baru ke dalam sistem.

```
switch (action) {
    case 'subscribe':
        this.handleSubscribe(ws, deviceId);
        break;

    case 'auth':
        this.system.userHandler.handleAuth(deviceId, data);
        break;

    case 'point':
        this.system.userHandler.handlePoint(deviceId, data);
        break;

    case 'reset':
        this.system.userHandler.handleResetCommand(deviceId, data.userId,
data.counter);
        break;

    case 'register':
        this.system.userHandler.handleRegister(deviceId, data);
        break;

    default:
        console.error(`Unknown WebSocket action: ${action}`);
        ws.send(JSON.stringify({ status: false, message: 'Unknown action' }));
}
```

WebSocketHandler mengelola komunikasi *real-time* melalui protokol *WebSocket* antara server dan antarmuka pengguna dalam sistem *Reverse Vending*

Machine. Handler ini mengimplementasikan lima tindakan utama untuk menangani berbagai aspek interaksi sistem. Mode 'subscribe' digunakan untuk menginisiasi koneksi WebSocket dengan perangkat, 'auth' menangani proses autentikasi pengguna, 'point' mengelola transaksi poin insentif, 'reset' memproses perintah reset, serta 'register' untuk menangani pendaftaran pengguna baru ke dalam sistem.

Pada iterasi kedua, server dibangun menggunakan bahasa Python untuk menggunakan OpenCV untuk memproses gambar. Ini memproses gambar untuk mengetahui apakah objek yang dimasukkan ke dalam mesin adalah botol sesuai dengan persyaratan. Proses pemrosesan gambar rata-rata membutuhkan 1-2 detik, yang mencakup analisis bentuk dan ukuran objek sebelum sistem mulai menambah poin.

Sistem pendeteksian botol pada *Reverse Vending Machine* (RVM) ini, yang dibangun menggunakan server Flask berbasis Python, menggunakan teknologi visi komputer melalui OpenCV. Flask berfungsi sebagai server web untuk menangani pengunggahan gambar, dan OpenCV mendeteksi objek botol dalam gambar yang diunggah dengan model MobileNet SSD. Selain itu, sistem ini terintegrasi dengan Firebase untuk mengunggah foto yang telah diproses ke penyimpanan cloud, dan Node.js API digunakan untuk mengontrol insentif pengguna setelah botol terdeteksi.

```
@app.route('/upload', methods=['POST'])
def upload_file():
    if 'imageFile' not in request.files:
        return jsonify({"error": "No image file provided"}), 400

    file = request.files['imageFile']
    if file.filename == '':
        return jsonify({"error": "No selected file"}), 400
```

Proses identifikasi botol dimulai saat pengguna mengirimkan foto melalui ESP32 CAM ke endpoint /upload menggunakan POST. Pada tahap awal, server Flask menerima foto dan memeriksa apakah foto tersebut ada dalam permintaan

dengan nama `imageFile`. Jika foto tidak ditemukan, server akan memberikan pesan kesalahan dengan status 400 yang menunjukkan bahwa foto tidak ditemukan.

```

    if file and file.filename.lower().endswith(('.png', '.jpg', '.jpeg')):
        file_bytes = file.read()

    else:
        return jsonify({"error": "Invalid file format. Only PNG and JPEG are
accepted."}), 400

```

Selanjutnya, sistem melakukan validasi untuk memastikan bahwa file yang diunggah memiliki nama dan format yang benar. Jika nama file kosong atau format file tidak sesuai dengan yang diharapkan, maka server akan mengembalikan pesan error. Jika file valid, server akan membaca konten file dalam bentuk byte menggunakan `file.read()` dan mencatat ukuran file dalam byte sebagai informasi tambahan. Jika terjadi kesalahan saat membaca file, server mengembalikan status error 500 dengan pesan kesalahan bahwa terdapat masalah dalam membaca file tersebut.

```

    frame = cv2.imdecode(np.frombuffer(file_bytes, np.uint8),
cv2.IMREAD_COLOR)

    if frame is None:
        return jsonify({"error": "Invalid image file"}), 400

    if not os.path.exists("tmp"):
        os.makedirs("tmp")

    filename = secure_filename(file.filename)
    original_file_path = os.path.join("tmp", filename)
    with open(original_file_path, 'wb') as f:
        f.write(file_bytes)

    bottle_found, annotated_frame, percentage =
detect_bottle_and_draw(frame)
    annotated_file_name = f"annotated_{uuid.uuid4()}_{filename}"
    annotated_file_path = os.path.join("tmp",
annotated_file_name)

```

```
cv2.imwrite(annotated_file_path, annotated_frame)
```

Setelah file berhasil dibaca, gambar di-*decode* dari *buffer byte* menjadi format gambar yang dapat diproses oleh OpenCV menggunakan ***cv2.imdecode()***. Jika *decoding* berhasil, gambar akan disimpan sementara di direktori tmp setelah server memastikan direktori tersebut ada atau membuatnya jika belum ada. Nama file akan diamankan menggunakan fungsi ***secure_filename()*** untuk memastikan nama file bebas dari karakter ilegal sebelum disimpan. Setelah gambar berhasil disimpan, server memverifikasi ukuran file untuk memastikan bahwa file tidak kosong atau berukuran 0 byte. Jika file kosong, server akan mengembalikan pesan error 400.

```
def detect_bottle_and_draw(frame):
    blob = cv2.dnn.blobFromImage(cv2.resize(frame, (300, 300)), 0.007843,
(300, 300), 127.5)
    net.setInput(blob)
    detections = net.forward()

    bottle_found = False
    percentage = 0

    for i in range(detections.shape[2]):
        confidence = detections[0, 0, i, 2]
        if confidence > 0.5:
            idx = int(detections[0, 0, i, 1])
            if CLASSES[idx] == "bottle":
                logging.info("Bottle detected!")
                bottle_found = True
                percentage = int(confidence * 100)

            box = detections[0, 0, i, 3:7] * np.array([frame.shape[1],
frame.shape[0], frame.shape[1], frame.shape[0]])
            (startX, startY, endX, endY) = box.astype("int")

            cv2.rectangle(frame, (startX, startY), (endX, endY), (0, 255,
0), 2)
```

```

        label = f"{CLASSES[idx]}: {percentage}%"
        y = startY - 8 if startY - 10 > 10 else startY + 10
        cv2.putText(frame, label, (startX, y),
cv2.FONT_HERSHEY_SIMPLEX, 1.0, (0, 255, 0), 2)
        break

    if not bottle_found:
        logging.info("No bottle detected.")

    return bottle_found, frame, percentage

```

Pada tahap selanjutnya, server memanggil fungsi `detect_bottle()` untuk mendeteksi apakah ada botol dalam gambar yang diunggah. Pada tahap deteksi botol, sistem menggunakan MobileNet SSD sebagai model deteksi objek dengan *confidence score* yang telah ditentukan sebesar 0.5. Ambang batas ini dipilih berdasarkan praktik umum dalam literatur *object detection* untuk menyeimbangkan dan menghindari deteksi palsu dan memastikan objek yang terdeteksi adalah objek nyata. Hal tersebut sesuai dengan rekomendasi penggunaan *threshold confidence* pada sistem deteksi objek untuk menghindari deteksi palsu, seperti yang diungkapkan oleh (Bouazizi et al., 2024) dalam penelitian mengenai implementasi SSD-MobileNet pada aplikasi ADAS.

Pada penelitian (Cai & Vasconcelos, 2018) juga menunjukkan bahwa penggunaan ambang batas sangat penting untuk menentukan apakah suatu nilai dikategorikan sebagai positif atau negatif. Nilai ambang batas yang diterapkan adalah 0,5. Nilai ini dianggap memberikan kriteria yang cukup untuk menetapkan suatu nilai sebagai positif.

```

def upload_to_firebase(file_path, firebase_path):
    try:
        logging.info("Starting upload to Firebase Storage")
        blob = bucket.blob(firebase_path)
        blob.upload_from_filename(file_path)
        blob.make_public()
        logging.info(f"File uploaded to Firebase Storage!")
    except Exception as e:
        logging.error(f"Error uploading file: {e}")

```

```

        return blob.public_url
    except Exception as e:
        logging.error(f"Error uploading to Firebase Storage:
{str(e)}")
        return None

```

Pengunggahan gambar ke Firebase dilakukan secara paralel tanpa menghalangi thread utama dengan menggunakan `ThreadPoolExecutor`. Executor ini diinisialisasi dengan parameter `max_workers=3`, yang berarti sistem dapat menjalankan maksimal tiga thread secara bersamaan.

```

def background_task(original_path, annotated_path, original_name,
annotated_name, percentage):
    try:
        original_firebase_path =
f'vending/original/{original_name}'
        annotated_firebase_path =
f'vending/label/{percentage}_{annotated_name}'

        original_url = upload_to_firebase(original_path,
original_firebase_path)
        annotated_url = upload_to_firebase(annotated_path,
annotated_firebase_path)

        if original_url:
            send_data_to_node_api(original_url)

        logging.info(f"Original and annotated images uploaded.
URLs: {original_url}, {annotated_url}")

        if original_url and annotated_url:
            if os.path.exists(original_path):
                os.remove(original_path)
                logging.info(f"Deleted temporary file:
{original_path}")
            if os.path.exists(annotated_path):
                os.remove(annotated_path)

