

THE UNIVERSITY OF ZAMBIA
SCHOOL OF NATURAL AND APPLIED SCIENCES
DEPARTMENT OF COMPUTING AND INFORMATICS

FINAL YEAR PROJECT REPORT

ACADEMIC YEAR 2026

**Road Setback Management System: A Web-Based GIS Tool for
Screening Building Setback Compliance along Roads in Lusaka**

A Final Year Project Report Submitted to the Department of Computing and Informatics in
partial fulfilment of the requirements for the award of

Bachelor of Science in Computer Science

Student Name: KUPALWA CHIBUYE **2021424278**

ELIJAH ZULU **2021385299**

LUBASI SIPANGULE **2022065786**

Supervisor: Dr. LIGHTON PHIRI

Date of Submission: 25th September 2026

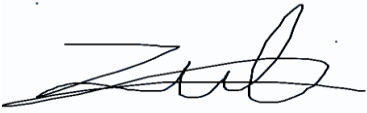
Declaration

I declare that this final year project report is my own work and has been prepared in accordance with the academic requirements of the University of Zambia. All sources of information, ideas, data, software, images, and other material that are not my own have been appropriately acknowledged and referenced.

Where this project was undertaken as part of a group, I have clearly identified my individual contributions and have not presented the work of other group members as my own.

Student Name: **Elijah Zulu**

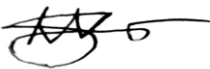
Computer Number: **2021385299**

Signature: 

Date: **22nd October 2026**

Student Name: **Lubasi Sipangule**

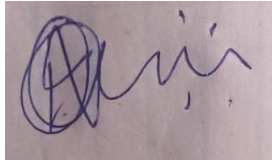
Computer Number: **2022065786**

Signature: 

Date: **22nd October 2026**

Student Name: **KUPALWA CHIBUYE**

Computer Number: **2021424278**

A handwritten signature in blue ink on a light brown background. The signature is stylized, starting with a large, circular, scribbled initial followed by several loops and a final upward stroke.

Signature:

Date: **22nd October 2026**

Approval

This final year project report has been submitted for examination with the approval of the project supervisor.

Supervisor Name: **Dr Lighton Phiri**

Signature: _____

Date: **22nd October 2026**

Departmental/Examiner Approval (if applicable):

Name: _____

Signature: _____

Date: _____

Abstract

Road reserves and building lines protect the space that roads need for future widening, drainage, utilities, sight distance and safety. In Lusaka, rapid urban growth has made encroachment of buildings into these setbacks a persistent development-control problem, while enforcement still depends largely on manual site inspection, paper records and case-by-case interpretation of the applicable distance. This project set out to design and implement a web-based Geographic Information System (GIS) tool that helps enforcement officers screen buildings along a selected road for possible setback violations.

An iterative, Agile-inspired development approach was followed. The resulting Road Setback Management System consists of a Next.js web client with a Leaflet map, an Express/Node.js GIS service, and a Supabase (PostgreSQL/PostGIS) database. Setback rules are held as configuration data rather than code: each OpenStreetMap road class is mapped to a regulatory class with a default distance (30.48 m, 18.288 m or 9.144 m), which can then be overridden by road-wide assignments and by structure, parcel or road-segment exceptions through a deterministic precedence procedure. Candidate buildings near the selected road are retrieved with a PostGIS spatial query, the shortest distance between each footprint and the road centreline is measured with Turf.js, and each building is classified as compliant, non-compliant or unknown. Every result records the rule that produced it and is flagged for human verification. Inspection runs are stored with an audit history, and administrators can review candidate status and manage user roles under row-level security.

The system was evaluated with 31 automated test cases covering rule resolution, geometric measurement, compliance evaluation and end-to-end candidate assessment using synthetic geometry with known distances. Twenty-nine tests passed and two exposed genuine defects (an exception-precedence tie check and an obsolete module entry point), which are documented together with further issues found through code review. The assessment engine processed 1,000 candidate buildings in about 0.33 s (≈ 0.33 ms per building). The project shows that configurable, explainable, open-source GIS screening is a practical first step towards more consistent and efficient road setback enforcement, provided that its outputs are treated as screening results that require field verification.

Keywords: road setback; building line; development control; GIS; PostGIS; OpenStreetMap; rule engine; web mapping

Acknowledgements

We would like to express our sincere gratitude to our supervisor, Dr Lighton Phiri, for his guidance, constructive criticism and patience throughout this project. His feedback shaped the direction of the work and the way it has been documented and evaluated.

We are grateful to the Department of Computing and Informatics at the University of Zambia and to the lecturers whose teaching in software engineering, databases, web development and human-computer interaction gave us the foundation for this work. We also thank one another for the commitment and cooperation shown throughout. The system is the product of three people working to a shared design, and each part of it depended on the others.

We further thank the seventeen respondents who took the time to use the system and complete the usability questionnaire, the OpenStreetMap contributors whose open data made the screening possible, and the developers of the open-source tools on which the system is built. Finally, we thank our families and friends for their encouragement and support throughout our studies.

Table of Contents

Declaration.....	1
Approval	3
Abstract	4
Acknowledgements.....	5
Table of Contents.....	6
List of Figures	11
List of Tables	12
List of Abbreviations and Acronyms	13
Chapter 1: Introduction	1
1.1 Background and Context.....	1
1.2 Problem Statement.....	1
1.3 Project Aim	2
1.4 Project Objectives	2
1.5 Research Questions	2
1.6 Scope of the Project	3
1.6.1 In Scope	3
1.6.2 Out of Scope	3
1.7 Significance of the Project.....	3
1.8 Limitations	4
1.9 Organisation of the Report.....	4
Chapter 2: Literature Review.....	5
2.1 Introduction.....	5
2.2 Key Concepts and Theories	5
2.2.1 Road Reserves, Building Lines and Setbacks.....	5
2.2.2 Geographic Information Systems and Spatial Analysis.....	5
2.2.3 Volunteered Geographic Information and OpenStreetMap	6
2.2.4 Spatial Data Formats, Databases and Web Mapping.....	6
2.3 Existing Systems and Related Work.....	6
2.3.1 Manual Field Inspection	6
2.3.2 Desktop GIS Analysis.....	6
2.3.3 Web-Based and Enterprise GIS Platforms.....	7
2.3.4 Automatically Extracted Building Footprints.....	7
2.4 Comparative Analysis.....	7
2.5 Research and Knowledge Gap	8
2.6 Conceptual Framework.....	8

2.7 Chapter Summary	8
Chapter 3: Methodology	9
3.1 Introduction.....	9
3.2.1 Research Paradigm.....	9
3.2.2 Influence of Prior Studies	9
3.3 Development Approach	10
3.3.1 Scrum as the Selected Agile Framework	10
3.4 Requirements and Data Collection	11
3.5 System Architecture.....	11
3.6 Tools and Technologies	12
3.7 Ethical and Legal Considerations	13
3.8 Evaluation Method.....	13
3.9 Chapter Summary	14
Chapter 4: System Analysis and Design	15
4.1 Introduction.....	15
4.2 Stakeholder and User Analysis	15
4.3 Functional Requirements	15
4.4 Non-Functional Requirements	16
4.5 Use Cases	16
4.6 System Architecture.....	18
4.7 Database Design.....	18
4.8 Interface Design	20
4.9 Algorithms and Technical Design	23
4.9.1 Rule Resolution.....	23
4.9.2 Candidate Retrieval.....	24
4.9.3 Distance Measurement.....	24
4.9.4 Compliance Evaluation.....	25
4.10 Chapter Summary	25
Chapter 5: Implementation	26
5.1 Introduction.....	26
5.2 Development Environment	26
5.3 System Components and Modules.....	26
5.4 Key Implementation Details	27
5.4.1 Rules as Configuration.....	27
5.4.2 Road Assessment	28
5.4.3 Distance Measurement.....	28
5.4.4 Persisting Inspections.....	28

5.4.5 Role-Based Access Control	28
5.5 Integration	29
5.6 Challenges and Solutions.....	29
5.7 Individual Contribution.....	30
5.7.1 Elijah Zulu: Database, Authentication, Access Control and Deployment.....	31
5.7.2 Lubasi Sipangule: Backend Development and Geospatial Screening Engine.....	32
5.7.3 Kupalwa Chibuye: Interface Design, Dashboard and Usability Evaluation.....	38
5.7.4 System Integration and Collaborative Development	39
5.8 Chapter Summary	40
Chapter 6: Testing, Results and Evaluation	41
6.1 Introduction.....	41
6.2 Testing Strategy	41
6.3 Test Cases and Results.....	42
6.3.1 Automated rule-engine tests	42
6.3.2 System-level tests on the deployed application	44
6.4 Experimental Results / Performance Results	44
6.5 User Evaluation.....	45
6.6 Analysis of Results	47
6.7 Objective-by-Objective Evaluation	48
6.8 Individual Results (group projects).....	48
6.8.1 Elijah Zulu	48
6.8.2 Lubasi Sipangule.....	49
6.8.3 Kupalwa Chibuye.....	51
6.9 Chapter Summary	51
Chapter 7: Discussion	52
7.1 Introduction.....	52
7.2 Interpretation of Findings	52
7.3 Comparison with Existing Work	53
7.4 Contributions of the Project	54
7.5 Limitations and Implications	54
7.6 Chapter Summary	55
Chapter 8: Conclusion and Recommendations	56
8.1 Introduction.....	56
8.2 Summary of the Project	56
8.3 Conclusions.....	56
8.4 Achievement of Objectives.....	57
8.5 Recommendations.....	57

8.6 Future Work.....	58
References.....	59
Appendix A: Individual Contribution Statement (Group Projects)	60
A.1 Project and Group Information	60
A.2 Overall Group Project	60
A.3 Individual Contribution Statement: Elijah Zulu.....	61
A.3.1 Individual Role and Responsibilities	61
A.3.2 Detailed Individual Contribution	61
A.3.3 Individual Deliverables	62
A.3.4 Contribution to the Main System or Research Output.....	62
A.3.5 Individual Implementation and Technical Work	62
A.3.6 Individual Testing and Evaluation Contribution.....	63
A.3.7 Collaboration with Other Group Members	64
A.3.8 Contribution Summary.....	64
A.3.9 Estimated Contribution	64
A.3.10 Statement of Authorship and Contribution	65
A.3.11 Supervisor Verification.....	65
A.4 Individual Contribution Statement: Lubasi Sipangule.....	65
A.4.1 Individual Role and Responsibilities	65
A.4.2 Detailed Individual Contribution	66
A.4.3 Individual Deliverables	67
A.4.4 Contribution to the Main System or Research Output.....	67
A.4.5 Individual Implementation and Technical Work	67
A.4.6 Individual Testing and Evaluation Contribution.....	68
A.4.7 Collaboration with Other Group Members	68
A.4.8 Contribution Summary.....	69
A.4.9 Estimated Contribution	69
A.4.10 Statement of Authorship and Contribution	70
A.4.11 Supervisor Verification.....	70
A.5 Individual Contribution Statement: Kupalwa Chibuye.....	70
A.5.1 Individual Role and Responsibilities	70
A.5.2 Detailed Individual Contribution	70
A.5.3 Individual Deliverables	71
A.5.4 Contribution to the Main System or Research Output.....	71
A.5.5 Individual Implementation and Technical Work	72
A.5.6 Individual Testing and Evaluation Contribution.....	72
A.5.7 Collaboration with Other Group Members	72

A.5.8 Contribution Summary.....	73
A.5.9 Estimated Contribution	73
A.5.10 Statement of Authorship and Contribution	74
A.5.11 Supervisor Verification	74
Appendix B: User Manual / Installation Guide	75
B.1 Using the System.....	75
B.2 Installing and Running the System Locally	76
B.3 Troubleshooting	77
Appendix C: Additional System Designs / Diagrams	78
C.1 Screening Statuses.....	78
C.2 GIS Service API Specification	78
C.3 Additional Database Tables	79
Appendix D: Additional Test Cases and Results	81
D.1 Rule-Engine Test Cases	81
D.2 Code Review Findings	82
Appendix E: Data Collection Instruments	84
E.1 System Usability Scale Questionnaire.....	84
E.2 System Usability Scale Results	87
Appendix F: Additional Code / Configuration	92
F.1 Standard Road Controls (config/setback-rules.json, excerpt)	92
F.2 Zone Assessment of a Candidate (src/detection/assessCandidate.js, simplified).....	92
F.3 Database Policies and Functions (excerpts)	92
Appendix G: Other Supporting Material	94
G.1 Deployment and Repository.....	94
G.2 Data Sources and Licensing.....	94
G.3 Rule Resolution Order	94
Appendix A: Individual Contribution Statement (Group Projects)	46
Appendix B: User Manual / Installation Guide	48
Appendix C: Additional System Designs / Diagrams	51
Appendix D: Additional Test Cases and Results	53
Appendix E: Data Collection Instruments	56
Appendix F: Additional Code / Configuration	63
Appendix G: Other Supporting Material	65

List of Figures

Figure 1: System architecture of the Road Setback Management System	12
Figure 2: Use case diagram.....	17
Figure 3: Sequence diagram for screening a road.....	18
Figure 4: Entity-relationship diagram	19
Figure 5: report page	21
Figure 6: Dashboard showing a screened road, its setback zone and candidate buildings	22
Figure 7: Administrator panel with user and candidate management.....	22
Figure 8: Rule resolution and compliance evaluation pipeline.....	24
Figure 9: Distribution of responses to the SUS items.....	46
Figure 10: SUS Questionnaire	84
Figure 11: SUS Questions 1.....	85
Figure 12: SUS Questions 2.....	87
Figure 13: SUS Questions 3.....	87
Figure 14: SUS Responses 1.....	88
Figure 15: SUS Responses 2.....	89
Figure 16: SUS Responses 3.....	91
Figure 17: SUS Responses 4.....	91

List of Tables

Table 1: Comparison of approaches to detecting road setback violations	7
Table 2: Development iterations and deliverables	10
Table 3: Tools and technologies used	12
Table 4: Stakeholders and their interests	15
Table 5: Functional requirements	15
Table 6: Non-functional requirements	16
Table 7: Use case description: Run setback screening	17
Table 8: Database tables	19
Table 9: Configured road classes and default road reserve distances.....	23
Table 10: Main components and modules	26
Table 11: Implementation challenges and how they were addressed	29
Table 12: Summary of automated rule-engine test results.....	42
Table 13: Representative rule-engine test cases	42
Table 14: Analysis of the failed test cases	43
Table 15: System-level test cases and outcomes	44
Table 16: SUS responses by item (n = 17; 1 = strongly disagree, 5 = strongly agree).....	45
Table 17: Evaluation of the project objectives	48
Table 18: Project and group information	60
Table 19: Environment variables	76
Table 20: Screening statuses returned by the GIS service	78
Table 21: Database tables added in the final migration	79
Table 22: Complete list of rule-engine test cases and results	81
Table 23: Defects and risks identified by code review	82

List of Abbreviations and Acronyms

Abbreviation	Meaning
API	Application Programming Interface
CORS	Cross-Origin Resource Sharing
CRS	Coordinate Reference System
DBMS	Database Management System
ERD	Entity-Relationship Diagram
GeoJSON	Geographic JavaScript Object Notation (RFC 7946)
GIS	Geographic Information System
HTTP(S)	Hypertext Transfer Protocol (Secure)
JSON	JavaScript Object Notation
JWT	JSON Web Token
LCC	Lusaka City Council
ODbL	Open Database License
OSM	OpenStreetMap
PostGIS	Spatial extension for the PostgreSQL database
REST	Representational State Transfer
RLS	Row-Level Security
RPC	Remote Procedure Call
SQL	Structured Query Language
SSR	Server-Side Rendering
SUS	System Usability Scale
UI	User Interface
WGS 84	World Geodetic System 1984 (EPSG:4326)

Chapter 1: Introduction

1.1 Background and Context

Roads are among the most valuable pieces of public infrastructure in any city, and the land beside them is almost as important as the carriageway itself. Road reserves and building lines keep a strip of land free of permanent structures so that a road can later be widened, so that drains, power lines, water pipes and telecommunications ducts can be installed and maintained, and so that drivers and pedestrians have adequate sight lines. When buildings, walls or other structures are erected inside this protected strip, the cost of future road improvements rises sharply, often requiring demolition, compensation and lengthy disputes.

