# BÁO CÁO DỰ ÁN: ECOCHECK - HỆ THỐNG THU GOM RÁC THẢI ĐỘNG ỨNG DỤNG DỮ LIỆU MỞ LIÊN KẾT

## CHƯƠNG 1: TỔNG QUAN VỀ DỰ ÁN

### 1.1 Lý do chọn đề tài

Trong kỷ nguyên số hóa, quản lý đô thị thông minh trở thành nhu cầu cấp thiết, đặc biệt tại các thành phố lớn như TP.HCM, nơi mật độ dân cư cao gây áp lực lên hạ tầng kỹ thuật. Một vấn đề nổi cộm là quy trình thu gom rác thải sinh hoạt vẫn vận hành theo cơ chế "Tĩnh" truyền thống, với lộ trình và khung giờ cố định bất kể lượng rác thực tế tại điểm thu gom. Điều này dẫn đến lãng phí nhiên liệu, nhân lực và đôi khi gây ô nhiễm cục bộ do rác ứ đọng chưa được xử lý kịp thời.

Cuộc thi Olympic Tin học Sinh viên Việt Nam 2025 (OLP 2025) với chủ đề "Ứng dụng dữ liệu mở liên kết (Linked Open Data - LOD) phục vụ chuyển đổi số" đặt ra bài toán về việc xây dựng các hệ thống có khả năng chia sẻ, tái sử dụng và hiểu dữ liệu theo ngữ nghĩa. Dự án EcoCheck được lựa chọn và phát triển nhằm giải quyết bài toán tối ưu hóa logistics trong thu gom rác thải (Dynamic Waste Collection), đồng thời áp dụng triệt để các tiêu chuẩn dữ liệu mở NGSI-LD để hướng tới việc tích hợp vào hệ sinh thái dữ liệu chung của Đô thị thông minh.

### 1.2 Đối tượng, phạm vi & mục tiêu nghiên cứu

Mục tiêu nghiên cứu chính của dự án là xây dựng một nền tảng chuyển đổi số toàn diện, giúp chuyển dịch từ mô hình thu gom rác thải tĩnh sang mô hình động dựa trên nhu cầu thực tế. Dự án hướng đến việc tạo ra một hệ thống kết nối chặt chẽ và thông minh giữa người dân, đơn vị thu gom và nhà quản lý.

Đối tượng thụ hưởng của dự án bao gồm:

* Các công ty môi trường đô thị mong muốn tối ưu hóa chi phí vận hành và nâng cao hiệu quả quản lý
* Người dân thành phố mong muốn một môi trường sống sạch đẹp, văn minh hơn
* Các cơ quan quản lý nhà nước cần dữ liệu thực tế để quy hoạch và ra quyết định

Phạm vi nghiên cứu của đề tài bao gồm việc thiết kế kiến trúc hệ thống tuân thủ chuẩn LOD, phát triển các ứng dụng di động đa nền tảng phục vụ người dùng cuối và lực lượng lao động, cùng hệ thống quản trị trung tâm dựa trên bản đồ số thời gian thực.

### 1.3 Cấu trúc báo cáo

Báo cáo này được cấu trúc thành bốn chương chính nhằm cung cấp cái nhìn toàn diện từ lý thuyết nền tảng đến thực tiễn triển khai.

Chương 1 giới thiệu tổng quan về bối cảnh dự án, lý do hình thành và phạm vi thực hiện.

Chương 2 đi sâu vào cơ sở lý thuyết, phân tích các công nghệ cốt lõi và quy trình nghiệp vụ.

Chương 3 phân tích chi tiết kiến trúc hệ thống, từ thiết kế cơ sở dữ liệu, mô hình hóa đối tượng đến giao diện người dùng.

Cuối cùng, Chương 4 tổng kết các kết quả đạt được, những thách thức kỹ thuật đã vượt qua và đề xuất định hướng phát triển trong tương lai của EcoCheck.

### 1.4 Phạm vi dự án

Phạm vi dự án EcoCheck bao gồm việc phát triển trọn bộ giải pháp phần mềm gồm: Backend server xử lý nghiệp vụ và tích hợp dữ liệu liên kết (Orion-LD integration), Ứng dụng di động dành cho người dân (EcoCheck User), Ứng dụng dành cho nhân viên thu gom (EcoCheck Worker), và Trang quản trị Web (Admin Dashboard).

Tuy nhiên, dự án vẫn tồn tại một số hạn chế nhất định do thời gian và nguồn lực có hạn. Hiện tại, hệ thống mới chỉ dừng lại ở mức độ mô phỏng dữ liệu GPS và các kịch bản thu gom trong môi trường kiểm thử (Sandbox). Việc triển khai thực tế sẽ đòi hỏi hạ tầng phần cứng IoT mạnh mẽ hơn, thiết bị định vị chuyên dụng gắn trên xe rác và sự phối hợp về mặt chính sách từ các cơ quan quản lý nhà nước để truy cập vào các nguồn dữ liệu mở về giao thông và dân cư.

### 1.5 Đối tượng người dùng

Hệ thống được thiết kế để phục vụ ba nhóm người dùng chính với các nhu cầu và vai trò riêng biệt:

**Nhóm thứ nhất - Người dân (Citizens):** Đóng vai trò là các "cảm biến xã hội" (Social Sensors), sử dụng ứng dụng để check-in điểm rác, gửi yêu cầu thu gom và nhận điểm thưởng từ hệ thống Gamification. Để khắc phục vấn đề người lớn tuổi không quen với công nghệ, hệ thống hỗ trợ tính năng "Family Account" (Quản lý hộ gia đình) cho phép người trẻ check-in thay người già. Ngoài ra, hệ thống tích hợp mô hình "Ví Rác Thải" (Waste Wallet) với cơ chế PAYT (Pay-As-You-Throw) mềm, cho phép người dân giảm phí rác dựa trên lượng rác thực tế đã giảm thiểu, tạo động lực tài chính thực tế thay vì chỉ dựa vào voucher.

**Nhóm thứ hai - Nhân viên thu gom (Workers):** Bao gồm tài xế và công nhân vệ sinh, những người cần lộ trình di chuyển tối ưu và công cụ để báo cáo sự cố tại hiện trường. Ứng dụng được thiết kế với UX "Smart Checklist" thay vì GPS Navigation liên tục, cho phép nhân viên tự lái theo cách họ quen thuộc trong khi hệ thống chỉ đóng vai trò là danh sách check-list cần làm. Tính năng "Dẫn đường bằng Giọng nói" (Voice Guidance) giúp nhân viên không cần nhìn màn hình liên tục khi đang lái xe. Lợi ích cốt lõi cho nhân viên là "Về sớm hơn" - hệ thống lọc bỏ các điểm "ma" (không có rác), giúp hoàn thành công việc nhanh hơn mà lương không đổi.