```

```

logging.info(f"Deleted temporary file:
{annotated_path}")

except Exception as e:
    logging.error(f"Error in background task: {str(e)}")

```

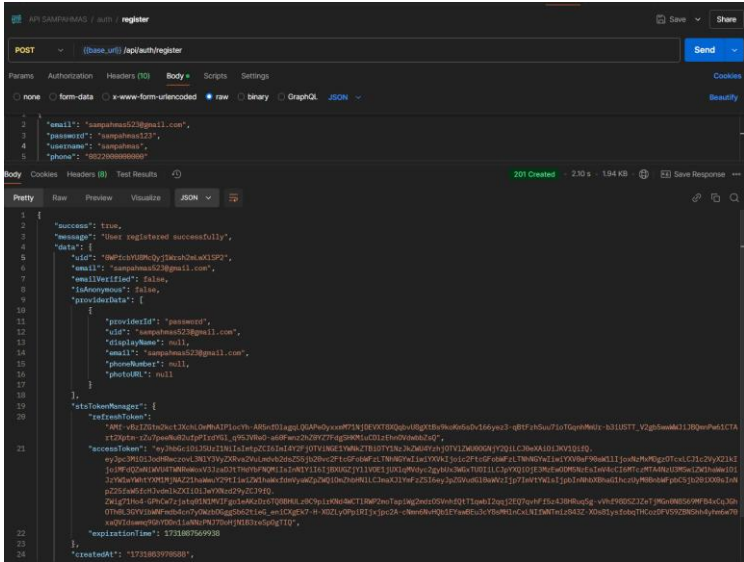
Executor ini akan memanggil fungsi `background_task()`, yang bertanggung jawab untuk melakukan pengunggahan gambar ke Firebase Storage. Proses ini dilakukan di latar belakang, sehingga server dapat terus merespons request pengguna tanpa harus menunggu pengunggahan gambar selesai. Thread yang menjalankan *background_task* tidak memblokir thread utama, sehingga aplikasi tetap responsif.

Fungsi `executor.submit()` digunakan untuk menjalankan tugas di latar belakang dan dieksekusi dengan argumen `file_path` (lokasi gambar yang disimpan sementara) dan `unique_file_name` (nama file). Setelah `executor.submit()` dipanggil untuk memulai pengunggahan di latar belakang, sistem akan langsung melanjutkan ke baris kode berikutnya yang mengembalikan respons "Bottle detected" atau "No bottle detected" kepada klien. Jadi, pengembalian pesan "Bottle detected" tidak diblokir oleh proses pengunggahan gambar ke Firebase. Pengunggahan gambar dan pengiriman data berjalan secara paralel, sementara sistem segera mengirimkan respons ke klien tanpa penundaan.

Jika botol berhasil terdeteksi, sistem mengembalikan respons ke klien dengan pesan bahwa botol telah terdeteksi dan status `True`. Jika tidak ada botol yang terdeteksi dalam gambar, sistem mengembalikan pesan bahwa botol tidak ditemukan dengan status `False`.

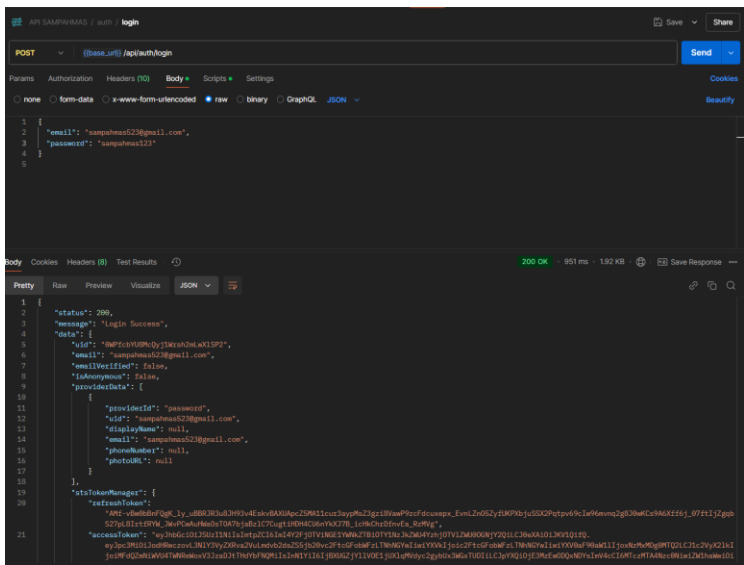
4.2.4 Testing

A. API Auth



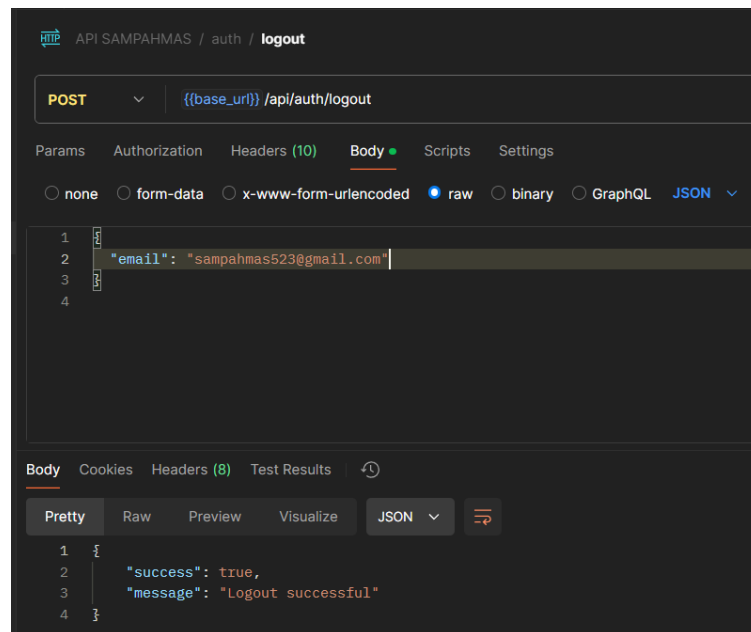
Gambar 4.11 API Daftar Pengguna

Endpoint pendaftaran pengguna (/api/auth/register) yang ditunjukkan pada Gambar 4.14 adalah salah satu kebutuhan data yang didukung oleh API Autentikasi. Terdapat dua data hasil yang ditunjukkan dengan format JSON, yaitu meta dan data. Pada hasil meta, terdapat key message yang memberikan informasi bahwa proses pendaftaran telah berhasil. Sedangkan pada hasil data, terdapat detail informasi mengenai pengguna baru, seperti ID pengguna, nama, alamat email, dan token autentikasi yang dapat digunakan untuk mengakses sistem.



Gambar 4.12 API Login

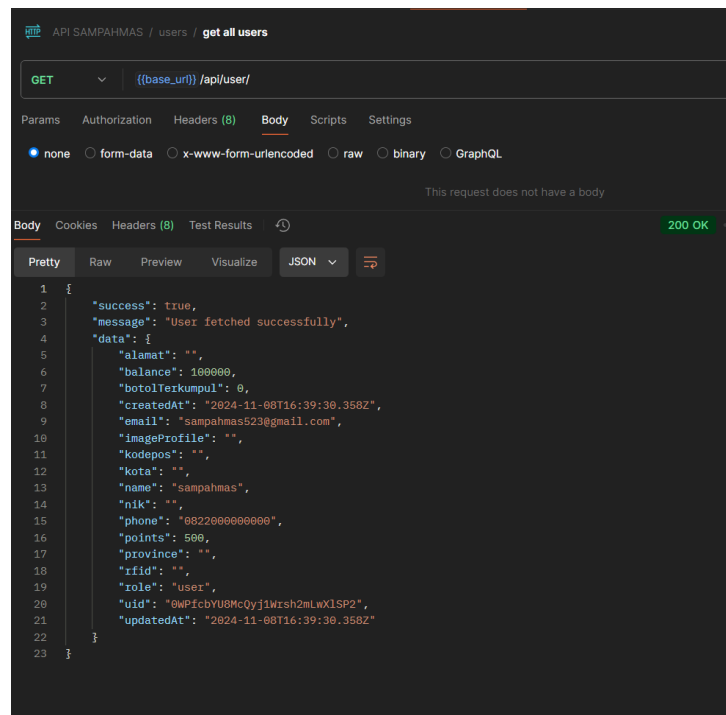
Endpoint login pengguna (/api/auth/login) yang ditunjukkan pada Gambar 4.15 adalah salah satu kebutuhan API Autentikasi. Respons JSON dari API ini mencakup dua bagian: meta dan data. Pada bagian meta, terdapat key message yang memberikan informasi tentang status login. Pada bagian data, terdapat informasi detail mengenai pengguna yang telah berhasil login, termasuk ID pengguna, alamat email, dan token autentikasi untuk mengakses layanan lain dalam sistem.



Gambar 4.13 API Logout

Endpoint logout (/api/auth/logout) yang ditunjukkan pada Gambar 4.16 adalah bagian dari API Autentikasi yang mengelola sesi pengguna. Respons JSON dari API ini memiliki key message di bagian meta, yang mengonfirmasi bahwa proses logout telah berhasil dilakukan. Sesi pengguna diakhiri, dan token autentikasi pengguna dihapus, memastikan keamanan data pengguna setelah logout.