Lusaka is one of the fastest-growing cities in the region, and much of its growth has taken place faster than planning and development-control institutions could follow [1]. Lusaka City Council (LCC), designated as the local planning authority under Zambia's Urban and Regional Planning Act, 2015, is responsible for regulating land use, approving developments, and enforcing development controls and setbacks alongside relevant legislation such as the Public Roads Act, 2002 [2] [3]. In practice, detecting encroachment still relies heavily on officers physically visiting sites, measuring distances by hand and recording findings on paper. This process is slow, inconsistent between officers, and difficult to audit or repeat.

At the same time, open geospatial data and web technologies have matured considerably. OpenStreetMap (OSM) provides freely licensed road networks and building footprints for most Zambian cities including [4], spatial databases such as PostGIS can answer proximity questions over large datasets in milliseconds [5], and browser-based mapping libraries make it possible to deliver GIS capabilities to non-specialist users without desktop software [6]. These developments create an opportunity to support setback enforcement with a desk-based screening tool that points officers towards the buildings most likely to be in violation before any site visit is made.

1.2 Problem Statement

Setback enforcement along Lusaka's roads is currently reactive and labour-intensive. Officers have no quick way to establish, for a given road, which setback distance applies and which buildings are closer to the road than that distance. The applicable distance depends on the road's regulatory class and may be modified by road-specific decisions or by exceptions granted for

particular plots or structures, which makes manual determination error-prone. Measurements taken in the field are not stored in a structured, searchable form, so there is little institutional memory of which roads have been inspected, what was found, and how each case was resolved. As a result, violations are detected late, enforcement is inconsistent, and scarce inspection resources are not directed to where they are most needed. A tool is needed that can apply the correct setback rule consistently, measure building-to-road distances automatically from spatial data, and keep an auditable record of inspections for human review.

1.3 Project Aim

To design, implement and evaluate a web-based GIS system that screens buildings along a selected road for potential road setback violations using configurable regulatory rules, and that records the results for review by enforcement officers.

1.4 Project Objectives

1. To design a configurable rule engine that maps road classes to their default regulatory setback distances and supports road-wide rule assignments and structure-, parcel- and road-segment-specific exceptions with a deterministic order of precedence.
2. To implement a geospatial screening component that retrieves candidate buildings near a selected road and measures the shortest distance between each building footprint and the road, classifying each building as compliant, non-compliant or unknown.
3. To develop a web-based map interface through which authenticated officers can select a road, view the applicable setback zone and screened buildings, and consult their inspection history.
4. To implement role-based access control and an administrative workflow for reviewing candidate status and managing user roles.
5. To evaluate the correctness and performance of the system through systematic testing.

1.5 Research Questions

1. How can statutory road setback requirements, together with road-specific assignments and exceptions, be represented so that the applicable distance for any building can be resolved automatically and explainably?
2. How accurately and efficiently can building-to-road distances be computed from open vector data in a web-based system?

3. To what extent can a web-based screening tool support, rather than replace, the work of enforcement officers?

1.6 Scope of the Project

1.6.1 In Scope

- Screening of buildings along a single road segment selected by the user, within the Lusaka road and building datasets loaded into the database.
- Rule resolution based on OSM road classification, with configurable default distances, road-wide assignments and exceptions.
- Measurement of shortest distance from the road centreline to building footprints (points, lines and polygons).
- A web client with login and sign-up, an interactive map, a results table, inspection history and an administrator panel.
- Persistent storage of inspections and screened building candidates, protected by authentication and row-level security.

1.6.2 Out of Scope

- Making legal determinations or issuing enforcement notices; all outputs are screening results that require field verification.
- Side-specific (left/right) setback analysis, which the rule engine can represent but the measurement component does not yet evaluate.
- Measurement from the legally surveyed edge of the road reserve; the prototype measures from the road centreline.
- Integration with the cadastral (land parcel) register, the building-permit system or payment systems.
- Automatic extraction of building footprints from satellite or aerial imagery.

1.7 Significance of the Project

For **Lusaka City Council**, the system offers a way to prioritise inspections, apply setback rules consistently and keep an auditable record of what was screened and decided. For **enforcement officers**, it replaces repetitive manual measurement with a map-based screening step that takes seconds per road. For **road authorities and the public**, earlier detection of encroachment protects road reserves and reduces the future cost of road upgrades. Technically, the project demonstrates a "rules-as-data" approach in which regulatory distances, overrides and exceptions are configuration rather than code, so they can be updated as regulations change

without redeploying the software. The system is built entirely on open-source components and open data, which keeps adoption costs low for a public institution.

1.8 Limitations

- **Data quality.** OSM road and building data are volunteered and vary in completeness and positional accuracy missing or misplaced footprints directly affect results.
- **Reference geometry.** Distances are measured from the road centreline, which is an approximation of the legally relevant line.
- **Rule mapping.** The mapping from OSM road classes to regulatory classes is partly provisional; roads without a configured mapping fall back to the most permissive class and are flagged as requiring verification.
- **Evaluation.** Testing used synthetic geometry with known distances and code review. A structured user evaluation with LCC officers had not been completed at the time of writing.
- **Time and resources.** The project was completed within one academic year, which limited field validation and the implementation of side-specific analysis.

1.9 Organisation of the Report

Chapter 2 reviews the concepts, technologies and existing approaches relevant to setback enforcement and identifies the gap this project addresses. Chapter 3 describes the development methodology, data sources, tools and evaluation method. Chapter 4 presents the analysis and design of the system, including requirements, architecture, database and algorithms. Chapter 5 describes the implementation and the challenges encountered. Chapter 6 reports the testing strategy, test results and an objective-by-objective evaluation. Chapter 7 discusses the findings, and Chapter 8 concludes with recommendations and future work. The appendices contain the individual contribution statement, a user and installation guide, the API specification, additional test details, a proposed usability questionnaire and selected configuration.

Chapter 2: Literature Review

2.1 Introduction

This chapter introduces the concepts on which the project rests: road reserves and building lines, geographic information systems and spatial analysis, volunteered geographic information, and the web mapping and spatial database technologies used to build the system. It then reviews existing approaches to detecting setback violations, compares them against criteria relevant to a municipal enforcement setting, and identifies the gap that motivates this project.

2.2 Key Concepts and Theories

2.2.1 Road Reserves, Building Lines and Setbacks

A *road reserve* is the strip of land legally set aside for a road, including the carriageway, shoulders, drainage and space for services. A *building line* or *setback* is the minimum distance that a building must be kept from a reference line, commonly the centreline or the edge of the road reserve. In Zambia, public roads are classified and administered under the Public Roads Act of 2002 [5], and planning authorities such as LCC exercise development control under the Urban and Regional Planning Act of 2015 [6]. The distances configured in this project, 100 ft (30.48 m) for main roads, 60 ft (18.288 m) for district and branch roads and 30 ft (9.144 m) for other roads, reflect the imperial building-line distances used in Zambian road legislation. [Roads and Road Traffic Act, chapter 464 of the Zambian Law, Section 13 of 1995. Because the applicable distance depends on the road class and may be varied by specific decisions, determining it correctly is itself a non-trivial step in enforcement.

2.2.2 Geographic Information Systems and Spatial Analysis

A GIS is a system for capturing, storing, analysing and presenting data that are referenced to locations on the Earth [8]. In the vector data model, roads are represented as lines (sequences of coordinates) and buildings as polygons. Two spatial operations are central to setback analysis. A *buffer* creates a polygon containing all points within a given distance of a feature, which corresponds directly to a setback zone. A *distance* operation computes the shortest separation between two features. Because geographic coordinates are angles rather than lengths, distances must be computed geodesically or in a suitable projected coordinate reference system; libraries such as Turf.js compute distances on a sphere from longitude/latitude coordinates [7].

2.2.3 Volunteered Geographic Information and OpenStreetMap

Goodchild described the phenomenon of citizens acting as "sensors" who collectively produce geographic data, which he termed volunteered geographic information. OpenStreetMap is the most prominent example: a global, openly licensed map built by volunteers and distributed under the Open Database License [8]. Studies of OSM quality have found that its positional accuracy can approach that of official datasets in well-mapped areas, while completeness varies considerably between places. For Lusaka, OSM is often the most complete freely available source of road and building data, but its classification of roads (for example "primary", "secondary" or "residential") is a descriptive tagging scheme rather than a legal classification.

2.2.4 Spatial Data Formats, Databases and Web Mapping

GeoJSON is a standard JSON-based format for encoding geographic features, using WGS 84 longitude/latitude coordinates [9]. PostGIS extends the PostgreSQL relational database with spatial types, spatial indexes and functions such as distance and within-distance queries, allowing proximity searches to be answered efficiently inside the database [10]. On the client side, Leaflet is a lightweight open-source JavaScript library for interactive maps that renders vector layers such as GeoJSON over tiled base maps [10], [11]. Together these technologies make it possible to build GIS applications that run in an ordinary web browser.

2.3 Existing Systems and Related Work

2.3.1 Manual Field Inspection

The traditional approach is for building inspectors to visit sites, measure the distance from the road with a tape or measuring wheel, and record findings on paper forms. This produces authoritative measurements but is slow, expensive in staff time and transport, and hard to scale across a large city. Results are difficult to aggregate and audit, and the determination of the applicable setback relies on each officer's knowledge of the regulations.

2.3.2 Desktop GIS Analysis

Planning departments with GIS capacity can use desktop software such as QGIS or ArcGIS to buffer road centrelines and overlay building footprints, identifying buildings that fall inside the buffer [12]. This is fast and accurate for a trained analyst, but it requires specialist skills and software, is typically performed as a one-off analysis, and does not by itself encode the rules for choosing the correct distance, handle exceptions, or keep a per-building record of review decisions.

2.3.3 Web-Based and Enterprise GIS Platforms

Enterprise web GIS platforms allow organisations to publish maps and analysis tools to non-specialist staff through a browser. They provide authentication, data management and configurable dashboards, but they are generally commercial products with licensing costs, and setback-specific logic such as class-based rules, overrides and exception precedence must still be built on top of them.

2.3.4 Automatically Extracted Building Footprints

Recent work has used deep learning to extract building footprints from high-resolution satellite imagery at continental scale, for example Google's Open Buildings dataset covering Africa [13] and Microsoft's global building footprints. These datasets can fill gaps in volunteered data and are well suited as input to a screening system, but they do not themselves determine compliance with any regulation.

Table 1: Comparison of approaches to detecting road setback violations

Criterion	Manual inspection	Desktop GIS	Enterprise web GIS	This project
Cost to the institution	High staff/transport cost	Low-medium (QGIS free)	Licence cost	Open source; low
Specialist GIS skills needed	No	Yes	For set-up	No (for officers)
Automatic rule selection	No	No	Must be built	Yes (configurable)
Handles overrides/exceptions	Officer judgement	Manual	Must be built	Yes, with precedence
Screens a road in seconds	No	Yes	Yes	Yes
Per-building audit trail	Paper records	No	Possible	Yes
Role-based review workflow	No	No	Yes	Yes
Authoritative measurement	Yes	Depends on data	Depends on data	No – screening only

2.4 Comparative Analysis

Table 1 compares the approaches reviewed above. Manual inspection remains the only source of authoritative measurement, but it does not scale and is not consistent. Desktop GIS is fast and accurate but requires specialist skills and leaves rule selection to the analyst. Enterprise web GIS offers accessibility and access control but at a licence cost and without setback-

specific logic. None of the approaches combines automatic, explainable selection of the applicable setback with fast screening and an auditable review workflow that non-specialist officers can use.

2.5 Research and Knowledge Gap

The review shows a gap between the availability of open spatial data and tools on the one hand and the practical needs of a municipal enforcement unit on the other. Specifically, there is no low-cost tool that (i) encodes setback regulations, road-specific decisions and exceptions as maintainable data; (ii) applies them consistently and records which rule produced each result; (iii) screens buildings along a road quickly from open data; and (iv) keeps results under human control through verification flags, history and administrative review. This project addresses that gap.

2.6 Conceptual Framework

The project is guided by a three-layer "screening, not deciding" framework. The first layer is *rules as data*: regulatory knowledge is held in configuration and database records, and a resolution procedure determines the applicable rule for each building (standard rule → road-wide assignment → exception, with explicit precedence). The second layer is *measurement*: spatial operations compute the observed distance independently of the rules. The third layer is *human-in-the-loop verification*: the system compares the two and produces a screening status, but every result carries the rule's provenance and a verification flag, and final decisions are made and recorded by authorised officers. Separating these layers keeps each one testable and makes results explainable to officers and property owners. The detailed pipeline is shown in Chapter 4.

2.7 Chapter Summary

This chapter reviewed the regulatory, geospatial and technological foundations of setback enforcement and compared existing approaches. It concluded that open data and web GIS make automated screening feasible, but that a tool combining configurable rule resolution, fast screening and auditable human review is lacking. The next chapter describes the methodology used to build such a tool.

Chapter 3: Methodology

3.1 Introduction

This chapter explains how the project was carried out: the development approach, how requirements and data were obtained, the overall architecture, the tools and technologies used, the ethical and legal considerations, and the method used to evaluate the result.

3.2 Research Design

3.2.1 Research Paradigm

This project follows a design science research approach — it builds and evaluates a software artefact (the Road Setback Management System) to solve a real-world problem: the slow and inconsistent screening of building setback compliance in Lusaka. The study is applied and developmental, using a mixed-methods design:

- Quantitative: distance measurement, rule resolution, performance testing.
- Qualitative: stakeholder requirements, regulatory review, code review, user feedback.

3.2.2 Influence of Prior Studies

The research design was shaped by two prior studies on setback compliance in African cities.

Akingbade et al. (2022) — Ile-Ife, Nigeria. Used GIS and remote sensing to assess building compliance with road setbacks. Found 6,284 buildings in contravention, mostly in the city centre. This showed that GIS can reliably detect setback violations at scale, justifying spatial analysis as the core screening method in this project. However, it was a one-off assessment, it did not produce a reusable tool.

Omollo (2020) — Kisii, Kenya. Studied compliance with building-line standards and the factors behind non-compliance. Found that non-compliance was largely caused by weak enforcement capacity, not ignorance of the rules. This motivated building a tool that makes enforcement faster, consistent and auditable.