**Nhóm thứ ba - Nhà quản lý (Managers/Dispatchers):** Sử dụng Dashboard để giám sát toàn bộ hoạt động của đội xe, phân tích hiệu suất thu gom và xử lý các tình huống khẩn cấp theo thời gian thực trên bản đồ số. Hệ thống cung cấp dữ liệu thực tế (Data-driven decision) thay vì ước lượng, giúp nhà quản lý trả tiền dựa trên dữ liệu thật và quy hoạch bãi rác chính xác hơn.

### 1.6 Công cụ và môi trường phát triển

Để hiện thực hóa dự án EcoCheck, nhóm phát triển đã sử dụng một bộ công cụ lập trình hiện đại và mạnh mẽ. Môi trường phát triển tích hợp (IDE) chính là Visual Studio Code cho cả Backend và Web Frontend, trong khi Android Studio và Xcode hỗ trợ cho việc biên dịch và gỡ lỗi ứng dụng Mobile Flutter. Hệ điều hành phát triển bao gồm cả Windows và macOS, đảm bảo tính tương thích đa nền tảng của mã nguồn. Các công cụ quản lý mã nguồn như Git/GitHub và công cụ chứa hóa Docker được sử dụng triệt để xuyên suốt dự án nhằm đảm bảo quy trình tích hợp liên tục (CI/CD) và khả năng triển khai đồng bộ trên nhiều môi trường máy chủ khác nhau.

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## CHƯƠNG 2: CƠ SỞ LÝ THUYẾT

### 2.1 Giới thiệu chung

Cơ sở lý thuyết của dự án xoay quanh khái niệm Web Ngữ nghĩa (Semantic Web) và Dữ liệu mở liên kết (Linked Open Data - LOD). Khác với các hệ thống thông tin truyền thống lưu trữ dữ liệu dưới dạng các bản ghi rời rạc trong các bảng cơ sở dữ liệu quan hệ, EcoCheck áp dụng tiêu chuẩn NGSI-LD (Next Generation Service Interfaces - Linked Data) do ETSI ban hành. Tiêu chuẩn này cho phép định nghĩa các thực thể (như Xe rác, Điểm tập kết, Công nhân) kèm theo ngữ cảnh (Context), giúp máy tính có thể tự động hiểu, suy diễn và liên kết dữ liệu từ nhiều nguồn khác nhau, tạo tiền đề cho các ứng dụng thông minh trong Smart City.

### 2.2 Công nghệ phía máy chủ (Backend)

Phía máy chủ (Server-side) của EcoCheck được xây dựng dựa trên nền tảng Node.js kết hợp với framework Express. Sự lựa chọn này dựa trên khả năng xử lý bất đồng bộ (non-blocking I/O) mạnh mẽ của Node.js, rất phù hợp với các ứng dụng yêu cầu thời gian thực cao như theo dõi vị trí xe. Đặc biệt, Backend tích hợp module xử lý Orion-LD, đóng vai trò là Context Broker để quản lý vòng đời của các thực thể dữ liệu liên kết, cho phép thực hiện các truy vấn ngữ nghĩa phức tạp và đăng ký nhận thông báo thay đổi trạng thái (Subscriptions) từ các thực thể trong hệ thống.

### 2.3 Công nghệ phía giao diện (Frontend)

Đối với phía máy khách (Client-side), dự án áp dụng chiến lược phát triển đa nền tảng để tối ưu nguồn lực. Ứng dụng quản trị (Web Dashboard) được xây dựng bằng thư viện React kết hợp với công cụ build Vite và framework CSS Tailwind, mang lại hiệu năng cao và tốc độ phát triển nhanh chóng. Đối với ứng dụng di động (User App và Worker App), framework Flutter của Google được lựa chọn nhờ khả năng biên dịch ra mã máy (native code) cho cả iOS và Android từ một codebase duy nhất, đảm bảo tính đồng nhất về giao diện và trải nghiệm người dùng mượt mà trên mọi thiết bị.

### 2.4 Hệ quản trị cơ sở dữ liệu

Hệ thống sử dụng PostgreSQL làm hệ quản trị cơ sở dữ liệu quan hệ chính, kết hợp với phần mở rộng PostGIS. Đây là yếu tố then chốt cho một ứng dụng mang tính chất hậu cần và bản đồ như EcoCheck. PostGIS cung cấp các kiểu dữ liệu hình học (Geometry) và các hàm xử lý không gian địa lý mạnh mẽ, cho phép thực hiện các truy vấn phức tạp như "tìm tất cả điểm rác chưa thu gom trong bán kính 500m" hoặc "tính toán lộ trình di chuyển tối ưu giữa các điểm" một cách hiệu quả và chính xác ngay tại tầng cơ sở dữ liệu. Ngoài ra, hệ thống còn tích hợp TimescaleDB để tối ưu hóa việc lưu trữ và truy vấn dữ liệu chuỗi thời gian (time-series data) như lịch sử vị trí xe, lịch sử check-in theo thời gian.

### 2.5 Môi trường triển khai

Quy trình phát triển và vận hành hệ thống được tự động hóa tối đa thông qua việc sử dụng Docker và Docker Compose. Toàn bộ các dịch vụ thành phần (Backend API, Database, Context Broker, Web Server) đều được container hóa, giúp loại bỏ hoàn toàn các vấn đề về sự khác biệt môi trường ("it works on my machine"). Ngoài ra, hệ thống các scripts tự động (PowerShell cho Windows và Shell script cho Linux/macOS) như setup.ps1, start-all.ps1, run-complete-test.ps1 được phát triển để hỗ trợ việc cài đặt môi trường, kiểm thử tự động và khởi chạy hệ thống một cách nhanh chóng.

### 2.6 Mẫu thiết kế và biểu đồ

Dự án áp dụng nhiều mẫu thiết kế phần mềm (Design Patterns) tiêu chuẩn để đảm bảo tính dễ bảo trì và mở rộng của mã nguồn. Ở phía Backend, kiến trúc Microservices được áp dụng thông qua việc chia nhỏ các dịch vụ. Trong mã nguồn ứng dụng Mobile (Flutter), mẫu BLoC (Business Logic Component) được sử dụng triệt để để tách biệt logic nghiệp vụ khỏi giao diện người dùng, giúp mã nguồn rõ ràng, dễ đọc và dễ kiểm thử. Các biểu đồ UML tiêu chuẩn như Class Diagram, Sequence Diagram, Activity Diagram và State Diagram được sử dụng trong giai đoạn thiết kế để mô hình hóa cấu trúc tĩnh và hành vi động của hệ thống trước khi tiến hành lập trình.