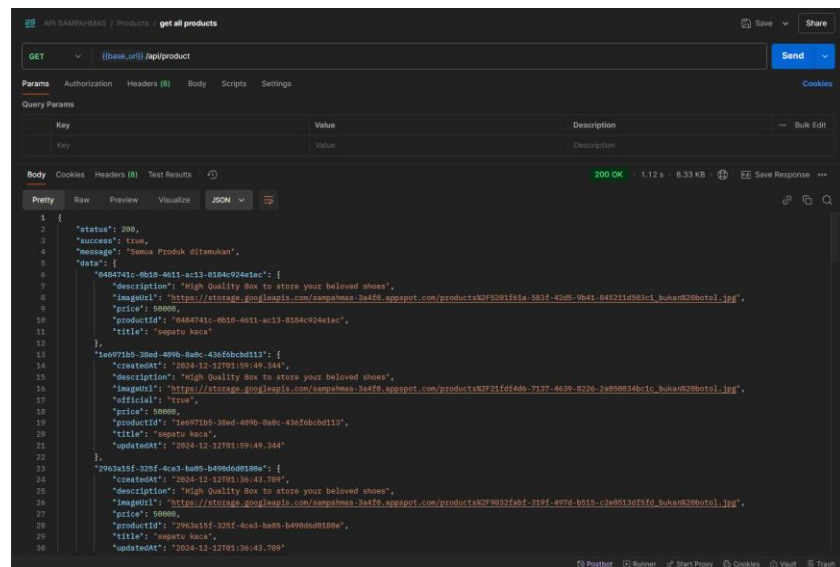
B. API User



Gambar 4.14 API Profile User

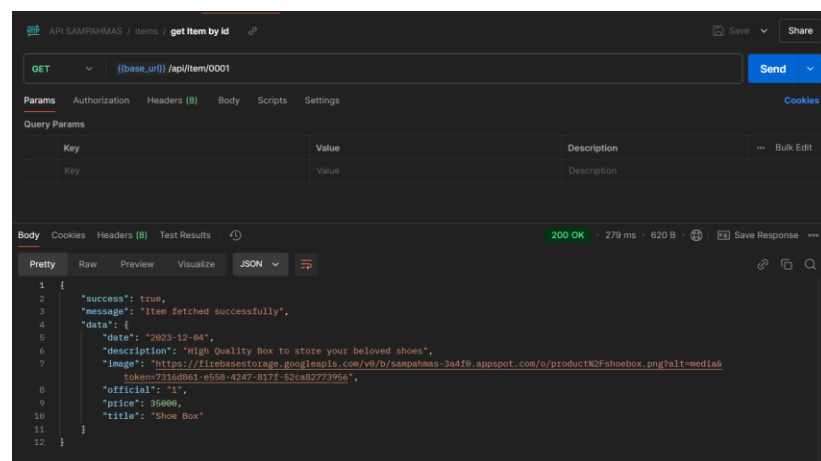
Endpoint profil pengguna (`/api/user`) yang ditunjukkan pada Gambar 4.17 menyediakan detail profil pengguna. API ini mengembalikan data dalam format JSON, yang terdiri dari meta dan data. Pada meta, terdapat pesan status dari API ini, sedangkan pada bagian data terdapat informasi lengkap tentang profil pengguna seperti ID, nama, alamat email, nomor telepon, dan detail profil lainnya yang disimpan dalam sistem.

C. API Product



Gambar 4.15 API Get Product

Endpoint daftar item (/api/Product) yang ditunjukkan pada Gambar 4.18 adalah salah satu kebutuhan data yang disediakan oleh *API Product*. Hasil yang ditunjukkan memiliki format JSON yang terdiri dari dua bagian, yaitu meta dan data. Pada bagian meta, terdapat key message yang memberikan informasi mengenai status permintaan data *product*. Sedangkan pada bagian data, berisi detail *product* yang tersedia.

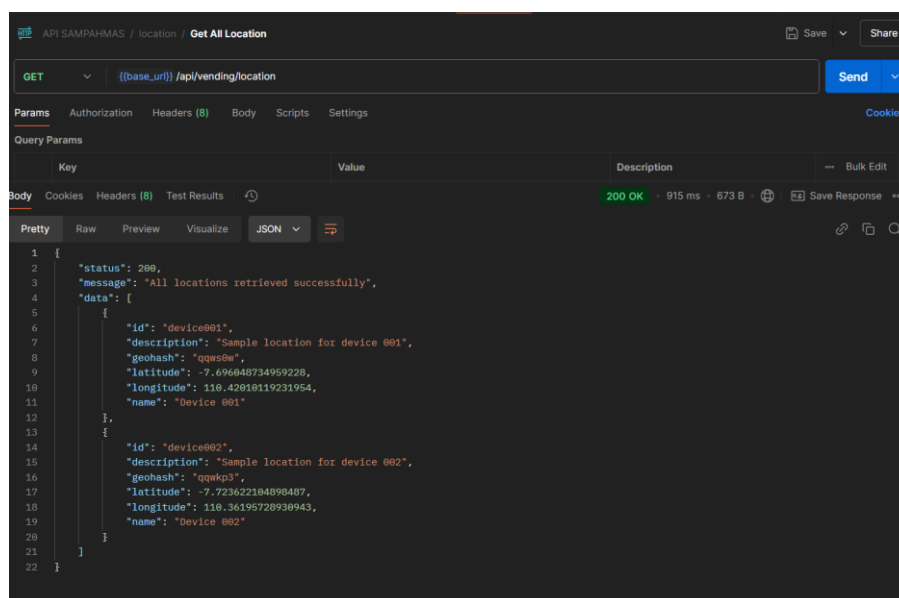


Gambar 4.16 API Get Product by id

Endpoint detail item berdasarkan ID (/api/product/:id) yang ditunjukkan pada Gambar 4.19 adalah salah satu kebutuhan yang disediakan oleh *API product*.

Respons JSON ini memiliki dua bagian, yaitu meta dan data. Pada meta, terdapat pesan status dari API ini, sedangkan pada data, terdapat informasi rinci mengenai item spesifik yang diminta seperti ID, nama, deskripsi, harga, dan stok item.

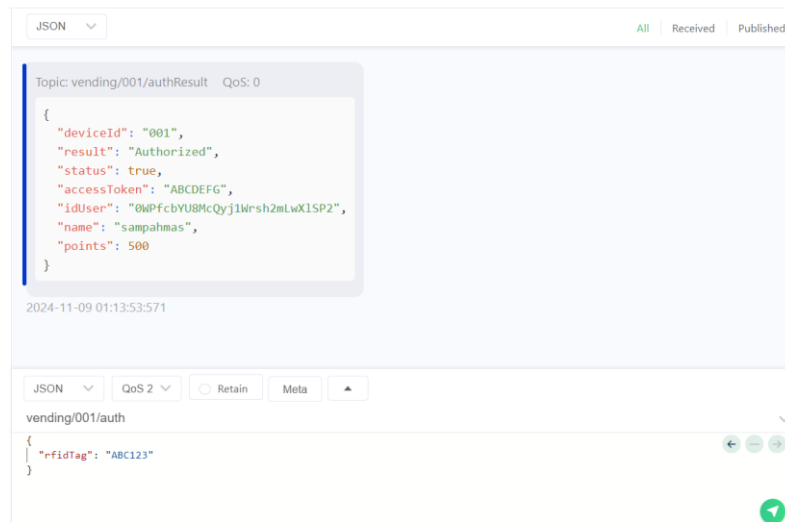
D. API Location



Gambar 4.17 API Get lokasi vending

Endpoint daftar lokasi *Reverse Vending Machine* (/api/vending/location) yang ditunjukkan pada Gambar 4.20 adalah salah satu kebutuhan yang disediakan oleh API Lokasi. API ini mengembalikan data dalam format JSON yang terdiri dari meta dan data. Pada bagian meta, terdapat key message yang memberikan informasi status respons API. Sedangkan pada bagian data, terdapat informasi lengkap mengenai lokasi mesin penjual otomatis, termasuk ID lokasi, nama lokasi, alamat, dan koordinat geografis dari lokasi tersebut.