Other supporting work (Qonono, 2025; Arimah & Adeagbo) confirmed that rule-based spatial screening is useful in African regulatory contexts.

3.3 Development Approach

An iterative and incremental approach inspired by Agile principles [14], [15] was adopted. Agile was preferred over a sequential Waterfall model because several requirements, notably how rules, overrides and exceptions interact and how candidate buildings should be retrieved efficiently, could only be settled by building and testing working increments. Each iteration delivered a usable slice of functionality that was reviewed before the next was planned. The evolution of the code base reflects this: an early version loaded all roads and buildings into the analysis function, and this was later replaced by a "road-first" design in which the database retrieves only buildings near the selected road (a legacy compatibility function was temporarily retained during the migration). Similarly, the database schema evolved through three ordered migrations, each correcting or extending the previous one.

Table 2: Development iterations and deliverables

Iteration	Focus	Main deliverables
1	Rule model	Setback configuration file; road-class resolution; compliance evaluation
2	Measurement	Road-to-feature distance for points, lines and polygons; setback zone buffer
3	Overrides and exceptions	Road-wide assignments; structure/parcel/road-segment exceptions; precedence rules
4	Road-first backend	PostGIS candidate search function; Express API (/api/analyze, /api/roads, /api/health)
5	Web client	Dashboard with Leaflet map, road selection, setback zone, results table
6	Accounts and records	Supabase authentication; inspections and building_candidates tables; history
7	Administration	Admin roles, profiles table, candidate status review, role management
8	Evaluation	Automated test suite, code review, documentation

3.3.1 Scrum as the Selected Agile Framework

Within the family of Agile methods, Scrum was selected as the specific framework for managing the work [16]. Scrum was chosen because:

It provides a lightweight, well-defined structure (roles, events and artifacts) that a small student team can adopt quickly.

It supports incremental delivery, which suited the modular nature of the system (rule engine, geometry, API, web client, database).

It encourages regular inspection and adaptation, which helped the team respond to defects found during testing (for example, the exception-precedence bug and the search-radius issue). It is the most widely used Agile framework in industry, so applying it gave the team relevant professional experience.

Because the team was new to Scrum and the project was run alongside other academic commitments, the framework was applied in a lightweight, "Scrum-informed" manner rather than in full enterprise form. The team explicitly acknowledges that this was a learning process: roles were rotated, sprint lengths were sometimes adjusted, and some ceremonies were simplified. This is documented transparently below.

3.4 Requirements and Data Collection

Requirements were derived from two sources:

- **Regulatory sources.** The setback distances for each road class and the concept of road-specific decisions and exceptions were derived from Zambian roads and planning legislation [2X], [3] and encoded in the configuration file `config/setback-rules.json`.
- **Spatial data.** Road centrelines and building footprints for the study area were loaded into a Supabase PostgreSQL/PostGIS database as the `roads` and `buildings` tables. The road attributes (`fid`, `name`, `fc`, `class`) follow the schema of the widely used Geofabrik OpenStreetMap extracts.

3.5 System Architecture

The system follows a three-tier architecture (Figure 1). The presentation tier is a Next.js web application that runs in the officer's browser. The application tier is an Express/Node.js GIS service that resolves rules, measures distances and classifies buildings. The data tier is a Supabase project providing a PostgreSQL database with the PostGIS extension, authentication and row-level security. The web client calls the GIS service through a server-side API route, and talks directly to Supabase for authentication and for storing inspection records under row-level security.

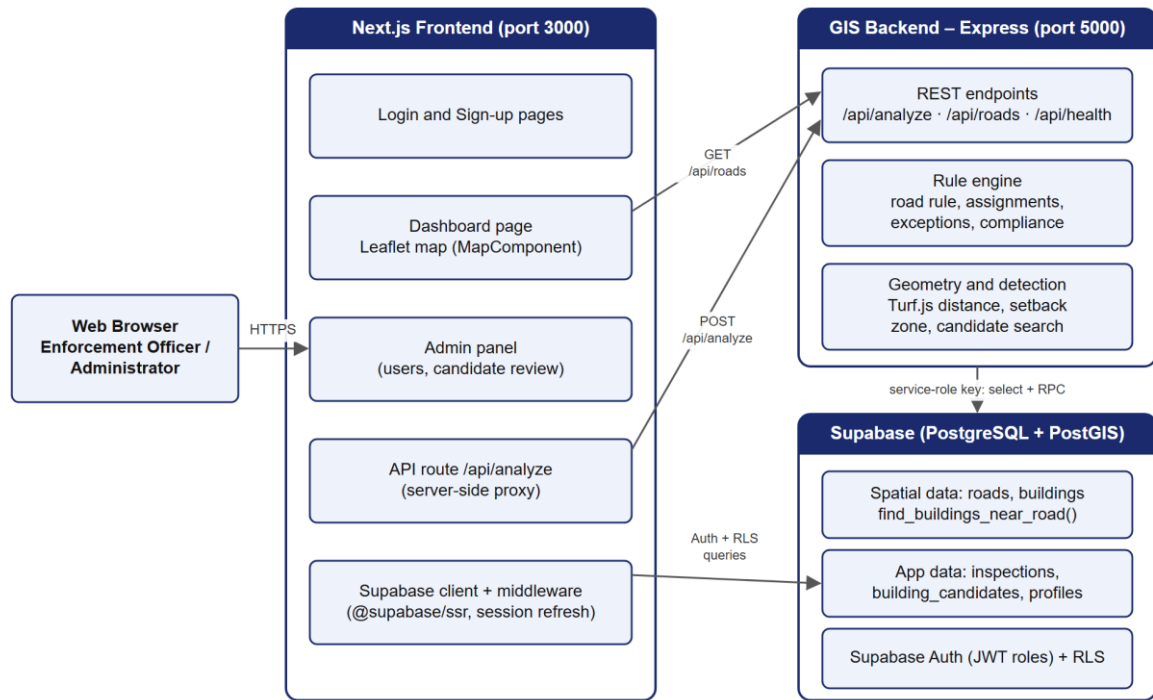


Figure 1: System architecture of the Road Setback Management System

3.6 Tools and Technologies

Table 3: Tools and technologies used

Category	Technology	Purpose
Language	JavaScript (ES modules), TypeScript	Backend logic (JavaScript); web client (TypeScript/TSX)
Runtime	Node.js (tested on v24.14.0), npm	Running the GIS service and build tools
Backend framework	Express 4, cors, dotenv	REST API, cross-origin access, environment configuration
Geospatial library	Turf.js 6/7	Distances, nearest points, buffers, intersection tests
Frontend framework	Next.js 16 (App Router), React 18	Pages, routing, server-side API proxy
Web mapping	Leaflet 1.9, OpenStreetMap tiles	Interactive map, road and building layers
Database	Supabase (PostgreSQL + PostGIS)	Spatial data, application data, spatial queries
Authentication	Supabase Auth, @supabase/ssr	Sign-up/log-in, sessions, JWT role claims
Security	PostgreSQL row-level security, GRANTS, SECURITY DEFINER functions	Data access control and role management
Testing	node:test, node:assert	Automated unit and integration tests
Version control	Git, GitHub	Source control and collaboration
Editor	Visual Studio Code	Development environment

3.7 Ethical and Legal Considerations

- **Personal data.** The only personal data processed are staff account details (e-mail address, role, sign-in times). These are stored in Supabase, protected by row-level security, and visible in full only to administrators, in line with the principles of the Data Protection Act No. 3 of 2021 [16]. Building records are not linked to owners' personal information.
- **Responsible use of results.** Screening results can affect property owners. The system therefore never presents a result as a legal determination: every candidate is flagged as requiring verification, provisional rule mappings are explicitly marked, and incomplete data yields an "unknown" status rather than "compliant".
- **Licensing.** OpenStreetMap data are used under the Open Database License, which requires attribution; the map displays the OSM attribution [17]. All software libraries used are open source.
- **Security.** Access requires authentication, administrative functions are restricted to users with an administrator role, and database credentials are kept in environment files excluded from version control. Security issues identified during review are reported in Chapter 6.
- **Institutional branding.** The prototype refers to Lusaka City Council as its intended user.

3.8 Evaluation Method

The system was evaluated against the objectives in Section 1.4 using the following methods:

- **Functional correctness** was tested with 31 automated test cases written with Node.js's built-in test runner. Geometric tests used synthetic roads and buildings placed at known distances near Lusaka (latitude -15.4°), so that the expected distance is known in advance; a tolerance of ± 0.5 m was allowed for geodesic approximation.
- **Performance** was measured by timing 1,000 consecutive candidate assessments in the GIS engine.
- **Code review** of the full code base was used to identify defects, security issues and usability problems not covered by the automated tests.
- **User evaluation** with the System Usability Scale (SUS) [18] was planned with enforcement officers (Appendix E).

3.9 Chapter Summary

The project followed an iterative, Agile-inspired approach, drew its requirements from legislation, stakeholder input and the available spatial data, and was built as a three-tier web application on open-source technologies. Evaluation combined automated testing, performance measurement and code review.

Chapter 4: System Analysis and Design

4.1 Introduction

This chapter analyses the users and requirements of the system and presents its design: use cases, architecture, database, user interface and the key algorithms for rule resolution and distance measurement.

4.2 Stakeholder and User Analysis

Table 4: Stakeholders and their interests

Stakeholder	Role	Interest in the system
Enforcement officer / building inspector	Primary user	Quickly find buildings likely to violate the setback on a road; keep a record of inspections
Administrator (senior planning officer)	Primary user	Review and override candidate status; manage which staff have administrative rights
Lusaka City Council management	Potential Stakeholder	Consistent, auditable and efficient enforcement; better use of inspection resources
Road authorities	Indirect beneficiary	Protected road reserves for future widening and services
Property owners and developers	Affected party	Fair, consistent and explainable decisions; early warning before building
System maintainers	Technical	Maintainable rule configuration, clear architecture, secure deployment

4.3 Functional Requirements

Table 5: Functional requirements

ID	Requirement	Priority
FR-01	The system shall allow a user to sign up and log in with an e-mail address and password.	High
FR-02	The system shall restrict the dashboard and administrator panel to authenticated users.	High
FR-03	The system shall display the road network on an interactive map.	High
FR-04	The system shall allow the user to select a road by clicking it on the map or by entering its ID.	High
FR-05	The system shall determine the applicable setback for the selected road from its classification, any road-wide assignment and any applicable exceptions.	High
FR-06	The system shall retrieve buildings within the applicable distance of the selected road.	High

ID	Requirement	Priority
FR-07	The system shall measure the shortest distance between each candidate building and the road.	High
FR-08	The system shall classify each candidate as compliant, non-compliant or unknown, and report its deviation from the required distance.	High
FR-09	The system shall display the setback zone and colour-coded candidates on the map with details in a pop-up and a results table.	High
FR-10	The system shall record each inspection run and its candidates, and show the user's recent inspection history.	Medium
FR-11	The system shall allow administrators to view and change the status of recorded candidates.	Medium
FR-12	The system shall allow administrators to view all users and change a user's role between user and admin.	Medium
FR-13	The system shall mark results based on provisional rule mappings as requiring verification.	High
FR-14	The system shall expose a health-check endpoint for the GIS service.	Low

4.4 Non-Functional Requirements

Table 6: Non-functional requirements

ID	Category	Requirement
NFR-01	Performance	Assessment of the candidates along one road should complete within a few seconds; the engine itself should process each candidate in well under 10 ms.
NFR-02	Usability	Officers without GIS training should be able to run a screening in no more than three actions (log in, select road, run).
NFR-03	Security	Only authenticated users may access data; only administrators may change candidate status or user roles; secrets must not be stored in source control.
NFR-04	Reliability	Missing or invalid data must never produce a "compliant" result; errors for one candidate must not abort the whole road assessment.
NFR-05	Explainability	Every result must include the rule applied, its source and the measured and required distances.
NFR-06	Maintainability	Setback distances and road-class mappings must be changeable through configuration without code changes; modules should have a single responsibility.
NFR-07	Portability	The client must run in a standard modern web browser without plug-ins.
NFR-08	Cost	The system should rely on open-source software and open data.

4.5 Use Cases

Figure 2 shows the use cases of the two actors. The Enforcement Officer signs in, selects a road, runs the screening, views results and history. The Administrator can additionally view all

users, change user roles and review or override the status of recorded candidates. Table 7 describes the central use case in detail.

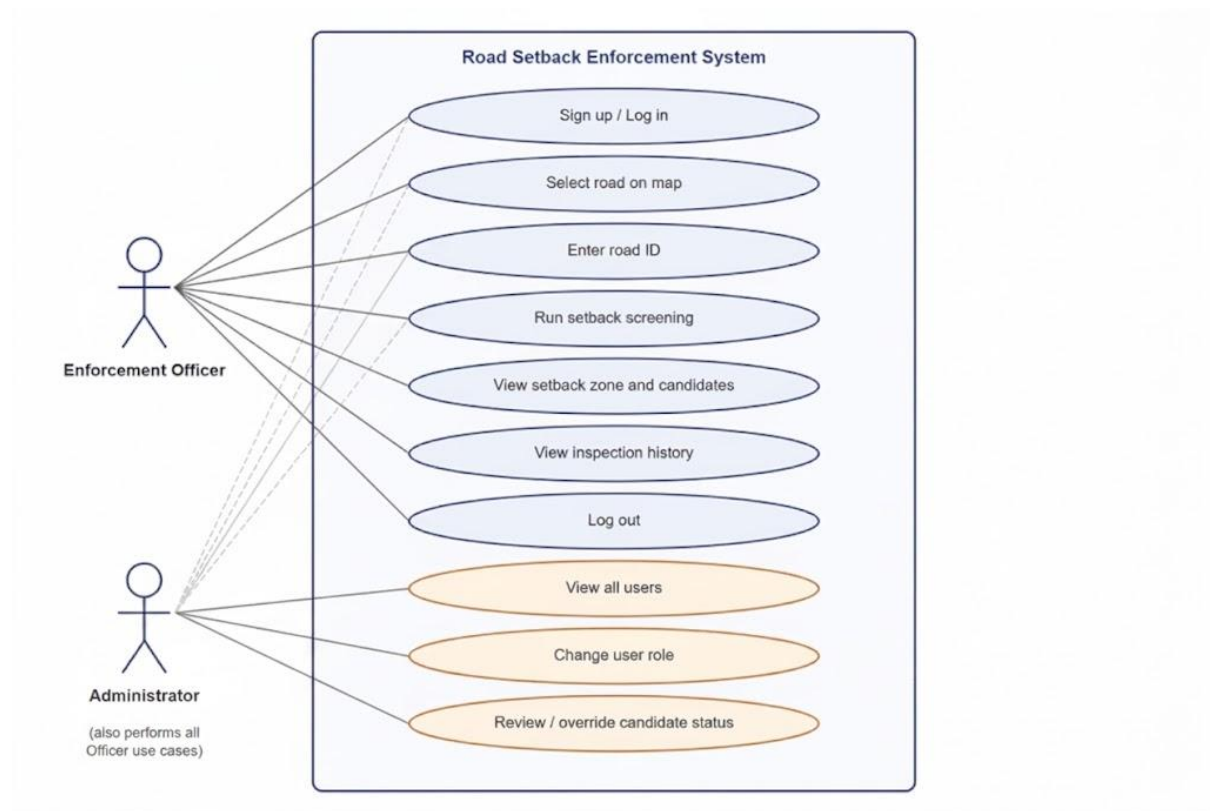


Figure 2: Use case diagram

Table 7: Use case description: Run setback screening

Item	Description
Use case	UC-04 Run setback screening
Actor	Enforcement Officer (or Administrator)
Pre-conditions	The user is logged in; road and building data are loaded; the GIS service is running.
Main flow	1. The user clicks a road on the map or enters a road ID and presses Run. 2. The system loads the road and resolves the applicable setback. 3. The system retrieves buildings within that distance. 4. The system measures each building's distance and classifies it. 5. The system displays the road, setback zone and colour-coded candidates, and updates the counters and results table. 6. The system saves the inspection and its candidates and refreshes the history.
Alternative flows	A1: invalid road ID – the user is asked to enter a valid ID. A2: side-specific rule – the road is reported with status "unknown" and a reason. A3: a candidate cannot be measured – that candidate is reported as "unknown" and the others are still assessed. A4: the GIS service is unreachable – an error message is shown.
Post-conditions	An inspection record and one building_candidates row per candidate exist in the database.