### 2.7 Phân tích nghiệp vụ

#### 2.7.1 Nghiệp vụ cốt lõi

Quy trình nghiệp vụ cốt lõi của EcoCheck bắt đầu từ việc người dân thực hiện hành động "Check-in" rác thải thông qua ứng dụng di động. Dữ liệu này (bao gồm hình ảnh, khối lượng ước tính và tọa độ GPS) được gửi về hệ thống trung tâm để tạo thành các "Nhu cầu thu gom" (Collection Requests). Hệ thống sau đó sử dụng thuật toán để gom nhóm các điểm rác, tính toán lộ trình tối ưu và điều phối (Dispatch) nhiệm vụ đến ứng dụng của tài xế xe rác gần nhất. Tài xế nhận nhiệm vụ, thực hiện thu gom theo lộ trình được dẫn đường và cập nhật trạng thái hoàn thành kèm bằng chứng ảnh chụp.

#### 2.7.2 Nghiệp vụ nâng cao

Ngoài quy trình thu gom cơ bản, hệ thống còn tích hợp các nghiệp vụ nâng cao để tăng cường hiệu quả. Nghiệp vụ Gamification (Hóa trò chơi) khuyến khích người dân tham gia phân loại rác thông qua việc tích điểm, thăng hạng và nhận huy hiệu (Badges). Nghiệp vụ Real-time Tracking cho phép nhà quản lý giám sát vị trí và trạng thái của toàn bộ đội xe theo thời gian thực qua giao thức Web Socket. Nghiệp vụ Reporting cho phép xử lý các sự cố ngoại lệ như rác quá khổ, xe hỏng hoặc điểm thu gom bị chặn lối vào, đảm bảo quy trình vận hành không bị gián đoạn.

#### 2.7.3 Mô hình hóa dữ liệu cảm biến theo SOSA/SSN

Dự án EcoCheck áp dụng ontology SOSA/SSN (Sensor, Observation, Sample, and Actuator / Semantic Sensor Network) của W3C để chuẩn hóa dữ liệu thu thập từ cộng đồng. Trong ngữ cảnh của EcoCheck:

* **Sensor (Cảm biến):** Ứng dụng di động của người dân (đóng vai trò là Social Sensor).
* **Feature of Interest (Đối tượng quan tâm):** Điểm tập kết rác (Waste Point) hoặc Thùng rác công cộng.
* **Observed Property (Thuộc tính quan sát):** Mức độ đầy của thùng rác (fillingLevel), loại rác thải (wasteType).
* **Observation (Sự quan sát):** Mỗi hành động "Check-in" của người dân được xem là một quan sát, ghi nhận giá trị tại một thời điểm và vị trí cụ thể.

Dữ liệu này được chuyển đổi sang định dạng JSON-LD để tương thích với NGSI-LD, cho phép các hệ thống khác "hiểu" được ngữ nghĩa của dữ liệu mà không cần định nghĩa lại cấu trúc.

### 2.8 Sử dụng FiWARE Smart Data Models

Để đảm bảo khả năng liên thông dữ liệu (Interoperability) với các thành phố thông minh khác, EcoCheck không tự định nghĩa lại toàn bộ cấu trúc dữ liệu mà kế thừa từ thư viện Smart Data Models ([https://smartdatamodels.org](https://smartdatamodels.org/)):

* **WasteContainer:** Được ánh xạ cho các điểm thu gom rác cố định.
* **Vehicle:** Sử dụng để mô hình hóa xe thu gom rác, mở rộng thêm các thuộc tính về tải trọng (cargoWeight) và lộ trình (route).
* **Point of Interest:** Sử dụng cho các trạm trung chuyển (Depot) và bãi xử lý (Dump).

Tất cả các mô hình này được định nghĩa trong file ngữ cảnh ecocheck.jsonld, giúp hệ thống Context Broker (Orion-LD) xác thực và truy vấn dữ liệu chính xác.

## CHƯƠNG 3: KIẾN TRÚC HỆ THỐNG

### 3.1 Phân tích yêu cầu hệ thống

#### 3.1.1 Yêu cầu chức năng

Hệ thống EcoCheck phải đáp ứng các yêu cầu chức năng chính bao gồm: Chức năng xác thực và phân quyền người dùng (Công dân, Nhân viên, Quản lý); Chức năng quản lý và hiển thị bản đồ số thời gian thực; Chức năng Check-in rác thải với định vị GPS chính xác và upload hình ảnh; Chức năng tính toán và tối ưu hóa lộ trình thu gom tự động dựa trên danh sách điểm rác; Chức năng quản lý hồ sơ nhân viên, phương tiện và điểm tập kết; Hệ thống báo cáo thống kê trực quan và phân tích hiệu suất hoạt động định kỳ.

#### 3.1.2 Yêu cầu phi chức năng

Về mặt phi chức năng, hệ thống yêu cầu độ trễ thấp trong việc cập nhật vị trí xe (Real-time latency < 2s) để đảm bảo tính chính xác của việc giám sát; Khả năng chịu tải cao khi có hàng nghìn người dùng cùng thực hiện Check-in vào giờ cao điểm; Tính bảo mật dữ liệu cá nhân người dùng và dữ liệu vị trí nhạy cảm; Khả năng mở rộng dễ dàng nhờ kiến trúc Containerization; và đặc biệt là tính tương thích chuẩn NGSI-LD để đảm bảo khả năng liên kết dữ liệu mở với các hệ thống khác.

#### 3.1.3 Phân tích Use Case

Các Use Case điển hình của hệ thống bao gồm: "Công dân gửi yêu cầu thu gom rác cồng kềnh", "Tài xế xem danh sách lộ trình và xác nhận hoàn thành điểm gom", "Quản lý theo dõi trực tuyến đội xe trên bản đồ số", "Hệ thống tự động cảnh báo khi xe đi sai lộ trình hoặc bỏ sót điểm rác". Mỗi Use Case đều được phân tích kỹ lưỡng về luồng sự kiện chính (Main Flow), luồng phụ (Alternative Flow) và các điều kiện ngoại lệ (Exception Flow) để đảm bảo tính bao quát của hệ thống.