E. MQTT Topic Auth

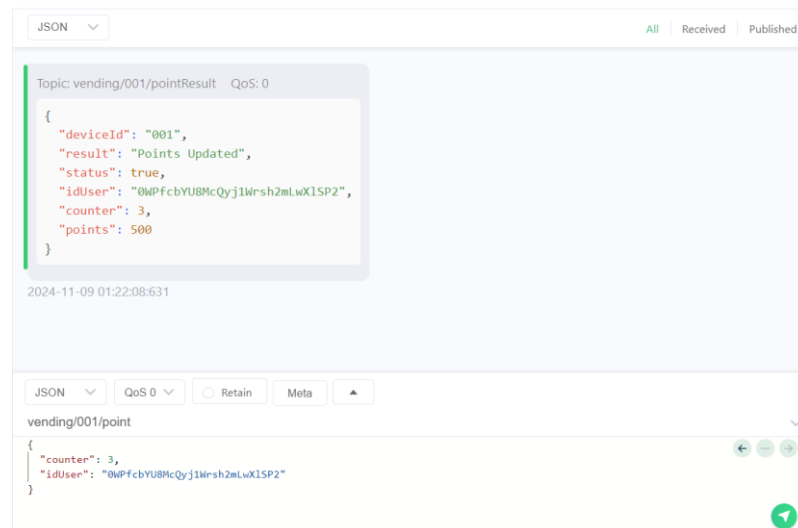


Gambar 4.18 Topic Auth

MQTT Topic Auth digunakan sebagai mekanisme autentikasi antara perangkat IoT vending machine dan server. Pada gambar tersebut, vending machine mengirimkan data autentikasi ke server melalui topik `vending/001/auth`, dengan payload JSON yang mencakup rfid, yaitu identitas unik yang digunakan untuk mengenali pengguna atau kartu RFID. Ketika server menerima data autentikasi ini, server merespons melalui topik `vending/001/authResult` dengan detail hasil autentikasi.

Respons dari server berisi `deviceId`, yang mengidentifikasi vending machine yang melakukan autentikasi dan `result` yang menunjukkan status autentikasi "Authorized" jika berhasil. Selain itu, terdapat status berupa nilai boolean (`true` atau `false`) untuk mengindikasikan apakah autentikasi berhasil. Server juga mengirimkan `accessToken`, yaitu token akses yang dapat digunakan untuk sesi berikutnya.

F. MQTT Topic Point



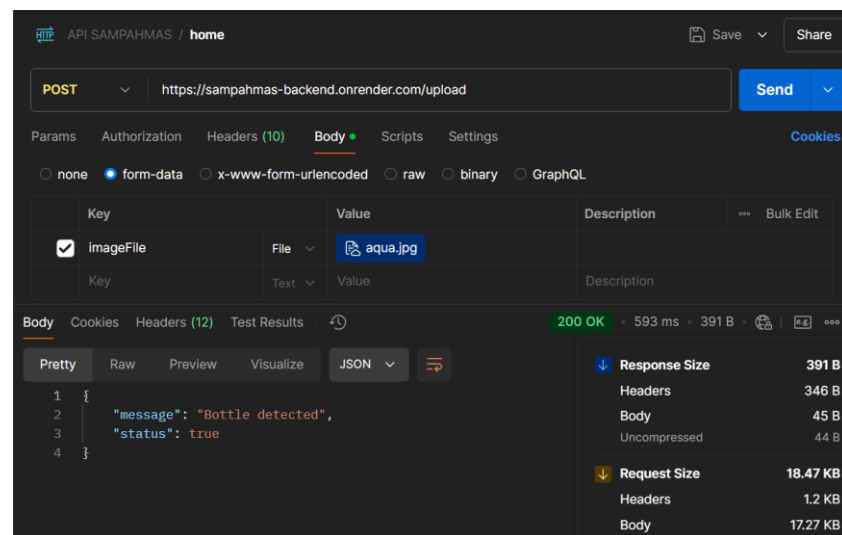
Gambar 4.19 Topic Point

MQTT Topic Point digunakan untuk menghitung dan memperbarui jumlah botol yang dikumpulkan oleh pengguna melalui vending machine. Pada Gambar 4.19, vending machine mengirimkan data ke server melalui topik vending/001/point. Data yang dikirim mencakup counter, yaitu jumlah botol yang telah dimasukkan oleh pengguna, dan uid atau userId, yang merupakan ID unik pengguna yang sedang menggunakan mesin tersebut.

Setelah menerima data tersebut, server merespons melalui topik vending/001/pointResult dengan beberapa informasi penting. Pertama, deviceId menunjukkan ID vending machine yang melakukan penghitungan, sedangkan result menunjukkan status dari proses pembaruan poin, misalnya "Points Updated" untuk mengindikasikan bahwa poin pengguna telah berhasil diperbarui. Selain itu, status bernilai true untuk menunjukkan bahwa pembaruan berhasil dilakukan.

Respons server juga mengembalikan uid atau userId untuk mengidentifikasi pengguna yang poinnya diperbarui, serta counter yang menunjukkan jumlah botol yang telah dikumpulkan dalam sesi tersebut. Terakhir, points menginformasikan total poin yang dimiliki oleh pengguna setelah pembaruan. Dengan demikian, vending machine dan pengguna dapat melihat bahwa poin telah diperbarui secara otomatis berdasarkan jumlah botol yang dikumpulkan, memastikan akumulasi poin yang transparan dan akurat.

G. Flask Python Deteksi Objek



Gambar 4.20 Upload Gambar Deteksi objek botol

Gambar 4.20 Upload Gambar Deteksi Objek Botol menunjukkan proses unggah file gambar untuk mendeteksi objek botol menggunakan API yang dibangun dengan Flask Python. Pada gambar ini, endpoint /upload menerima permintaan HTTP POST dengan file gambar (imageFile) yang diunggah oleh pengguna.

Setelah menerima gambar, server menjalankan proses deteksi objek untuk mengidentifikasi keberadaan botol di dalam gambar. Hasil dari deteksi tersebut ditampilkan dalam format respon JSON yang mencakup dua atribut utama yaitu message dan status. Pada pengujian ini, gambar dengan ukuran file sekitar 17.27 KB diunggah ke endpoint /upload. Hasil respons menunjukkan bahwa objek botol berhasil terdeteksi dengan message yang berisi "Bottle detected" dan status bernilai true. Proses ini berlangsung dengan waktu respons sebesar 593 ms dan menghasilkan ukuran respons sebesar 391 B.

Atribut message memberikan konfirmasi bahwa objek botol telah terdeteksi dalam gambar, dengan pesan "Bottle detected", sedangkan atribut status bernilai true, yang menunjukkan bahwa proses deteksi berjalan sukses. Pada testing yang dilakukan, sistem berhasil dan mampu mendeteksi objek botol dalam gambar yang diunggah dengan akurat dan efisien, baik dari segi waktu respons maupun ukuran data yang dihasilkan.

Tabel 4.1 Hasil pengujian deteksi objek

NO	OBJEK	SKOR	WAKTU	STATUS
1	Botol Aqua 600ml	77%	0.3997	TRUE
2	Botol LeMinerale 600ml	97%	0.3056	TRUE
3	Botol Nestle Pure Life 600ml	84%	0.4	TRUE
4	Botol Super02 600ml	82%	0.4056	TRUE
5	Botol LeMinerale 330ml	97%	0.3905	TRUE
6	Botol Aqua 300ml	99%	0.3225	TRUE
7	Botol Aqua 1.1 liter	54%	0.410	TRUE
8	Botol Nestle Pure Life 1.5 liter	65%	0.3982	TRUE

Berdasarkan hasil pengujian deteksi objek botol menggunakan sistem yang telah dikembangkan, diperoleh data berupa hasil deteksi oleh sistem dengan skor atau confidence yang bervariasi. Pada prototype *Reverse Vending Machine (RVM) Sampahmas*, ditemukan bahwa botol berukuran besar memiliki tingkat akurasi deteksi yang lebih rendah. Hal ini disebabkan oleh keterbatasan ruang dalam prototype *Reverse Vending*.

Oleh karena itu, pada fase ini, pengembangan dan pengujian sistem lebih difokuskan pada botol sampah dengan ukuran tanggung, seperti botol berkapasitas 600ml. Fokus ini bertujuan untuk memastikan bahwa sistem dapat berfungsi secara optimal pada jenis botol yang paling umum ditemukan dalam kategori sampah plastik. Pendekatan ini juga membantu meningkatkan efisiensi deteksi sekaligus memvalidasi kemampuan sistem untuk mendeteksi objek dengan ukuran yang konsisten di berbagai skenario. Untuk kedepannya, pengembangan sistem dapat diperluas untuk mencakup deteksi botol dengan ukuran lebih besar atau lebih kecil dengan penyesuaian pada desain ruang dan algoritma deteksi.

H. Hasil Testing

Tabel 4.2 Hasil pengujian deteksi objek

No.	Kegiatan Pengujian	Endpoint	Deskripsi	Hasil
1	Pendaftaran pengguna	/api/auth/register	Endpoint untuk mendaftarkan pengguna baru. Respons berupa JSON yang berisi informasi pengguna seperti ID, nama, email, dan token autentikasi.	Sesuai
2	Login pengguna	/api/auth/login	Endpoint untuk login pengguna. Respons JSON mencakup status login, ID pengguna, email, dan token autentikasi.	Sesuai
3	Logout pengguna	/api/auth/logout	Endpoint untuk mengelola sesi logout pengguna. Respons JSON memastikan sesi diakhiri dan token autentikasi dihapus.	Sesuai
4	Profil pengguna	/api/user	Endpoint untuk menampilkan detail profil pengguna, termasuk ID, nama, email, dan nomor telepon.	Sesuai
5	Daftar produk	/api/product	Endpoint untuk mendapatkan daftar item produk. Respons JSON berisi meta informasi dan detail produk.	Sesuai
6	Detail produk	/api/product/:id	Endpoint untuk mendapatkan detail produk berdasarkan ID. Respons JSON mencakup nama, deskripsi, harga, dan stok produk.	Sesuai
7	Lokasi vending machine	/api/vending/location	Endpoint untuk mendapatkan informasi lokasi Reverse Vending Machine, termasuk ID lokasi, nama lokasi, alamat, dan koordinat geografis.	Sesuai
8	Topik autentikasi vending	vending/001/auth	Topik MQTT untuk autentikasi perangkat vending machine menggunakan data seperti RFID. Server merespons dengan detail autentikasi dan token akses.	Sesuai
9	Topik poin vending	vending/001/point	Topik MQTT untuk memperbarui jumlah botol yang dikumpulkan pengguna melalui vending machine. Server merespons dengan informasi total poin setelah pembaruan.	Sesuai
10	Upload deteksi botol	/upload	Endpoint Flask untuk mengunggah gambar guna mendeteksi botol. Respons JSON mengembalikan status deteksi dan pesan seperti "Bottle detected" atau "No bottle detected".	Sesuai

Pengujian API endpoint secara keseluruhan menunjukkan bahwa seluruh fitur yang dikembangkan telah berfungsi sesuai dengan ekspektasi. Pengujian dilakukan pada berbagai endpoint utama termasuk autentikasi, pengelolaan data pengguna, produk, lokasi vending machine, serta deteksi botol melalui gambar.