4.6 System Architecture

The final architecture was introduced in Figure 1. Its components interact as follows. The **Next.js frontend** provides the login, sign-up, dashboard and admin pages. Its middleware refreshes the Supabase session on each request, and a server-side API route (`/api/analyze`) forwards analysis requests to the GIS service so that the browser does not need to know the service's address. The **GIS backend** exposes `POST /api/analyze` (screen one road), `GET /api/roads` (road layer for the map) and `GET /api/health`. Internally it is divided into a rule engine (`src/rules`), a geometry module (`src/geometry`) and a detection module (`src/detection`) coordinated by an orchestrator. **Supabase** stores the spatial layers and application tables, runs the PostGIS candidate query, and enforces authentication and row-level security. Figure 3 shows the interactions during a screening run.

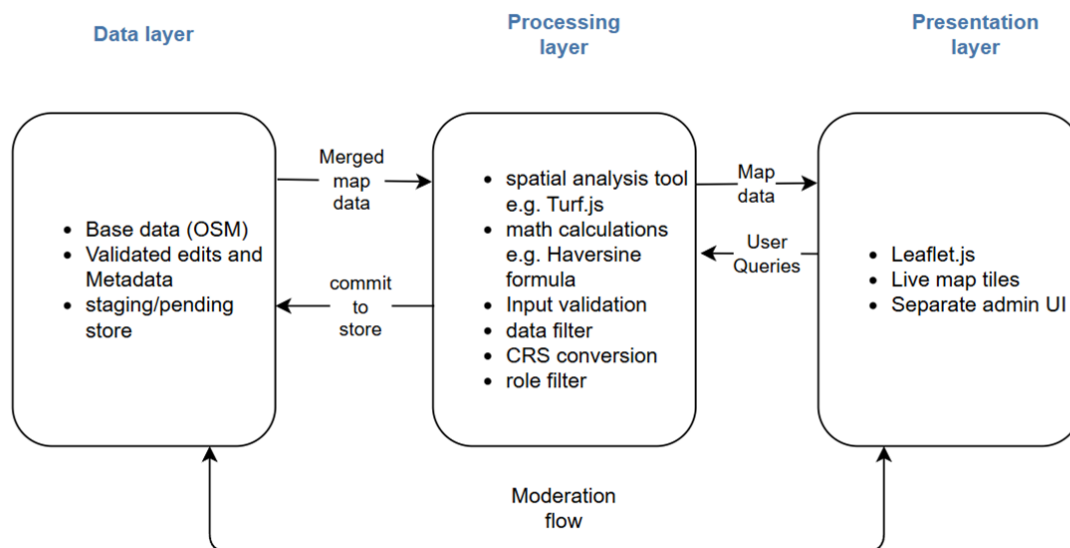


Figure 3: Sequence diagram for screening a road

4.7 Database Design

Figure 4 shows the data model and Table 8 summarises each table. The spatial layers (roads, buildings) are kept separate from the application tables, and the application never writes to them. Each inspection run creates one `inspections` row and one `building_candidates` row per screened building, preserving a full history even if a building is screened several times. User roles are held in Supabase's protected `app_metadata`, which users cannot edit themselves,

and mirrored into a profiles table by database triggers so that administrators can list users with an ordinary, RLS-protected query.

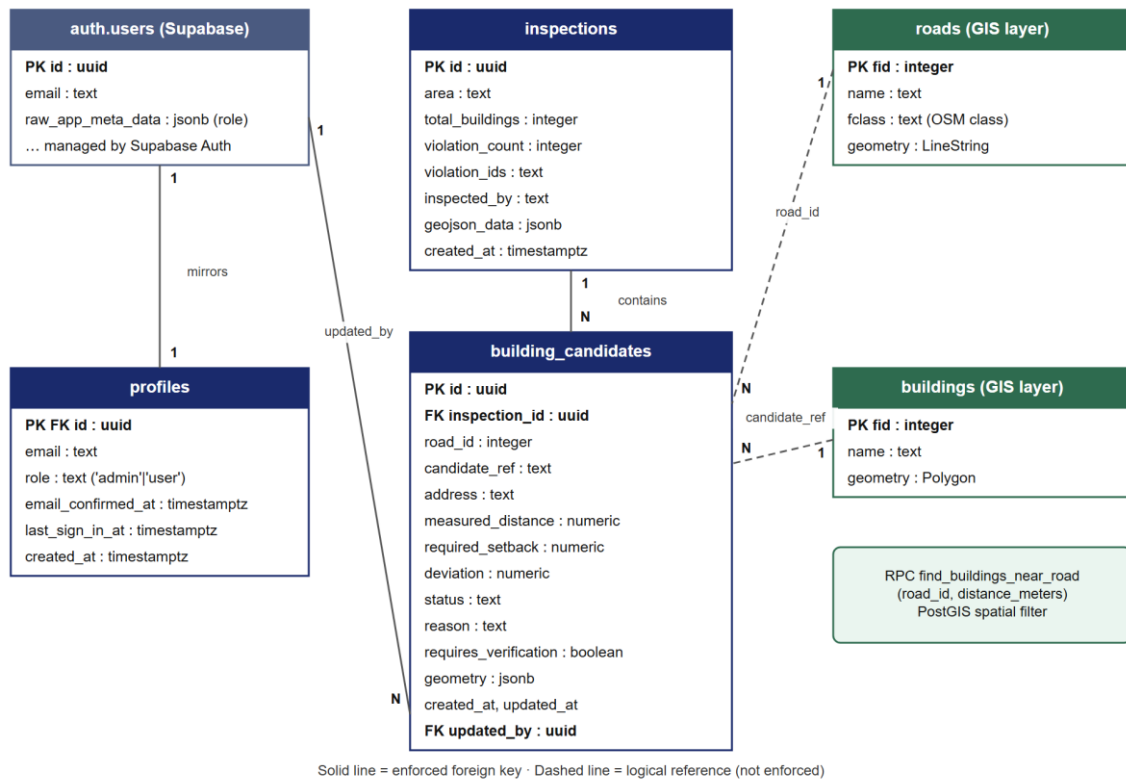


Figure 4: Entity-relationship diagram

Table 8: Database tables

Table	Purpose	Access control
roads	Road centrelines with OSM class (fclass) and name	Read by the GIS service with the service-role key
buildings	Building footprints	Read through the find_buildings_near_road() function
inspections	One row per screening run: area, counts, violation IDs, full result JSON, inspector	Authenticated users may select and insert
building_candidates	One row per screened building: distances, status, reason, verification flag, geometry	Authenticated users may select and insert; only admins may update
profiles	Mirror of user accounts and roles for the admin panel	Admins see all rows; users see their own row
auth.users	Supabase-managed accounts; role stored in app_metadata	Managed by Supabase; changed only via set_user_role()

4.8 Interface Design

The interface was designed for officers without GIS training and follows a simple map-plus-panel layout. The **login and sign-up pages** (Figure 5) use a single centred form with the council's name and the system title. The **dashboard** (Figure 6) has a navigation bar showing the user's e-mail, an Admin Panel link for administrators and a Logout button. About three-quarters of the width is taken by the map, which shows the whole road network in grey. The selected road is drawn in dark blue with its setback zone as a translucent orange band, and candidate buildings are coloured red (potential violation), green (compliant) or amber (needs review). The side panel contains the road-ID input and Run button, summary counters, a results table and the inspection history. The **administrator panel** (Figure 7) lists users with a role selector and recorded candidates with a status selector.

 **LCC** Report a Violation [← Back to Dashboard](#)

Use this form to log a setback violation directly, for example while out in the field, without needing to run a full automated analysis first.

Report category *

Select a category

Report type *

Select a report type

Requested action *

Select the requested action

Description *

Describe what you observed (structure, how close to the road, any other detail)...

Road ID (optional)

e.g. 52

Road name (optional)

e.g. Great East Road

Candidate ID (optional)

Paste the candidate UUID

If this report concerns a specific building candidate, enter its ID. Leave blank for general road or site reports.

 Use my current location

Photo (optional)

Choose file

No file chosen

Submit report

Figure 5: report page

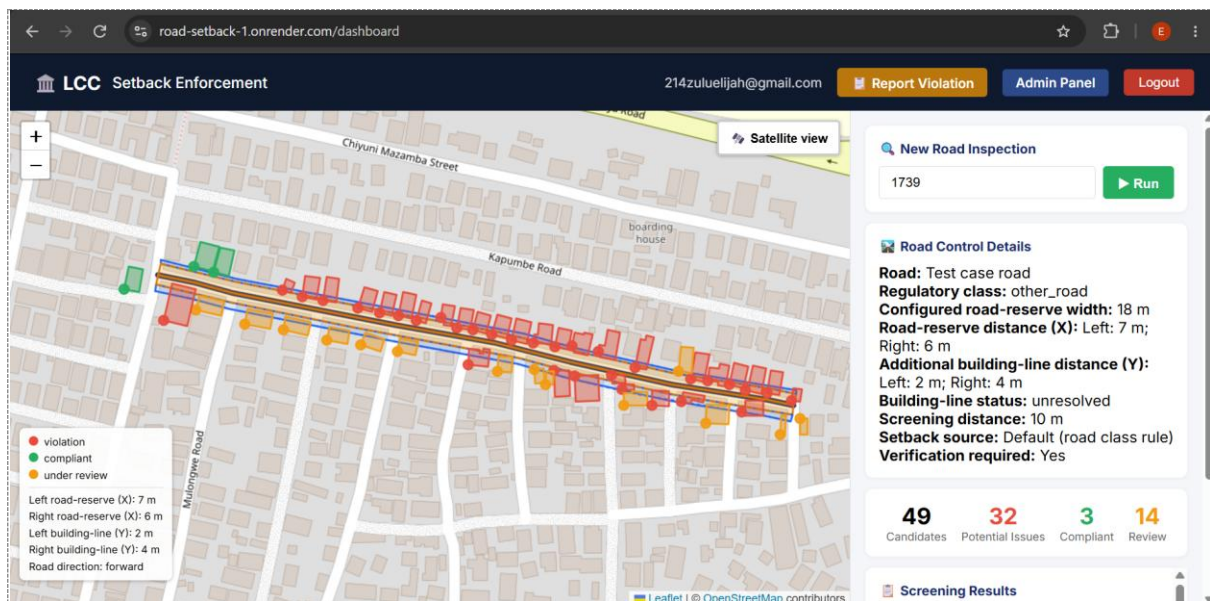


Figure 6: Dashboard showing a screened road, its setback zone and candidate buildings

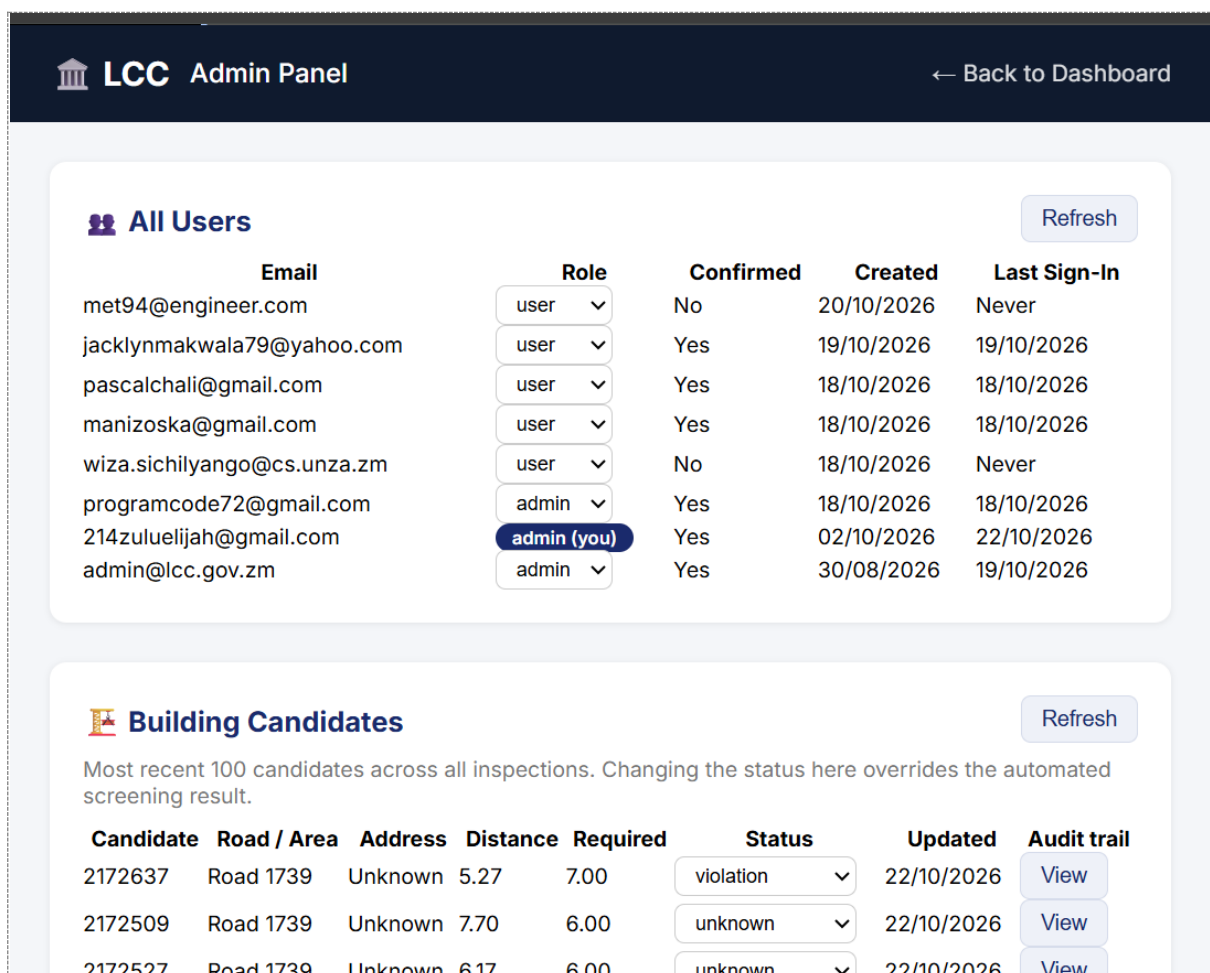


Figure 7: Administrator panel with user and candidate management

4.9 Algorithms and Technical Design

4.9.1 Rule Resolution

The applicable setback for a building is resolved by the pipeline in Figure 8. First, the road's OSM class (the highway tag, or fclass when that is absent) is mapped to a regulatory class and default distance using the configuration in Table 9. Unmapped classes fall back to the "other road" class and are marked provisional, which propagates a verification requirement to the result. Second, an approved road-wide assignment for that road, if any, replaces the default; where several exist, the one with the highest priority wins. Third, exceptions that match the building (by structure ID, parcel ID or road-segment ID, optionally restricted to certain feature types) are collected. Fourth, if several exceptions apply, they are ordered by explicit priority, then by scope specificity (road segment > structure > parcel). If two exceptions remain tied, the engine raises an error instead of silently choosing one. Finally, the selected exception is applied to give the final rule, which keeps a record of its source, reference, reason and the identifiers of the assignment and exception used.