### 3.2 Thiết kế hệ thống

#### 3.2.1 Tổng quan kiến trúc hệ thống

Kiến trúc tổng thể của EcoCheck là sự kết hợp giữa mô hình Client-Server truyền thống và Event-Driven Architecture hiện đại. Trung tâm của hệ thống là Backend Server và Context Broker (Orion-LD), giao tiếp với cơ sở dữ liệu không gian PostGIS. Các Client (Mobile Apps, Web Dashboard) kết nối với Server qua RESTful API cho các tác vụ quản lý dữ liệu và qua Web Socket cho các tác vụ truyền tải thông tin thời gian thực.

#### 3.2.2 Biểu đồ lớp (Class Diagram)

Biểu đồ lớp của hệ thống được tổ chức thành các phân hệ chính để quản lý dữ liệu NGSI-LD phức tạp, phản ánh đúng logic nghiệp vụ của EcoCheck. Các lớp chính bao gồm: User (với các subclass Citizen, Worker, Manager), Vehicle, Route, CollectionRequest, Point, Alert, Badge, và các lớp service như RouteOptimizer, DispatchService, AnalyticsService.

#### 3.2.3 Biểu đồ tuần tự (Sequence Diagrams)

Biểu đồ tuần tự mô tả chi tiết tương tác giữa các đối tượng trong các kịch bản quan trọng. Ví dụ: Kịch bản "Tối ưu lộ trình" bắt đầu khi Quản lý kích hoạt chức năng trên Web Dashboard, Server sẽ truy vấn toàn bộ CollectionRequest có trạng thái chưa xử lý từ Database, gọi thuật toán tối ưu hóa, lưu kết quả WorkerRoute mới và đẩy thông báo cập nhật xuống Worker App của tài xế được chỉ định.

#### 3.2.4 Biểu đồ hoạt động (Activity Diagrams)

Biểu đồ hoạt động mô tả luồng công việc của một Tài xế xe rác: Bắt đầu từ việc Đăng nhập vào ứng dụng -> Kiểm tra tình trạng xe -> Nhận lộ trình được phân công -> Di chuyển đến điểm gom đầu tiên (Navigation) -> Thực hiện thu gom rác -> Chụp ảnh xác nhận hoàn thành -> Cập nhật trạng thái lên hệ thống -> Di chuyển đến điểm tiếp theo -> Kết thúc ca làm việc tại bãi đổ rác.

#### 3.2.5 Biểu đồ trạng thái (State Diagram)

Biểu đồ trạng thái tập trung mô tả vòng đời của một CollectionRequest: Khởi tạo ở trạng thái Pending (Chờ xử lý) -> Chuyển sang Scheduled (Đã lên lịch/Gán xe) -> Chuyển sang In\_Progress (Đang được thu gom) -> Kết thúc ở trạng thái Completed (Hoàn thành) hoặc Cancelled (Hủy bỏ do người dùng hoặc hệ thống).

### 3.3 Thiết kế cơ sở dữ liệu

#### 3.3.1 Lược đồ cơ sở dữ liệu (Schema)

Cơ sở dữ liệu của EcoCheck được thiết kế chuẩn hóa (Normalized) nhưng được tối ưu hóa đặc biệt cho các truy vấn không gian. Schema bao gồm các bảng chính: users, waste\_collection\_requests, routes, route\_points, incidents. Đặc biệt, các trường lưu trữ tọa độ sử dụng kiểu dữ liệu GEOMETRY(Point, 4326) của PostGIS để đảm bảo độ chính xác và hiệu năng tính toán địa lý.

#### 3.3.2 Cấu trúc bảng

Cấu trúc chi tiết của các bảng được định nghĩa trong các file migration SQL của dự án. Ví dụ, bảng routes chứa các trường quan trọng như id, worker\_id, vehicle\_id, start\_time, end\_time, status và geometry (lưu trữ đường đi polyline). Bảng incidents chứa type, description, image\_url và khóa ngoại trỏ đến routes để liên kết sự cố với lộ trình cụ thể.

#### 3.3.3 Các mối quan hệ và ràng buộc

Các ràng buộc toàn vẹn dữ liệu được thiết lập chặt chẽ trong thiết kế cơ sở dữ liệu. Khóa ngoại (FOREIGN KEY) đảm bảo tính nhất quán, ví dụ: một Route phải thuộc về một Worker và một Vehicle tồn tại. Các chỉ mục không gian (SPATIAL INDEX) được tạo trên các cột tọa độ (latitude, longitude) để tăng tốc độ thực thi các truy vấn tìm kiếm theo vị trí địa lý (Spatial Queries).

### 3.4 Hiện thực hệ thống

#### 3.4.1 Ngăn xếp công nghệ (Technology Stack)

Ngăn xếp công nghệ của dự án được lựa chọn kỹ lưỡng bao gồm: Node.js/Express cho Backend API, Flutter cho ứng dụng Mobile đa nền tảng, React cho Web Dashboard, PostgreSQL/PostGIS cho Cơ sở dữ liệu, Socket.IO cho giao tiếp thời gian thực, và Docker cho cơ sở hạ tầng triển khai.

#### 3.4.2 Cấu trúc và Tổ chức mã nguồn

Mã nguồn dự án được tổ chức khoa học theo mô hình Clean Architecture. Tại thư mục frontend-mobile, mã nguồn được chia thành các lớp: data (repositories, models, data sources), domain (entities, usecases), và presentation (blocs, widgets, pages, screens). Phía Backend cũng được phân chia rõ ràng thành src/controllers, src/routes, src/models, và src/services để đảm bảo tính tách biệt trách nhiệm (Separation of Concerns).

#### 3.4.3 Hiện thực các tính năng chính

Các tính năng chính được hiện thực hóa thông qua việc kết hợp linh hoạt giữa các API RESTful và kết nối Socket thời gian thực. Tính năng bản đồ sử dụng thư viện react-leaflet trên Web và Maps\_flutter trên Mobile App. Tính năng Context Broker được giả lập và xử lý thông qua các adapter chuyển đổi dữ liệu từ cấu trúc SQL sang định dạng JSON-LD tiêu chuẩn trước khi trả về client, đảm bảo tuân thủ yêu cầu về Dữ liệu mở liên kết.