Dengan hasil pengujian yang valid, API ini siap untuk diintegrasikan dan digunakan oleh hipster untuk pengembangan frontend di aplikasi Sampahmas dan juga perangkat IoT Reverse Vending Machine. Integrasi dapat mendukung kebutuhan pengguna melalui antarmuka aplikasi yang dirancang secara intuitif dan fungsional.

4.2.5 Deployment



Gambar 4.21 Hasil *Reverse Vending Machine*

Pada tahap ini, hasil pengembangan dari *Hacker* diserahkan kepada *Hustler* untuk diterapkan dalam lingkungan nyata. Sistem yang mencakup autentikasi RFID, QR Code, dan deteksi botol diintegrasikan pada prototipe *Reverse Vending Machine* (RVM). *Hustler* bertanggung jawab untuk menguji implementasi ini secara langsung di lapangan, memastikan fitur-fitur yang telah dikembangkan dapat berfungsi sesuai kebutuhan pengguna. Selain itu, *Hustler* memanfaatkan sistem ini untuk menarik perhatian calon pengguna awal melalui uji coba terbatas, serta mengumpulkan umpan balik untuk pengembangan berikutnya.

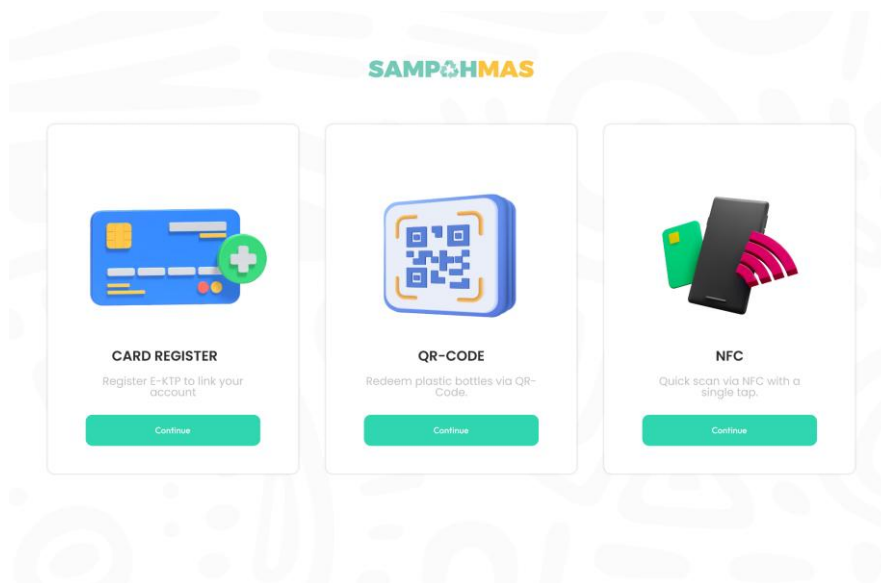
4.2.6 Review

Setelah deployment, *Hustler* melakukan evaluasi terhadap sistem melalui pengumpulan umpan balik dari pengguna awal dan observasi performa fitur dalam lingkungan nyata. Fokus review mencakup kecepatan dan akurasi autentikasi menggunakan RFID dan QR Code, serta kemampuan deteksi botol oleh ESP32-CAM yang terintegrasi dengan Flask server. Feedback yang diterima menunjukkan keberhasilan sistem dalam menjalankan fungsi-fungsi dasar dengan baik, namun terdapat beberapa masukan seperti optimasi waktu respons untuk deteksi botol yang dimasukkan secara berturut-turut dan peningkatan kecepatan sinkronisasi data poin pada Firebase.

Hasil review ini kemudian ditindaklanjuti oleh *Hacker* untuk melakukan penyempurnaan sistem. *Hacker* merancang langkah-langkah perbaikan, seperti optimalisasi algoritma pemrosesan gambar pada Flask server, pengaturan ulang prioritas koneksi WebSocket untuk mempercepat pembaruan data, serta penyesuaian mekanisme autentikasi untuk meningkatkan pengalaman pengguna. Masukan dari tahap ini menjadi panduan bagi *Hacker* dalam mempersiapkan iterasi berikutnya untuk mencapai sistem yang lebih efisien dan user-friendly.

4.3 Iterasi 3

4.3.1 Requirement

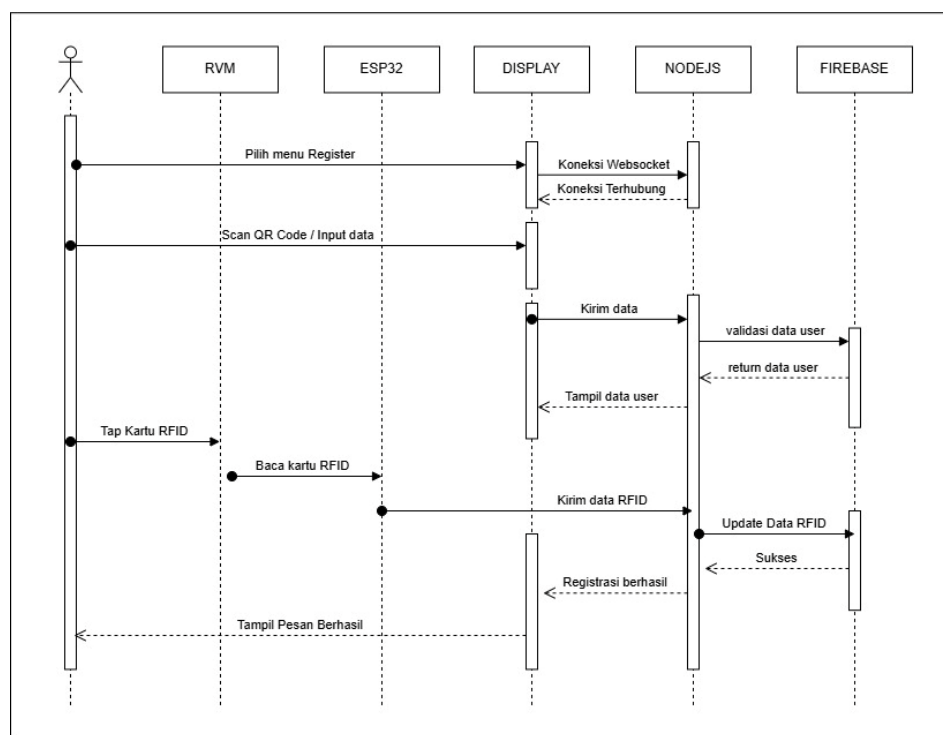


Gambar 4.22 Perubahan fungsi dan tampilan RVM

Pada iterasi ini, *Hacker* bertugas mengimplementasikan fitur baru untuk menyinkronkan kartu pengguna ke akun mereka melalui vending machine. Sistem dirancang agar pengguna dapat memindai barcode dari aplikasi Sampahmas untuk mengisi data akun secara otomatis, lalu menempelkan kartu mereka pada NFC reader untuk menyelesaikan proses sinkronisasi. Selain itu, *Hacker* juga harus memastikan fitur ini berjalan dengan lancar dan responsif, termasuk implementasi tombol interaktif yang hanya aktif jika semua data sudah valid.

4.3.2 Design

Pada tahap desain, *Hacker* merancang arsitektur sistem untuk mendukung proses sinkronisasi kartu. Desain mencakup integrasi antara vending machine, RFID reader dan *backend* sistem yang menggunakan Node.js serta Firebase.



Gambar 4.23 Sequence diagram register kartu

Pada gambar sequence diagram pendaftaran kartu menunjukkan alur interaksi antara pengguna (user) dengan berbagai komponen sistem dalam proses pendaftaran kartu RFID. Proses dimulai ketika pengguna melakukan scan QR code atau input data pada antarmuka RVM. Data ini kemudian diteruskan ke server Node.js yang telah terhubung melalui koneksi WebSocket.