Table 9: Configured road classes and default road reserve distances

OSM class	Regulatory class	Default setback	Status
primary	main_road	30.480 m (100 ft)	Configured
secondary	district_road	18.288 m (60 ft)	Configured
tertiary	branch_road	18.288 m (60 ft)	Configured
unclassified	other_road	9.144 m (30 ft)	Provisional – external data required
any other class	other_road	9.144 m (30 ft)	Provisional – external data required

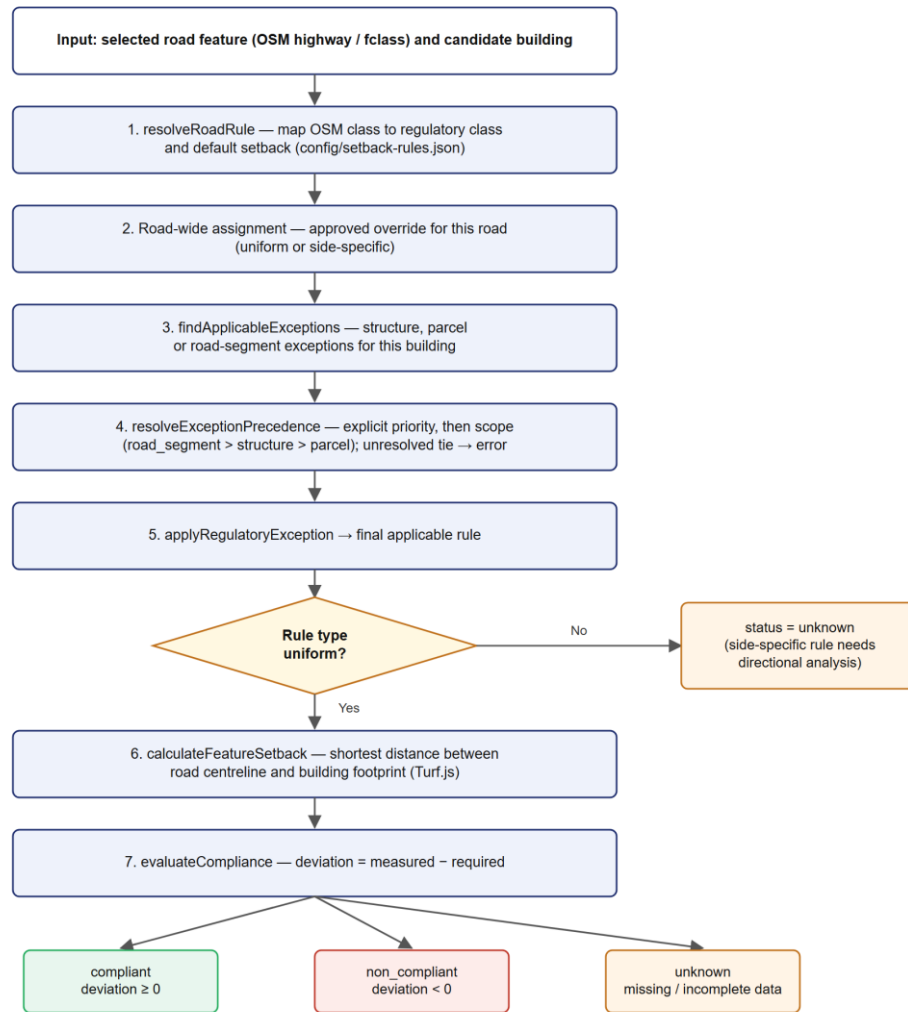


Figure 8: Rule resolution and compliance evaluation pipeline

4.9.2 Candidate Retrieval

Rather than transferring every building to the application, the GIS service calls the database function `find_buildings_near_road(road_id_input, distance_meters)`, which uses PostGIS to return only buildings within the applicable distance of the selected road. This "road-first" design keeps network traffic and processing proportional to the number of nearby buildings rather than the size of the city.

4.9.3 Distance Measurement

For each candidate, the engine computes the shortest distance between the road centreline and the building. If the two geometries intersect, the distance is zero. For a point feature, it is the point-to-line distance. For lines and polygons, the minimum is taken over (a) the distance from every vertex of the building to every road line and (b) the distance from every road vertex to every edge of the building's boundary. Checking both directions is necessary because the closest approach may occur at a road vertex facing the middle of a building edge, or at a

building corner facing the middle of a road segment. MultiLineString roads and MultiPolygon buildings are split into their parts. The algorithm is $O(n \cdot m)$ in the number of building and road vertices, which is small for individual buildings. Distances are computed geodesically in metres by Turf.js, and the nearest point on the road is returned for display.

4.9.4 Compliance Evaluation

Given the required distance R and the measured distance D , the deviation is $D - R$. A building is **compliant** when the deviation is zero or positive and **non-compliant** when it is negative. If either value is missing or the source data are marked incomplete, the status is **unknown** with a reason, so that missing data can never be mistaken for compliance. Roads whose final rule is side-specific are reported as unknown because they require directional analysis.

4.10 Chapter Summary

This chapter identified the stakeholders and requirements of the system and presented its design: a three-tier architecture, a data model that separates spatial layers from auditable inspection records, a simple map-centred interface, and algorithms for rule resolution, candidate retrieval, distance measurement and compliance evaluation.

Chapter 5: Implementation

5.1 Introduction

This chapter describes how the design was implemented: the development environment, the main components, key implementation details, how the parts were integrated, and the main challenges encountered.

5.2 Development Environment

- **Hardware:** HP laptop .
- **Operating system:** Windows 11
-
- **Programming languages:** JavaScript (ES modules), TypeScript
- **Frameworks:** Express 4, Next.js 16,
- **libraries** (React 18, Leaflet 1.9, Turf.js)
- **Database:** Supabase: PostgreSQL + PostGIS (HP laptop)
- **Local ports:** the web client runs on port 3000 and the GIS service on port 5000.
- **IDE / tools:** Visual Studio Code, Node.js v24.14.0 and npm, Supabase web console
- **Version control:** Git and GitHub (**Lubasi-unza/Road-Setback**)
- **Deployment:** Live on **Render:** <https://road-setback-1.onrender.com/> • **Supabase (hosted PostgreSQL + PostGIS & Auth**

5.3 System Components and Modules

Table 10: Main components and modules

Component	Location	Responsibility
GIS server	backend/server.js	Express app; /api/analyze, /api/roads, /api/health; loads the road from Supabase
Orchestrator	backend/src/orchestrator.js	assessRoad(): resolves the road rule, finds candidates, assesses each, builds the road-level result
Rule configuration	backend/config/setback-rules.json	Regulatory classes, distances and OSM-class mapping

Rule engine	Backend/src/rules/*.js	resolveRoadRule, find/applyRoadRuleAssignment, findApplicableExceptions, resolveExceptionPrecedence, applyRegulatoryException, resolveApplicableSetback, evaluateCompliance, normalizeFeature
Geometry	Backend/src/geometry/calculateFeatureSetback.js	Shortest road-to-feature distance and nearest point
Detection	Backend/src/detection/*.js	findCandidateFeatures (PostGIS RPC), assessCandidate, createSetbackZone
Login / sign-up pages	frontend/app/page.tsx, signup/page.tsx	Supabase e-mail/password authentication
Dashboard	frontend/app/dashboard/page.tsx	Road selection, running analysis, counters, results table, history, saving records
Map component	frontend/components/MapComponent.tsx	Leaflet map, road layer, selected road, setback zone, candidate styling and pop-ups
Admin panel	frontend/app/admin/page.tsx	User list and role changes; candidate list and status changes
API proxy	frontend/app/api/analyze/route.ts	Validates road_id and forwards requests to the GIS service
Session handling	frontend/middleware.ts, utils/supabase/*	Browser, server and middleware Supabase clients; session refresh
Database migrations	supabase/migrations/*.sql	Tables, indexes, triggers, RLS policies, grants and set_user_role()

5.4 Key Implementation Details

5.4.1 Rules as Configuration

Setback distances and road-class mappings are stored in a JSON file rather than in code, so that they can be corrected or extended without changing any logic. An excerpt is shown below; the full file is reproduced in Appendix F.

```
"road_setback_rules": {
  "main_road": { "setback_distance_m": 30.480 },
  "district_road": { "setback_distance_m": 18.288 },
  "branch_road": { "setback_distance_m": 18.288 },
  "other_road": { "setback_distance_m": 9.144 }
},
"osm_classification": {
  "primary": { "regulatory_class": "main_road", "classification_status": "configured" },
  "default": { "regulatory_class": "other_road", "classification_status": "provisional",
```

```
"external_data_required": true }  
}
```

5.4.2 Road Assessment

The orchestrator's `assessRoad()` function implements the screening workflow. It resolves the road-level rule, and if the rule is side-specific it returns immediately with an "unknown" status. Otherwise it retrieves candidates within the required distance and assesses each one inside its own try/catch block, so that a single malformed building yields an "unknown" candidate instead of failing the whole road. The road-level result is "provisional" if the rule mapping is provisional, and `requiresVerification` is true if the road or any candidate needs verification.

```
const candidates = candidateFeatures.map((candidateFeature) => {  
  try {  
    return assessCandidate({ roadFeature, candidateFeature,  
                             roadRuleAssignments, exceptionFeatures });  
  } catch (error) {  
    return { candidateId: candidateFeature.id ?? null, measuredDistanceMeters: null,  
            requiredSetbackMeters, rule: roadRule, screeningStatus: 'unknown',  
            reason: error.message, requiresVerification: true };  
  }  
});
```

5.4.3 Distance Measurement

`calculateFeatureSetback()` first converts the road into one or more `LineStrings` and returns zero if the road intersects the feature. It then evaluates vertex-to-line distances in both directions, as described in Section 4.9.3, using `Turf.js`'s `pointToLineDistance` and `nearestPointOnLine` with metres as the unit.

5.4.4 Persisting Inspections

After a successful analysis, the dashboard converts the backend response into GeoJSON features for display (mapping the engine's "non_compliant" status to the user-facing label "violation"), inserts an inspections row containing the counts and the full result JSON, and then inserts one `building_candidates` row per candidate linked to that inspection. Both inserts run under the logged-in user's session and are therefore subject to row-level security.

5.4.5 Role-Based Access Control

Administrator status is stored in the user's `app_metadata`, which is included in the signed JWT and cannot be modified by the user. Row-level security policies check this claim; for example, only administrators may update candidate status:

```
create policy "admins can update candidate status"
on public.building_candidates for update to authenticated
using      ( (auth.jwt() -> 'app_metadata' ->> 'role') = 'admin' )
with check ( (auth.jwt() -> 'app_metadata' ->> 'role') = 'admin' );
```

Changing a role requires writing to Supabase's protected `auth.users` table, which the client cannot do. This is handled by a `SECURITY DEFINER` function, `set_user_role(target_user_id, new_role)`, which first verifies that the caller is an administrator and that the new role is either "admin" or "user", and then updates both `auth.users` and `profiles`.

5.5 Integration

- **Client to GIS service.** The dashboard posts to the Next.js route `/api/analyze`, which validates the road ID and forwards it to `NEXT_PUBLIC_GIS_API_URL` (default `http://localhost:5000`), returning clear error messages if the service is unreachable. The map loads the road layer from the GIS service's `/api/roads` endpoint.
- **GIS service to database.** The service uses the Supabase JavaScript client with the service-role key (read from environment variables) to load roads and to call the PostGIS candidate function.
- **Client to database.** The browser uses the Supabase client with the public (publishable) key; all data access is therefore governed by the user's session and row-level security. Middleware refreshes the session cookie on every request.
- **Map rendering.** Because Leaflet requires the browser's window object, the map component is loaded with Next.js dynamic import and server-side rendering disabled.

5.6 Challenges and Solutions

Table 11: Implementation challenges and how they were addressed

Challenge	Solution
-----------	----------

Loading all roads and buildings into the analysis did not scale and slowed each request.	Redesigned the workflow to be road-first: the database returns only buildings near the selected road through a PostGIS function.
OSM road classes do not correspond one-to-one with legal road classes.	Introduced an explicit mapping with a "provisional" status for unmapped classes, which propagates a verification flag to results.
Several exceptions could apply to the same building, making the result ambiguous.	Defined a deterministic precedence (explicit priority, then scope specificity) and raised an error for unresolved ties instead of guessing.
Leaflet failed during server-side rendering in Next.js.	Loaded the map component on the client only via dynamic import with <code>ssr: false</code> .
Inserts and reads failed with "permission denied" even though RLS policies existed.	Added the missing table GRANTS for the authenticated role in a follow-up migration.
Listing users required the Admin API and the service-role key in the web server.	Created a profiles table kept in sync with <code>auth.users</code> by triggers and protected by RLS.
Role changes did not take effect immediately.	Documented that roles are embedded in the JWT; the user must sign in again or wait for the token to refresh.
Supabase credentials were accidentally committed to the repository.	Removed the file, added <code>.env</code> patterns to <code>.gitignore</code> .

5.7 Individual Contribution

The Road Setback Management System was developed collaboratively by three group members, with responsibilities distributed across frontend development, backend development and geospatial screening, and database management and deployment. Kupalwa Chibuye was primarily responsible for frontend development and user-interface design, Lubasi Sipangule for backend development and the geospatial screening engine, and Elijah Zulu for database management, data preparation, authentication and deployment. This division of responsibilities supported parallel development while ensuring that the system's major components could be developed and tested independently before integration.

The system follows a three-tier architecture consisting of a Next.js frontend, a Node.js and Express.js backend, and a PostgreSQL/PostGIS database hosted through Supabase. The frontend provides the interface for interacting with geographic data and screening results, the backend performs regulatory rule resolution and spatial assessment, and the database manages the geographic datasets and records required by the system. Although each member had a primary area of responsibility, the overall implementation required collaboration in defining

the interfaces between these components, resolving integration issues and ensuring that the application operated as a coherent system.

The contribution of each member is presented in turn: Elijah Zulu (Section 5.7.1), Lubasi Sipangule (Section 5.7.2) and Kupalwa Chibuye (Section 5.7.3), followed by a description of how the components were integrated (Section 5.7.4). Each member's individual statement is given in Appendix A (Sections A.3 to A.5).