#### 3.4.4 Tích hợp và Xử lý nguồn Dữ liệu mở

Hệ thống EcoCheck được thiết kế để hoạt động như một "Hub" dữ liệu mở:

* **Dữ liệu nền (Base Data):** Sử dụng OpenStreetMap (OSM) thông qua dịch vụ Nominatim để thực hiện Geocoding (chuyển đổi địa chỉ sang tọa độ) và OSRM để tính toán ma trận khoảng cách cho thuật toán tối ưu lộ trình.
* **Dữ liệu nghiệp vụ:** Hệ thống import dữ liệu mẫu về mạng lưới thu gom rác tại TP.HCM (Quận Gò Vấp/Thủ Đức) dưới dạng GeoJSON/NGSI-LD.
* **Khả năng mở rộng:** Kiến trúc Microservices cho phép dễ dàng tích hợp thêm các API dữ liệu mở khác trong tương lai như OpenWeatherMap (để dự báo thời tiết ảnh hưởng đến việc thu gom) hoặc OpenAQ (chất lượng không khí tại bãi rác).

### 3.5 Phân tích các thuật toán cốt lõi

#### 3.5.1 Thuật toán tối ưu hóa lộ trình (Vehicle Routing Problem - VRP)

Hệ thống EcoCheck sử dụng thuật toán Hybrid CI-SA (Cheapest Insertion + Simulated Annealing) để giải quyết bài toán VRP. Thuật toán này được chia thành hai giai đoạn:

Giai đoạn 1 - Cheapest Insertion (Xây dựng sườn lộ trình tốt ngay từ đầu):

Thuật toán bắt đầu với một route chỉ chứa điểm xuất phát (depot). Với mỗi điểm chưa được gán, thuật toán tính toán chi phí tăng thêm nếu chèn điểm đó vào mọi vị trí có thể trong route hiện tại. Chi phí được tính bằng công thức: addedCost = (u->p + p->v) - (u->v), trong đó u và v là hai điểm liên tiếp trong route, p là điểm cần chèn. Điểm có chi phí tăng thêm nhỏ nhất sẽ được chèn vào vị trí tối ưu. Quá trình này lặp lại cho đến khi tất cả các điểm đều được gán vào route.

Giai đoạn 2 - Simulated Annealing (Tinh chỉnh để đạt tối ưu):

Sau khi có route ban đầu từ Cheapest Insertion, thuật toán Simulated Annealing được áp dụng để cải thiện route. Thuật toán này sử dụng nguyên lý "nhiệt độ" giảm dần: bắt đầu với nhiệt độ cao (chấp nhận cả những thay đổi làm tăng chi phí), sau đó giảm dần nhiệt độ (chỉ chấp nhận những thay đổi tốt hơn). Các thao tác được thử nghiệm bao gồm: swap (đổi chỗ hai điểm), reverse (đảo ngược một đoạn route), và relocate (di chuyển một điểm sang vị trí khác). Mỗi thao tác được đánh giá bằng hàm chi phí: Cost = 0.7 \* Time + 0.3 \* Distance, trong đó Time và Distance được lấy từ ma trận OSRM hoặc tính bằng Haversine nếu OSRM không khả dụng.

Tối ưu hóa ma trận khoảng cách:

Để tăng tốc độ tính toán, hệ thống sử dụng OSRM (Open Source Routing Machine) để tính toán ma trận khoảng cách và thời gian di chuyển giữa các điểm. Ma trận này được cache để tránh tính toán lại. Nếu OSRM không khả dụng, hệ thống fallback về công thức Haversine để tính khoảng cách đường chim bay.

#### 3.5.2 Thuật toán phân cụm điểm thu gom (Clustering)

Hệ thống sử dụng thuật toán K-Means và Sweep Line để phân cụm các điểm thu gom theo khu vực địa lý:

K-Means Clustering:

Thuật toán K-Means được sử dụng để chia các điểm thu gom thành K cụm dựa trên tọa độ địa lý. Mỗi cụm sẽ được gán cho một xe thu gom. Thuật toán bắt đầu bằng việc chọn K điểm làm tâm cụm ban đầu (có thể random hoặc dựa trên vị trí depot), sau đó gán mỗi điểm vào cụm gần nhất, cập nhật tâm cụm, và lặp lại cho đến khi hội tụ.

Sweep Line Algorithm:

Đối với các điểm phân bố theo đường thẳng (ví dụ: dọc theo một con đường), hệ thống sử dụng thuật toán Sweep Line để tạo các cụm liên tục. Thuật toán này quét một đường thẳng (sweep line) qua không gian, và các điểm nằm gần đường quét sẽ được nhóm lại thành một cụm.

#### 3.5.3 Thuật toán dự đoán nhu cầu (Predictive Analytics)

Hệ thống tích hợp hai mô hình dự đoán:

ARIMA (AutoRegressive Integrated Moving Average):

Khi có đủ dữ liệu lịch sử (≥14 ngày), hệ thống sử dụng mô hình ARIMA để dự đoán nhu cầu thu gom trong tương lai. ARIMA là mô hình time-series forecasting mạnh mẽ, có khả năng xử lý xu hướng (trend) và tính mùa vụ (seasonality) trong dữ liệu. Mô hình được khởi tạo với tham số auto để tự động chọn bộ tham số (p,d,q) tối ưu. Quá trình training sử dụng Maximum Likelihood Estimation với Nelder-Mead optimizer.

Linear Regression (Fallback):

Nếu ARIMA không khả dụng hoặc dữ liệu không đủ, hệ thống fallback về Simple Linear Regression. Mô hình này giả định mối quan hệ tuyến tính giữa thời gian và lượng rác: y = slope \* x + intercept, trong đó x là số ngày từ ngày đầu tiên, y là lượng rác dự đoán. Hệ số slope và intercept được tính bằng phương pháp Least Squares.

#### 3.5.4 Thuật toán tìm đường ngắn nhất (A\* Search)

Để tính toán lộ trình chi tiết giữa hai điểm, hệ thống sử dụng thuật toán A\* (A-star) với heuristic là khoảng cách Haversine. Thuật toán A\* là sự kết hợp giữa Dijkstra (tìm đường chắc chắn tối ưu) và Greedy Best-First Search (tìm đường nhanh). Hàm đánh giá f(n) = g(n) + h(n), trong đó g(n) là chi phí thực tế từ điểm xuất phát đến n, và h(n) là ước lượng chi phí từ n đến đích (khoảng cách Haversine).