Server selanjutnya melakukan validasi data pengguna dengan mengirimkan request ke Firebase. Setelah data pengguna tervalidasi, Firebase mengembalikan data tersebut ke Node.js yang kemudian ditampilkan di display RVM. Pada tahap berikutnya, pengguna diminta untuk menempelkan kartu RFID yang akan didaftarkan pada reader RVM. ESP32 akan membaca data kartu RFID tersebut dan mengirimkannya ke server Node.js.

Server Node.js kemudian memproses data RFID yang diterima dengan melakukan update data RFID pengguna di Firebase. Setelah proses update berhasil, Firebase mengirimkan konfirmasi keberhasilan yang diteruskan oleh Node.js ke display RVM dalam bentuk pesan registrasi berhasil. Proses diakhiri dengan tampilnya pesan sukses pada display RVM yang memberitahu pengguna bahwa kartu RFID telah berhasil didaftarkan ke dalam sistem.

4.3.3 Development

Hacker mengimplementasikan fitur sinkronisasi dengan menambahkan *endpoint* dan *handler* untuk mendukung pengiriman data barcode dan NFC dari vending machine ke server Node.js. Pengembangan mencakup implementasi di sisi frontend dan *backend* untuk mendukung proses registrasi yang aman.

```

handleMessage(ws, message) {
    let data;
    try {
        data = JSON.parse(message);
    } catch (error) {
        console.error('Invalid WebSocket message:', error);
        ws.send(JSON.stringify({ status: false, message:
'Invalid message format' }));
        return;
    }

    const { action, deviceId } = data;

    try {
        switch (action) {
            ....
            case 'register':
                this.system.userHandler.handleRegister(deviceId, data);
                break;
            .....
        }
    }
}

```

Pada fungsi *handleMessage* berperan sebagai pengendali utama yang menerima dan memproses pesan *WebSocket*. Fungsi ini melakukan parsing data JSON yang diterima dan melakukan validasi format pesan. Apabila terjadi kesalahan dalam format pesan, sistem akan mengirimkan respons error kepada klien. Setelah validasi berhasil, sistem mengekstrak informasi penting seperti action dan deviceId dari pesan yang diterima.

```

async handleRegister(deviceId, data) {
  try {
    const { rfid, email, phone } = data;

    if (!email) {
      throw new Error('Email is invalid or missing');
    }
    if (!phone) {
      throw new Error('Phone is invalid or missing');
    }
    if (!rfid) {
      throw new Error('RFID is invalid or missing');
    }

    const user = await
userRepository.getUserByEmail(email);
    if (!user) {
      throw new Error('User not found with the provided
email');
    }

    const userId = user.uid;

    if (user.phone !== phone) {
      throw new Error('Phone number does not match the
user\'s record');
    }
    if (!userId) {
      throw new Error('User ID is invalid or missing');
    }
    const response = await
userRepository.updateRFID(userId, rfid);

```

```

        this.system.notifyWebSocketClient(deviceId, {
            action: 'registerResult',
            result: 'success',
            status: true,
            message: 'RFID registered successfully',
            data: response,
        });
    } catch (error) {
        console.error(`Error handling register for device
        ${deviceId}: ${error.message}`, error);

        if (this.system.isDeviceRegistered(deviceId) &&
        this.system.notifyWebSocketClient) {
            this.system.notifyWebSocketClient(deviceId, {
                action: 'registerResult',
                result: 'Error',
                status: false,
                message: error.message,
            });
        }
    }
}

```

Pada proses registrasi, fungsi `handleRegister` mengimplementasikan logika validasi untuk memastikan data pengguna. Fungsi ini memeriksa keberadaan dan validitas tiga parameter kunci yaitu email, nomor telepon, dan RFID. Selanjutnya, sistem melakukan verifikasi kesesuaian nomor telepon yang diberikan dengan data yang tersimpan dalam record pengguna.

Jika proses berhasil, sistem mengirimkan respons positif melalui WebSocket dengan format yang terstruktur berisi action 'rfid', status true, dan data RFID. Jika terjadi error, sistem mengirimkan respons negatif dengan detail error untuk penanganan di sisi client.

```

handleMessage(topic, message) {
    case 'register':

        this.system.userHandler.handleRegisterMQTT(deviceId, data);
}

```

```
break;
```

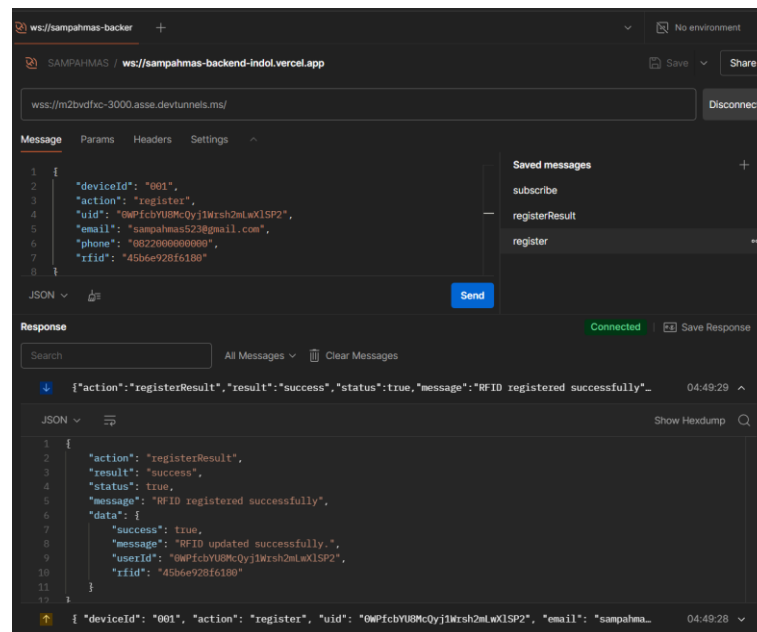
Pada implementasi MQTT Handler untuk fitur registrasi kartu NFC, sistem mengenali pesan dengan format 'vending/:deviceId/register'. Ketika pesan registrasi diterima, sistem akan meneruskannya ke *handleRegisterMQTT* pada *UserHandler*.

handleRegisterMQTT merupakan metode khusus untuk memproses data RFID yang diterima dari perangkat vending machine. Metode ini melakukan Validasi keberadaan data RFID kemudian pengiriman notifikasi ke client *WebSocket* dengan status dan data RFID

Setelah semua berhasil dilakukan, sistem melakukan pembaruan data RFID pengguna dalam database melalui method *updateRFID*. Proses ini menggunakan *userId* sebagai referensi untuk memastikan ketepatan pembaruan data. Sistem dirancang untuk memberikan umpan balik *real-time* kepada pengguna melalui notifikasi *WebSocket* yang mengembalikan status keberhasilan atau kegagalan proses registrasi ketampilan antarmuka.

Dari sisi tampilan, diimplementasikan halaman *register.html* yang menggabungkan pemindai QR code untuk autentikasi awal dan form registrasi kartu NFC. Tampilan dirancang dengan dua kolom kolom kiri untuk area pemindaian QR code dan instruksi penggunaan, sementara kolom kanan berisi form registrasi dengan field email, nomor telepon, dan RFID. Sistem akan memberikan *feedback* visual melalui modal untuk status registrasi

4.3.4 Testing



Gambar 4.24 Pengujian registrasi kartu melalui websocket

Dalam proses testing sistem vending machine, pengujian dilakukan terhadap sinkronisasi kartu RFID guna memastikan bahwa seluruh langkah berjalan sesuai dengan skenario yang telah ditentukan. Data yang dikirim ke server melalui protokol WebSocket mencakup informasi seperti deviceId, uid, email, phone, dan rfid. Data tersebut dikemas dalam format JSON dan dikirim menggunakan aksi register. Server kemudian memberikan respons berupa hasil registrasi kartu dengan informasi status yang menjelaskan bahwa RFID telah berhasil diregistrasi.

Metode pengujian ini menggunakan pendekatan *blackbox testing* untuk memverifikasi bahwa semua fungsi bekerja sesuai dengan ekspektasi tanpa memeriksa kode internal sistem. Dari hasil pengujian yang ditampilkan, server merespons dengan hasil sukses. Respons ini menandakan bahwa proses registrasi telah berhasil dikirim ke server dan disinkronisasi di database. Melalui metode ini, integrasi antara berbagai komponen mulai dari pemindaian RFID hingga penyimpanan data di cloud berlangsung dengan lancar dan optimal mendukung operasional vending machine.

4.3.5 Deployment

Setelah pengujian selesai, *Hacker* menyerahkan sistem yang telah dikembangkan kepada *Hustler* untuk deployment di lingkungan nyata. Proses ini mencakup instalasi perangkat keras pada vending machine, pengaturan koneksi *backend*, serta sinkronisasi dengan Firebase. Sistem yang diserahkan mencakup fitur sinkronisasi kartu, antarmuka baru pada vending machine, dan koneksi yang mendukung *real-time* update.