5.7.1 Elijah Zulu: Database, Authentication, Access Control and Deployment

This subsection identifies the part of the work for which I, **Elijah Zulu**, was primarily responsible and explains how it fits with the work of my team members, **Lubasi Sipangule** and **Kupalwa Chibuye**. A fuller account is given in Section A.3 of Appendix A.

My responsibilities.

- **Database and data layer.** I set up and managed the Supabase project (PostgreSQL with PostGIS) that holds the roads and buildings of Lusaka and the application tables: inspections, building_candidates, profiles, road_rule_assignments, candidate_status_history and manual_reports. This included the keys, indexes, check constraints and triggers, and the four versioned migrations (Table 8 and Table 21).
- **Road and building data.** I obtained the road and building layers for Lusaka, loaded them into the database, and set up the database side of retrieving the buildings that lie near a selected road, which the GIS service calls when it screens a road (Section 4.9.2).
- **User authentication and validation.** I implemented sign-up and sign-in with Supabase Auth, including input checks (required e-mail, a minimum password length of eight characters, password confirmation and e-mail confirmation), session handling with @supabase/ssr and a middleware that refreshes the session on every request, and redirection of unauthenticated visitors to the login page.
- **Access control for administrators and normal users.** I stored the administrator role in the protected app_metadata of each account, wrote the row-level security policies and table grants that depend on it, created the profiles table kept in step with auth.users by triggers, and wrote the set_user_role() function that lets only administrators change roles (Section 5.4.5). Normal users see the officer functions only; the administrator panel is restricted to administrators.
- **Deployment.** I deployed the application on Render, connected it to the hosted Supabase project, and configured the environment variables so that the service-role key stays on the server and only the publishable key reaches the browser (Table 19).

Work of other members. **Lubasi Sipangule** was responsible for the GIS service and its logic: the setback rules and configuration, road-class resolution, overrides and precedence, the distance calculations and the assessment of candidate buildings (Sections 4.9 and 5.4.1 to 5.4.3). **Kupalwa Chibuye** was responsible for the front-end design and user interface / user experience and carried out the usability evaluation with the System Usability Scale (Section 6.5). I describe their components only where needed to explain how my work connects to them, and I do not claim them as my own.

Where the work meets. My components connect to theirs at three points. The GIS service reads roads and calls the spatial function in my database using the service-role key. The dashboard saves each inspection and its candidate buildings into my tables under the signed-in user's session. The interface relies on the role claim and the policies I defined to decide what each user may see and change.

5.7.2 Lubasi Sipangule: Backend Development and Geospatial Screening Engine

The backend component provides the computational foundation of the system, coordinating regulatory rule processing, spatial candidate retrieval, geometric calculations and preliminary compliance assessment. This component was developed primarily by Lubasi Sipangule using Node.js, Express.js and Turf.js, with Supabase providing access to the PostgreSQL/PostGIS database.

The backend was organised into modules that separate regulatory classification, rule resolution, geometry processing, candidate assessment and workflow coordination. This modular approach improves maintainability, supports independent testing and allows changes to regulatory parameters without extensive modifications to the geometric processing functions.

a) Backend Architecture and API Development

The backend modules are organised under the backend/src/ directory. The rules/ directory contains functions for regulatory classification and rule resolution, while geometry/ handles spatial calculations. The detection/ directory implements candidate identification and assessment, and orchestrator.js coordinates the screening workflow.

An Express.js service exposes the screening functionality through REST API endpoints. The principal endpoint, POST /api/analyze, accepts a selected road identifier and initiates the assessment process. Supporting endpoints, including /api/roads and /api/health, provide road-related data access and service-health information.

A typical screening request involves validating the road identifier, retrieving the relevant road information and regulatory assignments, resolving the effective setback controls, and obtaining nearby candidate buildings from the spatial database. Each candidate is then assessed against the applicable regulatory thresholds, and the results are returned to the frontend.

The API response includes the road information, resolved regulatory classification, applicable setback distances, candidate counts, calculated distances and preliminary screening statuses. Explanatory reasons and verification flags provide additional context for interpreting the results. Candidate-level error handling also allows individual assessment failures to be isolated without automatically terminating the entire screening operation.

b) Configurable Regulatory Rules and Classification

A central component of the screening engine is the configurable regulatory rule system, which translates OpenStreetMap (OSM) road classifications into categories used for setback assessment.

OSM classes, including primary, secondary, tertiary, residential and unclassified, are mapped to the regulatory categories adopted by the project, including `main_road`, `district_road`, `branch_road` and `other_road`. Relevant route references and road attributes can also be used to support classification.

Default regulatory parameters are maintained in `config/setback-rules.json`, rather than being embedded directly within the application logic. The configuration records road classes, road-reserve widths, estimated centreline offsets, building-line distances, statutory sources and mappings between OSM classes and regulatory categories. This arrangement permits regulatory values and classification mappings to be updated without rewriting the underlying geometric processing functions.

The screening process also distinguishes eligible road features from excluded or unsupported features. Footways, pedestrian paths and unknown classes are not automatically treated as compliant. Where a valid regulatory control cannot be established, the system identifies the condition for verification rather than producing an unsupported compliance determination.

To accommodate differences between standard regulatory requirements and site-specific conditions, an eight-stage rule-resolution pipeline was implemented. It identifies the road's regulatory class, retrieves the standard control, identifies additional road-specific controls, resolves precedence between applicable controls, applies permitted adjustments, identifies

subject-specific overrides, resolves precedence between those overrides, and determines the final effective road control.

The precedence mechanism considers explicit priority and scope specificity. Supported scopes include roads, road sides, road segments, parcels and individual structures. This enables more specific controls to be considered alongside broader regulatory requirements.

Where applicable controls remain tied after the precedence criteria have been evaluated, the engine identifies the result as ambiguous and returns an `ADMIN_REVIEW_REQUIRED` status. The system therefore avoids selecting an arbitrary setback requirement when regulatory conditions cannot be resolved confidently. The resulting process provides a consistent basis for applying standard rules, administrator-managed assignments and subject-specific overrides (Section 4.9.1 and Appendix G.3).

c) Spatial Candidate Retrieval and Processing

The efficiency of the screening process depends on identifying relevant buildings without processing the entire geographic dataset for every request. The initial implementation relied on retrieving substantial quantities of geographic information and filtering candidate buildings in application memory. This introduced processing overhead, particularly when larger road and building datasets were involved.

The workflow was subsequently refactored to adopt a road-first screening approach. Rather than evaluating all buildings against all available roads, the process begins with a selected road and retrieves only buildings within the relevant screening area.

The backend invokes the PostGIS function `find_buildings_near_road` through Supabase Remote Procedure Calls (RPCs). Spatial filtering takes place within PostgreSQL/PostGIS using proximity operations, including `ST_DWithin`, supported by spatial indexing where configured. Only buildings within the specified screening distance are returned for detailed assessment.

The screening distance is determined from the applicable regulatory thresholds, including the road-reserve distance and any additional building-line requirement. This approach reduces unnecessary data transfer between the database and application server and limits subsequent geometric calculations to potentially affected structures.

The spatial retrieval function forms an important connection between the backend screening workflow and the database infrastructure maintained by Elijah Zulu. Its implementation required coordination over function parameters, geometry formats, database structures and query outputs to ensure that candidate features could be retrieved and processed consistently

d) **Geospatial Distance and Directional Analysis**

Geometric processing was implemented using Turf.js to calculate the minimum separation between road centrelines and candidate building geometries. The implementation accommodates supported GeoJSON geometry types, including Point, Polygon, MultiPolygon, LineString and MultiLineString.

The calculation process first checks whether the relevant geometries intersect. Where an intersection is identified, the minimum distance is recorded as zero. For non-intersecting features, the geometry-processing functions evaluate the relevant geometric components to determine the shortest separation between them.

The backend also supports the creation of setback zones around road geometries, providing the spatial basis for screening candidate buildings against the applicable thresholds.

Directional analysis determines whether a candidate lies to the left or right of a directed road segment. For a road segment extending from point (A) to point (B), and a candidate point (P), the two-dimensional cross-product is calculated as:

$$C=(B_x-A_x)(P_y-A_y)-(B_y-A_y)(P_x-A_x)$$

The sign of (C) indicates the candidate's relative position with respect to the directed segment, subject to the coordinate convention and geometry representation used. The implementation accounts for road direction and coordinate handling when evaluating side-specific setbacks.

This functionality enables the screening engine to distinguish candidate buildings on opposite sides of a road and apply different controls where asymmetric setback requirements are configured.

e) **Candidate Assessment and Compliance Classification**

Following candidate retrieval and geometric analysis, each building is assessed against the effective regulatory controls established by the rule engine.

The assessment distinguishes between the road-reserve zone and the additional building-line zone. The road-reserve threshold, represented by (X), defines the primary clearance distance from the road centreline, while the additional building-line distance, represented by (Y), extends the screening area beyond the primary reserve.

The combined outer screening threshold is expressed as:

$$D = X+Y$$

where (D) represents the total screening distance under the adopted assessment model.

Three principal outcomes are used to classify candidate structures:

- ROAD_RESERVE_CANDIDATE: the measured distance falls within the applicable road-reserve threshold.
- BUILDING_LINE_CANDIDATE: the measured distance exceeds the road-reserve threshold but remains within the combined road-reserve and building-line threshold.
- OUTSIDE_BUILDING_LINE_ZONE: the measured distance lies beyond the combined screening threshold.

Each assessment includes an explanatory reason and an appropriate verification flag to support interpretation of the result. The classification process provides a consistent mechanism for applying regulatory controls to the distances calculated by the geometry module.

These classifications represent preliminary screening outcomes rather than final legal determinations. The purpose of the automated assessment is to identify structures that may require further examination by municipal planning or enforcement officers. Cases involving ambiguous regulatory rules, unsupported road classifications or other verification requirements are flagged rather than treated as confirmed compliance decisions.

f) Backend Optimisation and Reliability

Several changes were introduced to improve the performance, maintainability and reliability of the screening workflow.

The initial implementation relied on external OpenStreetMap Overpass API requests to retrieve geographic features. This introduced network latency, potential rate-limiting errors and inconsistent response times. The principal screening workflow was therefore modified to use geographic data available through the PostgreSQL/PostGIS database.

The transition from broad in-memory candidate evaluation to targeted spatial queries reduced unnecessary retrieval and restricted detailed geometry processing to buildings within the relevant screening area.

The configuration-based rule system also reduced dependence on hardcoded conditional statements by separating regulatory parameters from application logic. In addition, geometry-processing functions were refined to address coordinate normalisation, multipart features and unexpected geometry structures. GeoJSON coordinate ordering was standardised to longitude and latitude, consistent with RFC 7946.

Road feature filtering was improved to distinguish eligible vehicular roads from pedestrian paths, footways and other features that should not automatically receive the same regulatory treatment as major roads.

Together, these modifications improved the consistency of candidate retrieval, reduced avoidable processing overhead and strengthened the handling of geometry-related failures and conflicting regulatory controls.

g) Backend Testing and Performance Evaluation

Backend testing was carried out using automated tests and command-line verification scripts. The `resolveRoadRule.test.js` suite assessed regulatory classification, default rule resolution, precedence handling and boundary conditions. Additional scripts, including `test-assess-candidate.js` and `test-assess-road.js`, supported independent testing of individual candidate assessments and complete road-screening operations without requiring the frontend.

Performance testing recorded approximately 0.33 seconds for 1,000 candidate assessments, equivalent to an average of approximately 0.33 milliseconds per candidate under the tested conditions. These results provide an indication of processing performance for the tested workload. They do not, by themselves, represent the total response time of a complete screening request, which also depends on database retrieval, network communication, geometry complexity and other processing overheads.

Testing and verification also helped identify issues involving coordinate ordering, invalid geometry structures, regulatory rule conflicts and database interactions. The resulting corrections improved the reliability of the screening workflow and supported more consistent assessment results (Sections 6.3.1 and 6.4).

5.7.3 Kupalwa Chibuye: Interface Design, Dashboard and Usability Evaluation

This subsection identifies the part of the work for which I, **Kupalwa Chibuye**, was primarily responsible and explains how it fits with the work of my team members, **Lubasi Sipangule** and **Elijah Zulu**. A fuller account is given in Section A.5 of Appendix A.

My responsibilities.

- **Interface design.** I designed how the system looks and flows for officers without GIS training: the map-plus-panel layout, the navigation bar, the colour scheme and the colour coding of results (red, green and amber), as described in Section 4.8 and shown in Figures 5 to 7.
- **Login and sign-up pages.** I designed and built the pages: a single centred form carrying the council's name and the system title, with clear loading, error and success messages. The authentication logic behind them was developed by Elijah Zulu.
- **Dashboard.** I implemented the dashboard page: the navigation bar, the road-ID input and Run button with a loading indicator, the road control details panel, the summary counters, the results table with colour-coded status badges and a toggle to show all candidates, and the inspection history list. I also wrote the client code that sends a road to the GIS service, converts the response into map features and saves the inspection and its candidates (Section 5.4.4).
- **Usability evaluation.** I carried out the evaluation of the system with the System Usability Scale: the online questionnaire and instructions (Appendix E), the collection of seventeen responses and the scoring and analysis reported in Section 6.5.

Work of other members. Lubasi Sipangule was responsible for the GIS service and its logic: the setback rules and configuration, road-class resolution, overrides and precedence, the distance calculations and the assessment of candidate buildings (Sections 4.9 and 5.4.1 to 5.4.3). Elijah Zulu was responsible for the Supabase database and migrations, the loading of the road and building data, authentication and validation, role-based access control, the data side of the supporting features, the mobile layout handling and the deployment on Render (Sections 5.4.4 and 5.4.5). I describe their components only where needed to explain how my work connects to them, and I do not claim them as my own.

Where the work meets. My components connect to theirs at three points. The dashboard and map consume the response of the GIS service, so the fields that service returns determine what the interface can show. The dashboard saves and reads inspections and candidates in the tables created by Elijah Zulu, under the signed-in user's session. The Admin Panel link and the

administrator pages are shown according to the role claim that his access-control layer provides.

5.7.4 System Integration and Collaborative Development

The effectiveness of the Road Setback Management System depended on the integration of its frontend, backend and database components. Although development responsibilities were allocated according to technical specialisation, the interfaces between the components required joint planning, implementation and verification.

A typical screening operation begins when a user selects a road through the mapping interface. The frontend submits the road identifier to POST /api/analyze, and the backend retrieves the relevant road information and regulatory assignments from the database. The rule engine resolves the effective controls, the spatial database identifies nearby candidate buildings, and the geometry and assessment modules calculate distances and assign preliminary classifications. The results are returned through the API for presentation on the frontend, while inspection and review information is supported by the database layer.

This workflow required coordination over API request and response formats, database query outputs, coordinate conventions and the interpretation of screening statuses. The handling of incomplete data, invalid geometries and unresolved rule conflicts also required consistent behaviour across the application layers.

Integration work included resolving issues that crossed component boundaries. For example, a map-loading issue associated with server-side rendering was addressed by ensuring that the affected map component was loaded only in the browser. The correction was completed by Lubasi Sipangule following initial work by Kupalwa Chibuye, illustrating how individual responsibilities could overlap during debugging and integration.