### 3.6 Phân tích ứng dụng Mã nguồn mở

Dự án EcoCheck sử dụng triệt để các công nghệ mã nguồn mở, đảm bảo tính minh bạch, khả năng tùy biến và không phụ thuộc vào nhà cung cấp độc quyền:

**Backend:**

* **Node.js:** Runtime JavaScript mã nguồn mở (MIT License)
* **Express.js:** Framework web nhẹ và linh hoạt (MIT License)
* **Socket.IO:** Thư viện real-time communication (MIT License)
* **PostgreSQL:** Hệ quản trị cơ sở dữ liệu quan hệ mã nguồn mở (PostgreSQL License)
* **PostGIS:** Extension không gian địa lý cho PostgreSQL (GPL v2)
* **TimescaleDB:** Extension time-series cho PostgreSQL (Apache 2.0)
* **Redis:** In-memory data store (BSD 3-Clause)
* **Orion-LD:** FIWARE Context Broker (Apache 2.0)

**Frontend Web:**

* **React:** Thư viện UI mã nguồn mở (MIT License)
* **Vite:** Build tool hiện đại (MIT License)
* **MapLibre GL:** Thư viện bản đồ mã nguồn mở (ISC License)

**Frontend Mobile:**

* **Flutter:** Framework đa nền tảng của Google (BSD 3-Clause)

**Công cụ phát triển:**

* **Docker:** Containerization platform (Apache 2.0)
* **Git:** Hệ thống quản lý phiên bản phân tán (GPL v2)

Tất cả các công nghệ này đều có giấy phép tương thích với MIT License của dự án, đảm bảo không có xung đột về bản quyền.

### 3.7 Phân tích ứng dụng Trí tuệ nhân tạo (AI)

Dự án EcoCheck tích hợp AI ở nhiều tầng khác nhau:

1. Predictive Analytics với ARIMA:

Như đã phân tích ở mục 3.5.3, hệ thống sử dụng mô hình ARIMA để dự đoán nhu cầu thu gom. Mô hình này được train trên dữ liệu lịch sử và tự động điều chỉnh tham số để phù hợp với pattern của dữ liệu. Độ chính xác của mô hình được đánh giá thông qua Mean Absolute Percentage Error (MAPE), và hệ thống tự động fallback về Linear Regression nếu ARIMA không đạt yêu cầu.

2. Route Optimization với Metaheuristics:

Thuật toán Simulated Annealing được sử dụng trong tối ưu hóa lộ trình là một kỹ thuật AI metaheuristic, cho phép tìm nghiệm gần tối ưu cho bài toán NP-hard như VRP trong thời gian hợp lý. Thuật toán này sử dụng nguyên lý "chấp nhận xác suất" để tránh bị kẹt ở cực tiểu địa phương.

3. Hướng phát triển AI trong tương lai:

Dự án đã lên kế hoạch mở rộng với các tính năng AI nâng cao:

* **Computer Vision:** Sử dụng MobileNet/EfficientNet để tự động phân loại rác từ ảnh check-in, nhận diện waste\_type và filling\_level.
* **Anomaly Detection:** Sử dụng Isolation Forest hoặc Autoencoder để phát hiện bất thường trong hoạt động (xe đi lệch route, dừng quá lâu).
* **Smart Scheduling:** Recommendation System dựa trên lịch sử, pattern, và thời tiết để đề xuất lịch thu gom tối ưu.
* **Fraud Detection:** Image similarity detection và pattern analysis để phát hiện check-in gian lận.

### 3.8 Luồng hoạt động cốt yếu của hệ thống

#### 3.8.1 Luồng Check-in và Tạo Yêu cầu Thu gom

**Bước 1:** Người dân mở ứng dụng EcoCheck User, chọn chức năng "Check-in rác thải".

**Bước 2:** Ứng dụng tự động lấy vị trí GPS hiện tại của người dân (sử dụng Geolocation API của Flutter).

**Bước 3:** Người dân chụp ảnh điểm rác và nhập thông tin (loại rác, khối lượng ước tính).

**Bước 4:** Ứng dụng gửi POST request đến /api/rt/checkin với dữ liệu JSON:

{

"lat": 10.8231,

"lon": 106.6297,

"waste\_type": "household",

"estimated\_weight": 5.2,

"image\_url": "uploads/checkin\_123.jpg"

}

**Bước 5:** Backend nhận request, thực hiện các bước:

* Lưu check-in vào bảng checkins trong PostgreSQL
* Tạo hoặc cập nhật điểm thu gom trong bảng points (sử dụng PostGIS để lưu geometry)
* Tạo entity NGSI-LD trong Orion-LD Context Broker
* Phát sóng sự kiện qua Socket.IO để Web Dashboard cập nhật real-time
* Tính điểm gamification cho người dân

**Bước 6:** Backend trả về response với thông tin điểm đã được tạo và điểm thưởng.

**Bước 7:** Ứng dụng hiển thị thông báo thành công và cập nhật điểm số của người dân.

#### 3.8.2 Luồng Tối ưu hóa Lộ trình và Phân công Nhiệm vụ

**Bước 1:** Quản lý mở Web Dashboard, chọn khu vực và ngày cần tối ưu lộ trình.

**Bước 2:** Dashboard gửi POST request đến /api/vrp/optimize với danh sách điểm thu gom và thông tin xe.

**Bước 3:** Backend thực hiện các bước:

* Truy vấn tất cả điểm thu gom chưa được gán trong khu vực từ database
* Lấy thông tin xe và nhân viên available
* Gọi hàm optimizeRouteHybrid() để tính toán lộ trình tối ưu:  
   - Phân cụm điểm bằng K-Means hoặc Sweep Line  
   - Với mỗi cụm, áp dụng Cheapest Insertion để tạo route ban đầu  
   - Áp dụng Simulated Annealing để tinh chỉnh route  
   - Tính toán ma trận khoảng cách bằng OSRM (hoặc Haversine fallback)
* Lưu route vào bảng routes và route\_stops trong database
* Tạo entity NGSI-LD cho route trong Orion-LD

**Bước 4:** Backend gửi thông báo qua Socket.IO đến Worker App của tài xế được phân công.

**Bước 5:** Worker App nhận thông báo, hiển thị lộ trình mới và danh sách điểm cần thu gom.

**Bước 6:** Tài xế xác nhận nhận nhiệm vụ, route chuyển sang trạng thái "assigned".

#### 3.8.3 Luồng Thu gom Thực tế và Cập nhật Trạng thái

**Bước 1:** Tài xế bắt đầu ca làm việc, mở Worker App và chọn route đã được phân công.

**Bước 2:** Ứng dụng hiển thị danh sách điểm cần thu gom dưới dạng "Smart Checklist" (không phải GPS navigation liên tục).

**Bước 3:** Khi tài xế đến gần điểm thu gom (dựa trên GPS), ứng dụng phát thông báo giọng nói: "300 mét nữa, rẽ trái vào hẻm 5. Có 3 điểm thu gom."