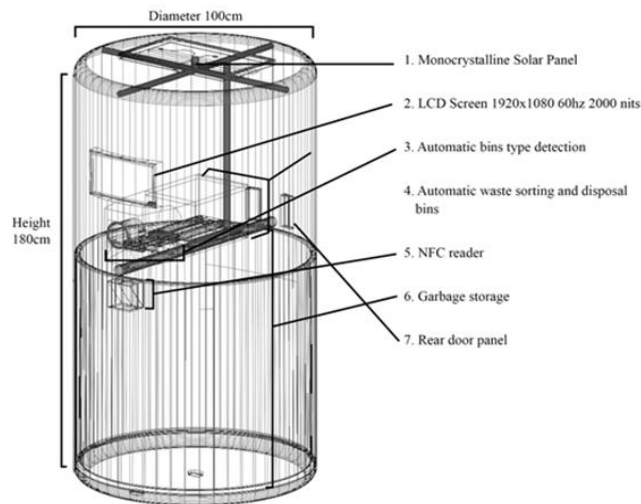
4.3.6 Review

Hustler melakukan review terhadap sistem yang telah dikembangkan oleh *Hacker* dengan mengumpulkan umpan balik dari pengguna awal. Berdasarkan hasil review, ditemukan bahwa sistem berjalan lancar dalam membaca kartu dan menghubungkannya ke akun pengguna. Namun, terdapat beberapa masukan terkait peningkatan responsivitas pembacaan kartu RFID dan penyempurnaan tampilan antarmuka. Umpan balik ini digunakan *Hacker* untuk merencanakan optimasi lebih lanjut pada iterasi berikutnya.

4.4 Hasil Akhir

Tim Haystudio telah mencapai kemajuan signifikan dalam pengembangan bisnis Sampahmas melalui pendekatan Lean Startup. Melalui tiga iterasi *Build-Measure-Learn*, tim berhasil mengembangkan solusi dan potensi nyata dalam mengatasi permasalahan pengelolaan sampah.

4.4.1 Reverse Vending Machine (RVM) Sampahmas



Gambar 4.25 Spesifikasi *Reverse Vending Machine*

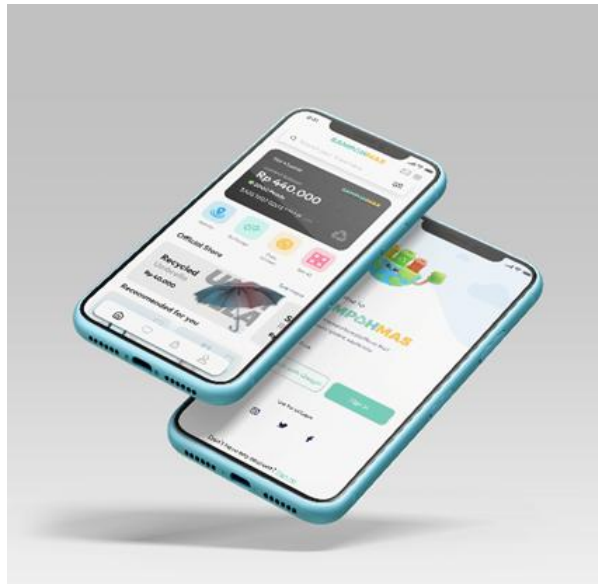
Pencapaian utama dari pengembangan ini adalah realisasi *Reverse Vending Machine* (RVM) Sampahmas, sebuah inovasi teknologi yang mengintegrasikan berbagai komponen canggih. RVM ini dilengkapi dengan layar LCD Full HD dengan kecerahan 2000 nits untuk antarmuka pengguna, sistem deteksi dan sortir sampah otomatis, serta pembaca NFC untuk identifikasi pengguna.



Gambar 4.26 Purwarupa Alpha Reverse Vending Machine

Saat ini, purwarupa Alpha RVM telah direalisasikan dan sedang dalam tahap pengujian di Perumahan Muslim Darussalam 3, Sleman, Yogyakarta, dengan target peluncuran di tahun 2026.

4.4.2 Mockup Aplikasi Mobile Sampahmas



Gambar 4.27 Mockup Aplikasi Sampahmas

Aplikasi mobile Sampahmas yang dirancang dengan memperhatikan prinsip User Experience (UX). Aplikasi ini menawarkan antarmuka yang intuitif dan ramah pengguna, alur kerja yang efisien, serta sistem reward berbasis poin yang terintegrasi. Proses pengembangan aplikasi ini telah melalui berbagai tahapan validasi pengguna untuk memastikan setiap fitur memberikan pengalaman yang lancar dan efektif.

4.4.3 Hasil Pencapaian & Penghargaan Ide Bisnis Sampahmas

Validasi pasar dan potensi bisnis Sampahmas semakin diperkuat dengan raihan berbagai penghargaan bergengsi, baik di tingkat nasional maupun internasional. Tim Haystudio berhasil meraih Juara 3 dalam HACKATHON 2024: *Business Idea Solution* yang diselenggarakan oleh ITC IT Centrum pada Desember 2024. Prestasi ini dilanjutkan dengan menjadi Juara 1 PEKAN IT 2024: UI/UX Design dari Universitas Singaperbangsa Karawang pada Oktober 2024.

Di tingkat internasional, tim berhasil meraih Juara 1 dalam kompetisi bisnis INTERNATIONAL INVENTION, INNOVATION & DESIGN COMPETITION 2024: Business Competition yang diselenggarakan oleh INDES Universitas Teknologi MARA Malaysia pada Juni 2024. Tim sebelumnya juga telah memenangkan sejumlah penghargaan, termasuk Juara 1 PORSEMATIK 2023:

UI/UX Contest dari HMIF LEM FTI UII pada Desember 2023, Juara 2 Permikom Nasional: UI/UX Contest dari Permikomnas Indonesia pada November 2023, dan Juara 1 Informatik Expo 2021/2022: Mobile App Development.

Serangkaian prestasi ini menunjukkan bahwa Sampahmas dapat menjadi solusi yang inovatif dan kualitasnya. Peluang untuk bekerja sama dengan berbagai pemangku kepentingan tersedia di tim. Selain itu, prestasi ini meningkatkan kemampuan tim untuk mendapatkan dana yang diperlukan untuk kemajuan di masa mendatang.

BAB V

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Kegiatan pengembangan sistem *Reverse Vending Machine* telah selesai dan menghasilkan beberapa hasil, seperti yang disebutkan pada bab sebelumnya. Berikut adalah kesimpulan dari kegiatan yang telah dilakukan untuk mengembangkan sistem.

- a. Sistem autentikasi ganda menggunakan RFID dan QR Code telah berhasil diimplementasikan dan telah lolos uji coba menunjukkan bahwa sistem mampu memberikan fleksibilitas akses bagi pengguna RVM. Penggunaan WebSocket untuk komunikasi *real-time* antara server dan display tablet memastikan respon yang cepat dalam proses autentikasi.
- b. Integrasi ESP32-CAM dengan OpenCV berhasil melakukan deteksi botol menggunakan MobileNet SSD. Sistem trigger menggunakan sensor IR meningkatkan efisiensi dalam proses pengambilan gambar. Penggunaan tablet sebagai display terpisah berhasil mengatasi keterbatasan ESP32 dalam menampilkan antarmuka pengguna.
- c. Arsitektur sistem yang mengkombinasikan Node.js dan Flask server memungkinkan pemrosesan yang efisien untuk berbagai jenis data, sementara implementasi Firebase sebagai database *real-time* mendukung sinkronisasi data yang cepat dan reliable. Metodologi pengembangan dengan tiga iterasi memungkinkan penyempurnaan sistem secara bertahap, termasuk penambahan fitur registrasi RFID pada iterasi ketiga yang meningkatkan aksesibilitas sistem.
- d. Pengembangan API siap dan dapat digunakan untuk kebutuhan data frontend atau client dirancang oleh hipster yang akan memberikan fleksibilitas dalam pengembangan dan integrasi sistem ke berbagai platform.

5.2 Saran

Melalui refleksi dan kesimpulan dari kegiatan yang telah dilaksanakan terdapat beberapa poin yang bisa disampaikan sebagai saran dalam pengembangan-pengembangan sistem selanjutnya yaitu sebagai berikut:

- a. Penggunaan mikrokontroler dengan spesifikasi yang lebih tinggi seperti *Raspberry Pi* untuk melakukan deteksi objek secara mandiri. *Raspberry Pi* memiliki beberapa keuntungan signifikan, yaitu:
 4. Kemampuan melakukan komputasi *machine learning* secara lokal tanpa membutuhkan server Python eksternal
 5. Dapat digunakan langsung sebagai display dengan menggunakan layar sentuh menggantikan kebutuhan *display* terpisah dan koneksi WebSocket
 6. Konektivitas yang lebih fleksibel dengan berbagai jenis sensor dan perangkat tambahan
- b. Pengembangan sistem deteksi perlu ditingkatkan dengan implementasi machine learning untuk mengenali berbagai jenis botol dan kondisinya. Dengan menggunakan *Raspberry Pi*, *model machine learning* dapat dijalankan langsung pada perangkat yang akan meningkatkan kecepatan deteksi dan mengurangi ketergantungan pada koneksi internet.

Antarmuka pengguna dapat dikembangkan langsung pada *Raspberry Pi* menggunakan sistem desktop modern agar lebih responsif dan terintegrasi.