Overall, the division of responsibilities enabled the frontend, backend and database components to be developed and verified independently before integration. Kupalwa Chibuye's frontend development, Lubasi Sipangule's backend and geospatial screening implementation, and Elijah Zulu's database management and deployment collectively contributed to the completed system. Through their integration, the project produced a web-based GIS application capable of performing automated preliminary road setback screening, presenting geographic assessment results and supporting the administrative review of potentially non-compliant developments.

5.8 Chapter Summary

The system was implemented as a modular Express GIS service, a Next.js web client and a set of Supabase migrations. Rules are configuration-driven, candidate retrieval is performed in the database, each candidate is assessed independently, and access control is enforced in the database through JWT role claims and row-level security.

Chapter 6: Testing, Results and Evaluation

6.1 Introduction

This chapter reports how the Road Setback Management System was tested and evaluated against the objectives set out in Section 1.4. It describes the testing strategy, presents the results of the automated rule-engine tests, the system-level checks of the deployed application and the System Usability Scale (SUS) survey, interprets those results, and then assesses each project objective in turn.

6.2 Testing Strategy

Testing combined automated and manual techniques at several levels, reflecting the three-layer "screening, not deciding" framework of Section 2.6: the rule layer, the measurement layer and the human-verification layer were each tested on their own terms.

- **Unit testing (automated).** The rule-engine modules (regulatory classification, standard road controls, road-wide assignments, precedence resolution and subject-specific overrides) were tested with Node.js's built-in test runner (node:test). These modules depend only on the configuration file, so they can be tested without a database or network connection. 31 test cases were written and executed against the final code base (Appendix D).
- **Geometric measurement testing.** As described in Section 3.7, distance measurement is tested with synthetic roads and buildings placed at known distances near Lusaka (latitude about -15.4), allowing a tolerance of 0.5 m for geodesic approximation. These tests need the Turf.js libraries and are run separately from the rule-engine suite.
- **System and integration testing (manual).** The deployed application on Render (frontend, GIS service and Supabase) was exercised end to end: authentication, road selection, screening, persistence of results, the administrator panel and the manual report form.
- **Usability testing.** A System Usability Scale questionnaire (Appendix E) was completed by 17 respondents after using the deployed system.
- **Code and security review.** The full code base and database migrations were reviewed for defects, security weaknesses and inconsistencies between components (Appendix D, Table 23).

Two limitations of the test assets should be noted. First, the repository's original test folder contains a file named `resolveRoadRule.test.js` that holds test data in JSON form rather than executable tests, and two scripts (`test-assess-candidate.js` and `test-assess-road.js`) that only run against a live Supabase project and print their output. They are useful for manual inspection but are not automated tests. Second, the PostGIS function `find_buildings_near_road()` is not stored in the versioned migrations, so the candidate-retrieval step could not be tested in isolation from the live database.

6.3 Test Cases and Results

6.3.1 Automated rule-engine tests

The rule-engine suite (`backend/test/ruleEngine.test.js`) was run with `node --test` on Node.js v22.22.2. Table 12 summarises the results by group and Table 13 lists representative cases. The complete list of 31 cases is given in Appendix D (Table 22).

Table 12: Summary of automated rule-engine test results

Group	Area tested	Cases	Passed	Failed
RC-01 to RC-09	Regulatory classification of OSM road classes	9	9	0
SC-01 to SC-04	Configured standard road controls (reserve width, centreline offset, building line)	4	4	0
RA-01 to RA-07	Road-wide assignments (uniform, side-specific, priority, ambiguity)	7	6	1
RP-01 to RP-04	Precedence between controls (priority, scope, ties)	4	4	0
RO-01 to RO-07	Subject-specific overrides (building, parcel, feature type, ambiguity)	7	6	1
Total		31	29	2

Table 13: Representative rule-engine test cases

ID	Test case	Expected result	Status
RC-05	Residential, service, unclassified and primary_link roads	Fallback class other_road; flagged for verification	Pass
RC-06	Footway, path, track, steps and cycleway	Excluded from screening; no regulatory class	Pass
RC-07	Unknown OSM class (motorway)	Unresolved; not silently mapped to other_road	Pass

ID	Test case	Expected result	Status
SC-01	Configured reserve widths and centreline offsets	trunk 100/50 m, main 60/30 m, district, branch and urban 36/18 m, park and other 18/9 m	Pass
SC-02	Building-line distance (Y) for every class	Unresolved by default; verification required	Pass
RA-02	Uniform road-wide assignment of 12 m on a secondary road	Offset replaced by 12 m; source road_rule_assignment	Pass
RA-05	Two equal-priority assignments on one road	Resolution status AMBIGUOUS; verification required	Pass
RP-02	Structure and parcel overrides at equal priority	Structure wins (more specific scope)	Pass
RP-04	Two controls with equal priority and scope	AMBIGUOUS with both listed as conflicts	Pass
RO-04	Building override on building 7 only	Offset 3 m for building 7; class default for building 8	Pass
RA-07	Side-specific road assignment (left 6 m, right 8 m)	Road-level offset unresolved (null), so the road is reported as unresolved	Fail
RO-06	Override with scope "structure"	Override applied (scope accepted by finder and precedence table)	Fail

The two failing cases exposed genuine defects rather than faulty tests. They are analysed in Table 14.

Table 14: Analysis of the failed test cases

ID	Observed behaviour	Cause	Impact and recommended fix
RO-06	Throws "Unsupported road control override scope: structure".	findRoadControlOverrides() and the precedence table accept the scope "structure", but applyRoadControlOverride() lists only "building" among its supported scopes.	Structure-level exceptions cannot be applied. assessRoad() catches the error per candidate, so the building is reported as unknown with a reason and the road assessment continues (NFR-04 holds). Fix: add "structure" to the supported scopes.
RA-07	A side-specific assignment leaves the class-default centreline offset (18 m for a secondary road) in force.	applyRoadControlAdjustment() keeps the base road reserve when the assignment is side-specific, so the "unsupported" marker set earlier is never acted on.	The road is silently screened with the class default instead of being reported as unresolved, contradicting the design in Section 4.9.4. Fix: set the offset to null (or check unsupportedControlType) for side-specific assignments until directional analysis is implemented.

6.3.2 System-level tests on the deployed application

Table 15 records the system-level checks carried out on the application deployed on Render. Evidence for each check is the corresponding screen in Chapter 4 (Figures 5 to 7) or the feedback collected in the usability survey.

Table 15: System-level test cases and outcomes

ID	Test case	Observed outcome	Status
ST-01	Accounts and sign-in	The administrator panel lists eight accounts with confirmation state, creation date and last sign-in time (Figure 7).	Pass
ST-02	Map shows the road network and a road can be selected by ID	Dashboard loads the road layer; road 1739 was selected by ID (Figure 6).	Pass
ST-03	Run screening on a road	Road 1739 returned 49 candidate buildings: 32 potential issues, 3 compliant and 14 for review, drawn with the setback zone (Figure 6).	Pass
ST-04	Result explains the rule applied	Panel shows road name, regulatory class other_road, configured reserve width, screening distance, rule source "Default (road class rule)" and "Verification required: Yes" (Figure 6).	Pass
ST-05	Inspection and candidates are stored	Candidates 2172637, 2172509 and 2172527 from Road 1739 appear in the administrator candidate list with measured and required distances (Figure 7).	Pass
ST-06	Administrator can review a candidate status and view its audit trail	Status selectors and "View" audit-trail buttons are present for each candidate (Figure 7).	Pass
ST-07	Administrator can manage user roles	Role selector per user; the signed-in administrator is marked "admin (you)" (Figure 7).	Pass
ST-08	Field officer can log a violation without running an analysis	Report form offers category, type, requested action, description, road and candidate IDs, location capture and photo upload (Figure 5).	Pass (form displayed)
ST-09	Run with no valid road selected	Survey respondent received the raw message "GIS server error: 500 ... Cannot coerce the result to a single JSON object" instead of a friendly prompt.	Fail
ST-10	Select a road on a mobile browser	One respondent could not select specific roads on a mobile browser.	Fail / partial

6.4 Experimental Results / Performance Results

Non-functional requirement NFR-01 asks that the engine process each candidate in well under 10 ms and that a road be assessed within a few seconds. Two measurements bear on this. First,

the timing run described in Section 3.7 assessed 1,000 consecutive synthetic candidates in about 0.33 s, or roughly 0.33 ms per building, about thirty times faster than the 10 ms target. This figure covers the assessment engine only and excludes network and database time. Second, the 31-case rule-engine suite completed in about 0.13 s in total on the test machine.

End-to-end latency of a screening run on the hosted system (browser to Render to Supabase and back) was not measured formally, so NFR-01 is demonstrated for the engine but only observed informally for the whole system. The screening of road 1739 in Figure 6, which returned 49 candidates, was completed interactively without noticeable delay.

6.5 User Evaluation

Usability was evaluated with the 10-item System Usability Scale [18], delivered as an online questionnaire (Appendix E) together with step-by-step instructions for using the system. Seventeen respondents completed it. Table 16 and Figure 9 show the distribution of responses to each item.

Table 16: SUS responses by item (n = 17; 1 = strongly disagree, 5 = strongly agree)

Item	Statement	1	2	3	4	5	Mean
Q1	I think that I would like to use this system frequently.	1	0	5	5	6	3.88
Q2	I found the system unnecessarily complex.	3	5	5	2	2	2.71
Q3	I thought the system was easy to use.	0	2	5	6	4	3.71
Q4	I think that I would need the support of a technical person to be able to use this system.	2	5	4	4	2	2.94
Q5	I found the various functions in this system were well integrated.	0	1	2	9	5	4.06
Q6	I thought there was too much inconsistency in this system.	4	7	5	1	0	2.18
Q7	I would imagine that most people would learn to use this system very quickly.	1	0	5	6	5	3.82
Q8	I found the system very cumbersome to use.	5	9	3	0	0	1.88
Q9	I felt very confident using the system.	0	1	6	4	6	3.88
Q10	I needed to learn a lot of things before I could get going with this system.	5	6	4	1	1	2.24

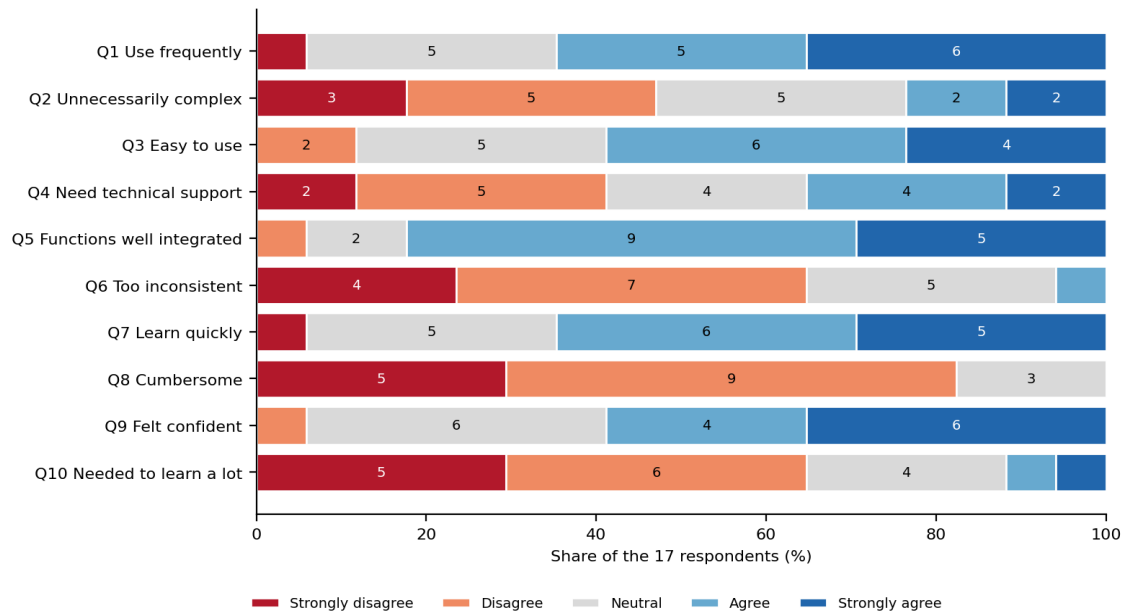


Figure 9: Distribution of responses to the SUS items

Applying the standard SUS scoring rule (odd items scored as response minus 1, even items as 5 minus response, summed and multiplied by 2.5) to the response counts gives an overall SUS score of **68.5 out of 100**. This is in line with the commonly cited average of about 68 [19], that is, "average" usability: acceptable for a prototype but with room for improvement. Because the score was computed from the aggregated item distributions in Table 16 rather than from individual response sheets, the spread of individual scores is not reported.

The strongest results were for integration and ease of handling: 82.4% agreed or strongly agreed that the functions were well integrated (Q5), and 82.4% disagreed or strongly disagreed that the system was cumbersome (Q8). About two-thirds disagreed that it was inconsistent (Q6) or that they had to learn a lot before starting (Q10), and 64.7% would like to use it frequently (Q1) and expected most people to learn it quickly (Q7). The weakest results were for perceived complexity (Q2, 23.5% agreed it was unnecessarily complex) and the need for technical support (Q4, 35.3% agreed they would need it, against 41.2% who disagreed). These two items contributed least to the overall score.

Four respondents left free-text comments. They asked for (i) a clear, user-friendly message when no road has been selected, in place of the raw 500 error shown in ST-09; (ii) testing by someone from Lusaka City Council who would use the system regularly; (iii) a more reliable way to select roads on a mobile browser; and (iv) a clear search or reset option so that a new query can be started without using the browser's back button. One respondent found everything "just okay".

The survey has limitations that qualify these results. The sample is small (17), the respondents' professional backgrounds were not recorded, and one comment suggests that few or none were LCC enforcement officers, so the results indicate general usability rather than acceptance by the intended users. The questionnaire form also used "tick all that apply" for each item, although every item received exactly 17 responses, and the wording of the neutral option contains a typographical error in some items. A structured evaluation with enforcement officers remains outstanding.

6.6 Analysis of Results

The automated results show that the rule layer behaves as designed in the great majority of cases: classification, class defaults, road-wide assignments, precedence and ambiguity detection all passed. In particular, the design principle that missing or unresolved information must never read as compliance is supported by RC-06, RC-07, SC-02 and SC-03, where unknown, excluded or unresolved inputs produce unresolved results and verification flags. The two failures are located in the less commonly used paths (structure-scope exceptions and side-specific assignments), and both are small, well-localised fixes. The more serious of the two, RA-07, is serious precisely because it fails silently, and it is a reminder that every unsupported path must fail visibly.

The system-level checks show that the principal workflow works on the deployed system: select a road, resolve its rule, retrieve nearby buildings, classify them, show the setback zone and persist the run for administrative review. The failed checks (ST-09 and ST-10) are usability and robustness problems at the edges of that workflow rather than failures of the core analysis, and the SUS score of 68.5 is consistent with that picture: a usable but unpolished prototype.

The code review (Appendix D, Table 23) identified further issues that automated tests do not cover, chiefly an unauthenticated GIS endpoint that uses the service-role key, schema drift between the frontend and the versioned migrations, and a mismatch between the field names the dashboard reads and those the current assessment engine returns. These do not invalidate the screening logic but must be resolved before any operational deployment.