**Bước 4:** Tài xế thực hiện thu gom rác tại điểm, chụp ảnh xác nhận và cập nhật trạng thái "completed" trong ứng dụng.

**Bước 5:** Ứng dụng gửi POST request đến /api/routes/:id/complete-stop với thông tin điểm đã hoàn thành.

**Bước 6:** Backend cập nhật database:

* Cập nhật trạng thái route\_stop sang "completed"
* Cập nhật tải trọng hiện tại của xe
* Cập nhật entity NGSI-LD trong Orion-LD
* Phát sóng sự kiện qua Socket.IO

**Bước 7:** Web Dashboard tự động cập nhật bản đồ real-time, hiển thị điểm đã hoàn thành và vị trí hiện tại của xe.

**Bước 8:** Khi hoàn thành tất cả điểm, tài xế di chuyển đến bãi đổ rác (dump), cập nhật trạng thái route là "completed".

#### 3.8.4 Luồng Giám sát Real-time và Phát hiện Sự cố

**Bước 1:** Web Dashboard kết nối với Backend qua WebSocket khi mở trang.

**Bước 2:** Backend duy trì một in-memory store (sử dụng module realtime.js) để lưu trữ trạng thái real-time của tất cả xe và route.

**Bước 3:** Worker App gửi cập nhật vị trí định kỳ (mỗi 5-10 giây) qua WebSocket khi đang trong ca làm việc.

**Bước 4:** Backend nhận cập nhật vị trí, thực hiện:

* Cập nhật in-memory store
* Lưu vào bảng vehicle\_tracking (TimescaleDB) để phân tích sau
* Phát sóng sự kiện vehicle:location:update qua Socket.IO

**Bước 5:** Web Dashboard nhận sự kiện, cập nhật marker trên bản đồ để hiển thị vị trí real-time của xe.

**Bước 6:** Hệ thống tự động phát hiện sự cố:

* Nếu xe đi lệch route quá xa (>500m), tạo alert "route\_deviation"
* Nếu xe dừng quá lâu tại một điểm (>30 phút), tạo alert "long\_stop"
* Nếu điểm thu gom bị bỏ sót (đã qua thời gian dự kiến nhưng chưa completed), tạo alert "missed\_point"

**Bước 7:** Quản lý nhận thông báo alert trên Dashboard, có thể gán xe khác để xử lý sự cố (Dynamic Dispatch).

### 3.9 Mối liên hệ giữa các thành phần trong hệ thống

Hệ thống EcoCheck được thiết kế với kiến trúc phân tầng rõ ràng, mỗi tầng có trách nhiệm riêng biệt nhưng liên kết chặt chẽ với nhau:

**Tầng Presentation (Giao diện người dùng):**

* Mobile Apps (Flutter) và Web Dashboard (React) là điểm tiếp xúc với người dùng
* Giao tiếp với Backend qua RESTful API và WebSocket
* Không chứa logic nghiệp vụ, chỉ hiển thị và thu thập dữ liệu

**Tầng Application (Logic nghiệp vụ):**

* Backend API (Node.js/Express) xử lý tất cả logic nghiệp vụ
* Các service như RouteOptimizer, DispatchService, AnalyticsService được tổ chức theo mô hình Service-Oriented Architecture
* Giao tiếp với Database và Context Broker

**Tầng Data (Dữ liệu):**

* PostgreSQL/PostGIS lưu trữ dữ liệu quan hệ và không gian
* TimescaleDB tối ưu hóa dữ liệu chuỗi thời gian
* Orion-LD Context Broker lưu trữ dữ liệu ngữ nghĩa (NGSI-LD)
* Redis cache dữ liệu thường truy cập

**Tầng Infrastructure (Hạ tầng):**

* Docker containers đảm bảo tính nhất quán giữa các môi trường
* Nginx (trong production) đóng vai trò reverse proxy và load balancer

Mối liên hệ giữa các thành phần được thể hiện qua các luồng dữ liệu:

* **Luồng dữ liệu người dùng → hệ thống:** Mobile/Web → Backend API → Database/Orion-LD
* **Luồng dữ liệu hệ thống → người dùng:** Database/Orion-LD → Backend API → Socket.IO → Mobile/Web
* **Luồng xử lý nghiệp vụ:** Backend API → Service Layer → Database/Orion-LD → Response

### 3.10 Thiết kế giao diện người dùng

#### 3.10.1 Nguyên lý thiết kế UI/UX

Nguyên lý thiết kế UI/UX của EcoCheck tập trung vào sự đơn giản, trực quan và dễ sử dụng. Ứng dụng Mobile dành cho tài xế ưu tiên các nút bấm lớn, thông tin hiển thị rõ ràng để dễ thao tác khi đang di chuyển. Dashboard Web sử dụng layout dạng lưới nhiều cột để hiển thị tối đa thông tin giám sát trên một màn hình. Màu sắc chủ đạo được chọn là Xanh lá (Eco Green) tạo cảm giác thân thiện với môi trường.

#### 3.10.2 Wireframes và Mockups

Giao diện người dùng được phát triển dựa trên các bản Wireframe phác thảo luồng đi của người dùng. Các màn hình chính như "Danh sách nhiệm vụ thu gom" (Task List), "Bản đồ dẫn đường" (Navigation Map), "Báo cáo sự cố" (Report Dialog) và "Bảng điều khiển quản trị" (Admin Dashboard) đều được thiết kế chi tiết để đảm bảo trải nghiệm người dùng liền mạch và nhất quán trên toàn hệ thống.

## CHƯƠNG 4: PHÁT TRIỂN HỆ THỐNG VÀ ĐỊNH HƯỚNG TƯƠNG LAI

### 4.1 Tổng kết dự án

Dự án EcoCheck đã hoàn thành xuất sắc mục tiêu xây dựng một hệ sinh thái phần mềm toàn diện phục vụ công tác thu gom rác thải thông minh. Hệ thống đã hiện thực hóa thành công việc ứng dụng công nghệ Dữ liệu mở liên kết (LOD) vào quy trình vận hành thực tế, giải quyết được bài toán tối ưu hóa lộ trình thu gom và kết nối cộng đồng người dân với đơn vị quản lý môi trường.

### 4.2 Thành tựu và Sản phẩm bàn giao

Các kết quả đạt được của dự án bao gồm: Bộ mã nguồn hoàn chỉnh và chất lượng cao cho Backend, Web Manager và 2 ứng dụng Mobile (User & Worker); Bộ tài liệu hướng dẫn cài đặt và phát triển chi tiết (Documentation); Hệ thống kịch bản kiểm thử tự động; và Bộ dữ liệu mẫu chuẩn NGSI-LD. Sản phẩm đã chạy ổn định và mượt mà trong môi trường giả lập Docker, chứng minh được tính khả thi và hiệu quả của giải pháp đề xuất.