DAFTAR PUSTAKA

- HAstuti, I., Ariestya, W. W., & Solehudin, B. (2022). Deteksi Objek Daun Semanggi Secara Real Time Menggunakan CNN-Single Shot Multibox Detector (SSD). *Jurnal Ilmiah FIFO*, 14(1), 47. <https://doi.org/10.22441/fifo.2022.v14i1.005>
- Astuti, R. D., Prakoso, G. B., & Jatiningrum, W. S. (2023). FAKTOR PENGHAMBAT NIAT DAN PERILAKU DALAM KEGIATAN BANK SAMPAH Studi Kasus di Kabupaten Sleman, DI Yogyakarta. *JISI: Jurnal Integrasi Sistem Industri*, 10(1), 39. <https://doi.org/10.24853/jisi.10.1.39-52>
- Bouazizi, O., Azroumahli, C., El Mourabit, A., & Oussouaddi, M. (2024). Road Object Detection using SSD-MobileNet Algorithm: Case Study for Real-Time ADAS Applications. *Journal of Robotics and Control (JRC)*, 5(2), 551–560. <https://doi.org/10.18196/jrc.v5i2.21145>
- Cai, Z., & Vasconcelos, N. (2018). Cascade R-CNN: Delving into High Quality Object Detection. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 6154–6162. <https://doi.org/10.1109/CVPR.2018.00644>
- Choirudin, R., & Adil, A. (2019). Implementasi Rest Api Web Service dalam Membangun Aplikasi Multiplatform untuk Usaha Jasa. *MATRIK: Jurnal Manajemen, Teknik Informatika Dan Rekayasa Komputer*, 18(2), 284–293. <https://doi.org/10.30812/matrik.v18i2.407>
- Efendi, Y. (2018). Internet Of Things (Iot) Sistem Pengendalian Lampu Menggunakan Raspberry Pi Berbasis Mobile. *Jurnal Ilmiah Ilmu Komputer*, 4(2), 21–27. <https://doi.org/10.35329/jiik.v4i2.41>
- Febrianto Widyoutomo, Hamidillah Ajie, & Widodo. (2021). Pengembangan Web Service Modul Mahasiswa Pada Sistem Informasi Akademik Universitas Negeri Jakarta. *PINTER: Jurnal Pendidikan Teknik Informatika Dan Komputer*, 5(1), 68–75. <https://doi.org/10.21009/pinter.5.1.9>
- Handoko, P., Hermawan, H., & Jaya, S. (2018). Reverse Vending Machine Penukaran Limbah Botol Kemasan Plastik Dengan Tiket Sebagai Alat Tukar

- Mata Uang. *Jurnal Universitas Muhammadiyah Jakarta*, 1–12.
- Hasanudin, U. (2023). *Pemda DIY Batasi Sampah ke TPST Piyungan 600 Ton Per Hari*.
<https://jogjapolitan.harianjogja.com/read/2023/06/08/511/1137900/pemda-diybatasi-sampah-ke-tpst-piyungan-600-ton-per-hari>
- Hermawan, L. C., Mubarak, M. R., Mairudin, H., Mahdiyan, A., & Yulianti, Y. (2020). Pengujian Black Box pada Aplikasi Verifikasi Data Nasabah dengan Menggunakan Metode Boundary Value Analysis. *Jurnal Teknologi Sistem Informasi Dan Aplikasi*, 3(3), 119. <https://doi.org/10.32493/jtsi.v3i3.5331>
- Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., & Adam, H. (2017). *MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications*.
<http://arxiv.org/abs/1704.04861>
- Irawan, J. D., & Adriantantri, E. (2018). Pemanfaatan QR-Code Sebagai Media Promosi Toko. *Jurnal MNEMONIC*, 1(2), 57.
- Irsyad, R. (2018). Penggunaan Python Web Framework Flask Untuk Pemula. *Khazanah Informatika : Jurnal Ilmu Komputer Dan Informatika*, 4(1), 8–15.
- Iskandar, D., Satria Fajar Kusumah, F., & Fajri, H. (2024). Pembuatan Web Service Sistem Cari Kerja Perguruan Tinggi Dengan Metode Rest. *JATI (Jurnal Mahasiswa Teknik Informatika)*, 8(3), 2968–2975.
<https://doi.org/10.36040/jati.v8i3.9604>
- Islam, S. U., Ferraioli, G., Pascasio, V., & Vitale, S. (2024). *Performance Analysis of YOLOv3 , YOLOv4 and MobileNet SSD for Real Time Object Detection*. July.
- Kehutanan, K. L. H. dan. (2023). *Sistem Informasi Pengelolaan Sampah Nasional*. Kementerian Lingkungan Hidup Dan Kehutanan.
<https://sipsn.menlhk.go.id/sipsn/public/>
- Khairun Nisa Meiah Ngafidin, Arista, A., & Nisa Sofia Amriza, R. (2021). Implementasi Firebase Realtime Database pada Aplikasi FeedbackMe sebagai Penghubung Guru dan Orang Tua. *Jurnal RESTI (Rekayasa Sistem Dan Teknologi Informasi)*, 5(2), 327–334. <https://doi.org/10.29207/resti.v5i2.2909>

- Kojansow, C. W., Manembu, P. D. K., & Rumagit, A. M. (2024). Pengembangan Platform Internet Of Things (IoT) menggunakan komunikasi WebSocket. *Jurnal Teknik Informatika*, 19(3), 259–269. <https://doi.org/10.35793/jti.v19i3.53680>
- Law, K. L., Starr, N., Siegler, T. R., Jambeck, J. R., Mallos, N. J., & Leonard, G. H. (2020). The United States' contribution of plastic waste to land and ocean. *Science Advances*, 6(44), 1–7. <https://doi.org/10.1126/sciadv.abd0288>
- Mulana, L., Prihandani, K., Rizal, A., Singaperbanga, U., & Abstract, K. (2022). Analisis Perbandingan Kinerja Framework Codeigniter Dengan Express.Js Pada Server RESTful Api. *Jurnal Ilmiah Wahana Pendidikan*, 8(16), 316–326. <https://doi.org/10.5281/zenodo.7067707>
- Raharjana, I. K., & Justitia, A. (2015). Pembuatan Model Sequence Diagram Dengan Reverse Engineering Aplikasi Basis Data Pada Smartphone Untuk Menjaga Konsistensi Desain Perangkat Lunak. *JUTI: Jurnal Ilmiah Teknologi Informasi*, 13(2), 133. <https://doi.org/10.12962/j24068535.v13i2.a482>
- Santoso, H., Pungki, D., Aziz, A., & Zaini, A. (2022). Implementasi Agile Scrum pada Proses Pengembangan Aplikasi Monitoring MBKM di UNIKAMA. *Jurnal Terapan Sains & Teknologi*, 4(4), 208–215.
- Srinath, K. R., & Associate. (2017). Python – The Fastest Growing Programming Language K. *Journal of Science and Medicine in Sport*, 20(7), 678–683.
- Struk, M. (2019). Waste Management Incentive Programs and Municipal Finance. *7th International Conference on Sustainable Solid Waste Management, June*.
- Suarna, D., & Sopyan, E. (2023). Implementasi Internet of Things (IoT) dalam Memonitoring Komsumsi Listrik. 4(2), 163–170.
- Susanto, B. M., Atmadji, E. S. J., & Brenkman, W. L. (2018). Implementasi Mqtt Protocol Pada Smart Home Security Berbasis Web. *Jurnal Informatika Polinema*, 4(3), 201–205. <https://doi.org/10.33795/jip.v4i3.207>
- Tasya. (2023). *Jogja Darurat Sampah, Ahli UGM: Kenapa Baru Gaduh Sekarang?* 11 Agustus 2023, 16.21. <https://ugm.ac.id/id/berita/jogja-darurat-sampah-ahli-ugm-kenapa-baru-gaduh-sekarang/>
- Umam, K., & Negara, B. S. (2016). Deteksi Obyek Manusia Pada Basis Data Video

- Menggunakan Metode Background Subtraction Dan Operasi Morfologi. *Jurnal CoreIT: Jurnal Hasil Penelitian Ilmu Komputer Dan Teknologi Informasi*, 2(2), 31. <https://doi.org/10.24014/coreit.v2i2.2391>
- Wagya, A. (2019). Prototipe Modul Praktik untuk Pengembangan Aplikasi Internet of Things (IoT). *Setrum: Sistem Kendali-Tenaga-Elektronika-Telekomunikasi-Komputer*, 8(2), 238. <https://doi.org/10.36055/setrum.v8i2.6561>
- Yusup, M. (2022). Teknologi Radio Frequency Identification (RFID) Sebagai Tools System Pembuka Pintu Outomatis pada Smart House. *Jurnal Media Infotama*, 18(2), 367–373.
- Zebua, E. T. P., & Rosyani, P. (2024). Perancangan Deteksi Objek Kendaraan Bermotor Berbasis OpenCV Python menggunakan Metode HOG-SVM untuk Analisis Lalu Lintas Cerdas. *Jurnal AI Dan SPK: Jurnal Artificial Intelligent Dan Sistem Penunjang Keputusan*, 2(1), 16–26.
- Zulkhaidi, T. C. A.-S., Maria, E., & Yulianto, Y. (2020). Pengenalan Pola Bentuk Wajah dengan OpenCV. *Jurnal Rekayasa Teknologi Informasi (JURTI)*, 3(2), 181. <https://doi.org/10.30872/jurti.v3i2.4033>