#### 4.2.1 Kết quả Demo các kịch bản chính

Hệ thống đã được kiểm thử thành công trên môi trường local với các kịch bản thực tế:

**Kịch bản 1 - Người dân báo cáo:** Ứng dụng User gửi thành công request kèm tọa độ và ảnh rác thải. Hệ thống backend ghi nhận sự kiện và tạo một entity NGSI-LD mới.

**Kịch bản 2 - Tối ưu lộ trình (VRP):** Với đầu vào là 10 điểm thu gom rải rác, thuật toán VRP (Vehicle Routing Problem) đã tính toán và trả về lộ trình tối ưu giảm 15-20% tổng quãng đường so với di chuyển tuần tự.

**Kịch bản 3 - Giám sát thời gian thực:** Vị trí của nhân viên thu gom (Worker) được cập nhật real-time trên bản đồ Dashboard quản lý thông qua Socket.IO.

### 4.3 Thách thức và Giải pháp

Thách thức lớn nhất mà nhóm gặp phải là việc tích hợp và đồng bộ dữ liệu thời gian thực giữa hàng loạt thiết bị di động và máy chủ trung tâm, cũng như việc làm quen và áp dụng chuẩn dữ liệu mới NGSI-LD. Nhóm đã giải quyết vấn đề này bằng cách sử dụng Socket.IO cho giao tiếp hai chiều độ trễ thấp và xây dựng các lớp chuyển đổi dữ liệu (Data Mapper) thông minh để đảm bảo tính tương thích giữa cơ sở dữ liệu quan hệ truyền thống và định dạng Linked Data mới.

Một thách thức khác là việc tối ưu hóa hiệu năng của thuật toán VRP khi số lượng điểm thu gom lớn (>100 điểm). Nhóm đã giải quyết bằng cách:

* Sử dụng phân cụm (clustering) để chia nhỏ bài toán
* Cache ma trận khoảng cách OSRM
* Áp dụng Simulated Annealing với số lần lặp có giới hạn để đảm bảo thời gian phản hồi < 5 giây

### 4.4 Bài học kinh nghiệm

Qua quá trình thực hiện dự án, nhóm nghiên cứu đã rút ra nhiều bài học quý giá về kỹ thuật quản lý trạng thái trong ứng dụng đa nền tảng với Flutter BLoC, kỹ thuật tối ưu hóa truy vấn không gian với PostGIS, và tầm quan trọng của việc thiết kế kiến trúc hệ thống hướng dịch vụ (Microservices) ngay từ giai đoạn đầu để đảm bảo khả năng mở rộng và bảo trì hệ thống về sau.

### 4.5 Cải tiến trong tương lai

Trong tương lai, EcoCheck định hướng tích hợp sâu hơn các công nghệ Trí tuệ nhân tạo (AI) và Học máy (Machine Learning) để dự báo lượng rác thải phát sinh tại từng khu vực dựa trên dữ liệu lịch sử check-in. Hệ thống cũng sẽ mở rộng khả năng kết nối với các thiết bị IoT (như cảm biến mức đầy thùng rác) để tự động hóa hoàn toàn quy trình thu thập dữ liệu đầu vào, giảm thiểu sự phụ thuộc vào thao tác thủ công của con người.

#### 4.5.1 Mô hình Hybrid (Lai) - Sự thỏa hiệp thông minh

Thay vì thay thế hoàn toàn hệ thống thu gom truyền thống, EcoCheck được thiết kế để hoạt động như một lớp tối ưu (Optimization Layer) phủ lên hệ thống cũ. Cụ thể:

* **Rác sinh hoạt hàng ngày (hữu cơ):** Có thể vẫn giữ cố định 80% (vì ngày nào cũng có)
* **Rác biến động (20%):** Rác tái chế, rác cồng kềnh, rác ít/nhiều thất thường cần DWC để xử lý
* **Lợi ích:** Giảm thiểu rủi ro, dễ triển khai từng bước, không gây xáo trộn lớn

#### 4.5.2 Tính năng AI nâng cao

**Python Microservice với Prophet:**

* Tạo Python microservice riêng (FastAPI) với Facebook Prophet cho time series forecasting
* Độ chính xác cao hơn (90-95% vs 60-70% của simple regression)
* Xử lý seasonality (tuần, tháng, năm), trends, và changepoints tự động
* Cung cấp confidence intervals (upper/lower bounds)

**Computer Vision:**

* Sử dụng MobileNet/EfficientNet để tự động phân loại rác từ ảnh check-in
* Tự động nhận diện waste\_type và filling\_level
* Phát hiện ảnh không hợp lệ

**Anomaly Detection:**

* Isolation Forest hoặc Autoencoder
* Phát hiện xe đi lệch route, dừng quá lâu
* Phát hiện check-in bất thường

### 4.6 Kiến nghị

Để dự án có thể triển khai rộng rãi và mang lại hiệu quả thực tế cao nhất, nhóm kiến nghị cần có sự hỗ trợ về mặt chính sách từ chính quyền để mở các nguồn dữ liệu hạ tầng đô thị. Đồng thời, cần có các chương trình tuyên truyền, giáo dục để nâng cao ý thức người dân trong việc sử dụng ứng dụng công nghệ để phân loại và báo cáo rác thải, biến EcoCheck thành một nền tảng cộng đồng thực thụ vì một thành phố xanh - sạch - đẹp.

### 4.7 Đóng góp xây dựng nguồn Dữ liệu mở cho Việt Nam

Một trong những đóng góp quan trọng của dự án là việc tạo lập bộ dữ liệu mẫu (Dataset) chuẩn hóa cho bài toán quản lý rác thải tại Việt Nam, bao gồm:

* **Bộ dữ liệu Depots & Dumps:** Vị trí các trạm trung chuyển và bãi rác mô phỏng tại TP.HCM, định dạng JSON-LD.
* **Bộ dữ liệu Vehicles:** Thông số kỹ thuật của các loại xe thu gom rác phổ biến.
* **Context File (ecocheck.jsonld):** File định nghĩa ngữ nghĩa giúp cộng đồng lập trình viên Việt Nam dễ dàng tái sử dụng các mô hình dữ liệu này cho các dự án Smart City khác.

Toàn bộ dữ liệu này được công khai trong thư mục seeds/ngsi-ld của mã nguồn dự án, tuân thủ nguyên tắc "Dữ liệu mở" (Open Data) và có thể được sử dụng tự do bởi các dự án khác.