6.7 Objective-by-Objective Evaluation

Table 17: Evaluation of the project objectives

Objective	Evidence	Assessment
1. Configurable rule engine with deterministic precedence	Rules held in config/setback-rules.json and the road_rule_assignments table; 29 of 31 rule tests pass, including priority, scope specificity and tie detection (RP-01 to RP-04, RA-05).	Largely achieved. Two defects (RO-06, RA-07) remain; side-specific rules are represented but not evaluated.
2. Geospatial screening: retrieve candidates and measure distance	Road-first PostGIS retrieval and Turf.js measurement implemented; engine timing of about 0.33 ms per building; road 1739 screened with 49 candidates.	Achieved for screening. Results are zone-based (inside the reserve or building-line band) rather than final compliance verdicts, and field validation of distances was not done.
3. Web map interface for authenticated officers	Dashboard with map, road selection, setback zone, colour-coded candidates, results table and history (Figures 5 and 6); SUS score 68.5.	Achieved. Error handling for invalid road selection and mobile road selection need improvement (ST-09, ST-10).
4. Role-based access control and administrative workflow	Roles in app_metadata, row-level security, set_user_role(), candidate status review with audit trail, road-rule override management (Figure 7, Appendix F).	Achieved. Backend /api/analyze is not yet authenticated (Table 23).
5. Evaluate correctness and performance through systematic testing	31 automated rule tests, 10 system-level checks, performance timing, SUS survey (n = 17) and code review.	Partially achieved. Geometry and database-dependent paths lack an automated suite in the repository, and no test with LCC officers or field measurements was carried out.

6.8 Individual Results (group projects)

The results attributable to each member's contribution are presented in turn: Elijah Zulu (Section 6.8.1), Lubasi Sipangule (Section 6.8.2) and Kupalwa Chibuye (Section 6.8.3).

6.8.1 Elijah Zulu

The results attributable to my contribution concern the data layer, authentication, access control, supporting features and deployment. They are drawn from the system-level checks in Table 15, the code review in Table 23 and the usability survey, and are summarised below.

Area	Evidence	Outcome
Accounts and sign-in	ST-01: the administrator panel lists eight accounts with confirmation state, creation date and last sign-in time.	Pass

Area	Evidence	Outcome
Storing inspections	ST-05: candidates from road 1739 appear in the administrator list with measured and required distances.	Pass
Review and audit trail	ST-06: status selectors and an audit-trail view are present for each candidate.	Pass
Role management	ST-07: a role selector is shown for each user; the signed-in administrator is marked "admin (you)".	Pass
Field reports	ST-08: the report form offers location capture and photo upload.	Pass (form displayed)
Hosted deployment	All system-level checks were run on the application deployed on Render with the hosted Supabase project.	Working
Mobile use	ST-10: one respondent could not select specific roads on a mobile browser.	Fail / partial
Access-control and data issues found in review	Table 23, items 1, 6 and 8: page protection is client-side only; the roads and buildings tables, spatial function and report-form objects are not in the versioned migrations; insert policies are permissive and an administrator e-mail is hard-coded in a migration.	Open

Taken together, the results show that the parts I was responsible for work as designed on the deployed system: accounts can be created and listed, inspections and candidate buildings are stored and reviewable, roles can be changed by administrators only, and changes to candidate status leave an audit trail. The usability survey supports this indirectly, since 82.4% of respondents agreed that the functions of the system were well integrated (Q5), although that result reflects the work of all three members and cannot be attributed to one.

The same evidence shows that my area carries most of the open security and maintainability issues. Page protection currently relies on the client, the database objects created outside the migration files mean the database cannot be fully recreated from the repository, and the insert policies are broader than they need to be. The mobile road-selection problem reported in ST-10 also remains. The first improvements I would make are to verify the session in the middleware and the proxy route, to move all database objects into the migrations, to restrict inserts to the signed-in user, and to move administrator seeding out of the migration files.

6.8.2 *Lubasi Sipangule*

This section summarises the evaluation of the key system components, including the backend GIS rule engine and the geospatial measurement logic, evaluated against the project requirements. The results are drawn from the automated unit test suite (Section 6.3.1),

performance timing benchmarks (Section 6.4) and the system-level checks on the deployed application (Table 15).

Area	Evidence	Outcome
Regulatory classification	RC-01 to RC-09: all nine cases pass, including footways excluded from screening (RC-06) and an unknown class left unresolved rather than mapped to other_road (RC-07).	9 of 9 pass
Standard road controls	SC-01 to SC-04: reserve widths and offsets as configured; building-line distance unresolved by default (SC-02).	4 of 4 pass
Road-wide assignments	RA-01 to RA-07: priority and ambiguity handled; RA-07 shows that a side-specific assignment silently leaves the class default in force.	6 of 7 pass
Precedence	RP-01 to RP-04: priority outranks scope, scope breaks ties, equal controls are ambiguous.	4 of 4 pass
Subject-specific overrides	RO-01 to RO-07: RO-06 shows that an override with scope "structure" throws an error.	6 of 7 pass
Engine performance	1,000 consecutive assessments in about 0.33 s (about 0.33 ms per building), roughly thirty times faster than the 10 ms target; the 31-case suite ran in about 0.13 s.	Target met for the engine
End-to-end screening	ST-03 and ST-04: road 1739 returned 49 candidates with the rule applied, its source and a verification flag.	Pass
Backend issues found in review	Table 23, items 1, 2, 9, 10, 11 and 12: unauthenticated analysis endpoint using the service-role key with open CORS; diagnostic endpoint; obsolete entry point; data file in place of an executable test; oversized request limit; inconsistent versions. Items 3 and 4 correspond to RA-07 and RO-06.	Open

Taken together, the results show that the rule layer behaves as designed in the great majority of cases. Classification, class defaults, road-wide assignments, precedence and ambiguity detection all passed, and the design principle that missing or unresolved information must never read as compliance is supported by RC-06, RC-07, SC-02 and SC-03. The engine is also fast enough for interactive use: about 0.33 ms per building is far inside the target of 10 ms, although this figure excludes network and database time.

The two failures are in the less commonly used paths and are small, well-localised fixes. RO-06 arises because the override module lists only the scope "building" while the finder and precedence table accept "structure". RA-07 is the more serious of the two because it fails silently: a side-specific assignment is screened with the class default instead of being reported as unresolved. The first improvements I would make are to return an unresolved status for side-

specific assignments until directional analysis exists, to add the "structure" scope, to authenticate the analysis endpoint and restrict its CORS origin, to remove the diagnostic endpoint and reduce the request limit, to delete the obsolete entry point, and to replace the data file in the test folder with executable tests. Two limits qualify these results: no distances were compared with surveyed measurements, and the geometry and database-dependent paths have no automated suite in the repository.

6.8.3 Kupalwa Chibuye

The results attributable to my contribution concern the interface, the map visualisation and the usability evaluation. They are drawn from the system-level checks in Table 15, the code review in Table 23 and the usability survey, and are summarised below.

Area	Evidence	Outcome
Usability	SUS score of 68.5 from 17 respondents; 82.4% agreed the functions were well integrated (Q5) and 82.4% disagreed that the system was cumbersome (Q8).	Average usability
Interface issues found in review	Table 23, items 5, 7 and 13: the dashboard reads fields the assessment engine does not return; raw database errors reach the user; the label "violation" reads as a legal determination.	Open

6.9 Chapter Summary

Testing combined 31 automated rule-engine tests (29 passed, 2 exposed genuine defects), ten system-level checks on the deployed application, an engine timing run well inside the performance target, a SUS survey of 17 respondents yielding a score of 68.5, and a code review. Together they show that the core screening workflow works and that the system is reasonably usable, while identifying the defects, security gaps and validation steps that remain.

Chapter 7: Discussion

7.1 Introduction

This chapter interprets the results of Chapter 6 in relation to the problem, the research questions and the existing work reviewed in Chapter 2.

7.2 Interpretation of Findings

Representing rules as data (RQ1). The rules-as-data approach worked: the applicable distance for a road is resolved automatically from its OSM class, then adjusted by road-wide assignments and subject-specific overrides, and every result carries the source of the rule, a reason and a verification flag. Testing showed that precedence and ambiguity handling are deterministic, which is what makes the result explainable. The model also changed during development. The initial design (Table 9) used a single setback distance per class, measured from the road centreline. The final engine instead distinguishes an estimated road-reserve centreline offset (X), derived from the statutory reserve width, from an additional building-line distance (Y) that is unresolved unless an administrator supplies it. This reflects the legal reality that a reserve and a building line are different controls, and it forced the system to say "unresolved" where the data do not exist rather than invent a distance. As a result, the statuses reported by the engine describe where a building lies relative to a screening zone (within the estimated reserve, within the building-line band, or outside), and the dashboard presents them as potential issue, compliant or needs review.

Measurement accuracy and efficiency (RQ2). Retrieving only buildings near the selected road in PostGIS and measuring them with Turf.js made the engine fast: about 0.33 ms per building. The accuracy of the numbers, however, is bounded less by the algorithm than by the data and the reference line. Distances are measured from an OSM centreline to an OSM footprint, so positional error in either layer, or an unmapped building, passes straight into the result. For this reason the figures should be treated as indicative, which is consistent with the verification flag carried by every result.

Supporting rather than replacing officers (RQ3). The design keeps people in control: nothing is presented as a legal determination, unresolved cases are marked for review, administrators can override candidate status with an audit trail, and field officers can log

observations that disagree with the data through the report form. The survey supports the view that officers could adopt the tool without specialist training (about two-thirds disagreed that they needed to learn a lot first), although this needs confirmation with the actual users. The scale of the output also deserves comment. For road 1739, 32 of 49 candidates were flagged as potential issues. Because the candidate search envelope is the screening zone itself, the flagged count measures how many buildings lie inside the zone, not how many are confirmed violations. A count of that size demonstrates why prioritisation and verification, not automatic enforcement, is the appropriate use.

7.3 Comparison with Existing Work

Akingbade et al. used GIS and remote sensing in Ile-Ife, Nigeria, and found 6,284 buildings in contravention of road setbacks, showing that spatial analysis can detect violations at scale. Their work was a one-off assessment. The present system applies the same underlying idea of measuring buildings against road-based limits, but packages it as a reusable tool with configurable rules, persistent records and review workflows. Omollo's study in Kisii, Kenya, attributed non-compliance largely to weak enforcement capacity rather than ignorance of the rules. That finding supports the design emphasis on speed, consistency and an audit trail for officers, but it also implies that a screening tool alone will not change outcomes unless the institution acts on its results.

Against the approaches compared in Table 1, the system occupies the gap identified in Section 2.5. Compared with manual inspection it screens a road in seconds and keeps a per-building record; compared with desktop GIS it removes the need for specialist skills and encodes rule selection; compared with enterprise web GIS it relies entirely on open-source software and open data. Like open building-footprint datasets such as Google's Open Buildings [13], it supplies candidate evidence but not an authoritative measurement, so it complements rather than replaces field verification. Its principal weakness relative to the alternatives is evaluation: manual inspection is authoritative, and this project has not yet validated its distances against surveyed measurements.

7.4 Contributions of the Project

- **Practical:** a working, hosted web tool that lets an officer screen the buildings along any road in the loaded Lusaka dataset in a few actions and keep a history of the results.
- **Technical:** a rules-as-data engine with explicit provenance, deterministic precedence, separation of reserve (X) and building line (Y), and safe handling of unresolved and ambiguous cases; a road-first PostGIS retrieval design; and database-enforced role-based access with audit trails.
- **Institutional:** a transparent, explainable record of what was screened, under which rule and with what outcome, which supports consistent enforcement and later review.
- **Methodological:** an evaluation approach for a screening tool that combines synthetic-geometry tests, rule-engine tests, user survey and code review, and which documents defects openly.

7.5 Limitations and Implications

- **Data quality.** OSM completeness and positional accuracy vary, and unmapped or misplaced buildings affect results directly. Results are therefore screening indications only.
- **Reference geometry and rules.** Measurement is from the road centreline, and the reserve offset is an estimate from the configured statutory width. The OSM-to-regulatory class mapping is provisional and Y is unresolved unless configured, so every result requires verification. The statutory basis and its applicability to local-authority roads should be confirmed with LCC and the road authorities.
- **Unvalidated accuracy.** No comparison was made with surveyed or field-measured distances, so the real-world error of the measurements is unknown.
- **Open defects.** The failures in Table 14, the unauthenticated backend endpoint, schema drift and front-end/back-end contract mismatches (Table 23) mean the prototype is not ready for operational use.
- **Evaluation sample.** The SUS sample was small and probably not made up of enforcement officers, so usability results cannot be generalised to the intended users.

The implication is that the system is best presented as a prototype decision-support tool: useful for prioritising inspections, but not yet for generating evidence in enforcement proceedings.

7.6 Chapter Summary

The findings support the central idea of the project: configurable, explainable, open-source GIS screening is feasible and fast, and can keep officers in control. They also show that the quality of the outcome depends on input data, rule configuration and verification, and that the prototype needs the fixes and validation set out in the next chapter.

Chapter 8: Conclusion and Recommendations

8.1 Introduction

This chapter summarizes the project, states its conclusions, assesses the achievement of the objectives and makes recommendations and suggestions for future work.

8.2 Summary of the Project

Setback enforcement in Lusaka relies largely on manual site inspection, which is slow, inconsistent and poorly recorded. The project designed and built the Road Setback Management System, a three-tier web application consisting of a Next.js client with a Leaflet map, an Express/Node.js GIS service using Turf.js, and a Supabase PostgreSQL/PostGIS database with authentication and row-level security. The system resolves the applicable road-reserve and building-line controls from configuration data and road-specific assignments, retrieves nearby buildings with a spatial query, measures their distance to the road centreline, classifies them with an explanation and a verification flag, and stores each inspection for review by administrators. It was deployed on Render and evaluated through automated rule tests, system-level checks, performance timing, a 17-respondent SUS survey and a code review.

8.3 Conclusions

- Setback rules, road-specific assignments and exceptions can be represented as data and resolved deterministically and explainably, with unresolved and ambiguous cases surfaced rather than hidden.
- Open data (OpenStreetMap), PostGIS and Turf.js are sufficient to screen the buildings along a road in a fraction of a second per candidate and to present the results on an interactive map for non-specialist users.
- The system achieved a SUS score of 68.5, indicating average usability for a first prototype, with identified improvements in error handling and mobile use.
- The tool is a screening aid. Its outputs depend on the quality of open data and provisional rule mappings and must be verified in the field before any enforcement action.

8.4 Achievement of Objectives

- **Objective 1 (rule engine): largely achieved.** The configurable engine with deterministic precedence is implemented and tested; two defects (structure-scope overrides and side-specific assignments) must be corrected.
- **Objective 2 (screening component): achieved.** Candidate retrieval and distance measurement work, producing zone-based screening statuses; field validation of distances remains to be done.
- **Objective 3 (web map interface): achieved.** Authenticated officers can select a road, view the setback zone and screened buildings and consult their inspection history.
- **Objective 4 (access control and administration): achieved.** Roles, row-level security, status review with audit trail and role management are in place; the GIS endpoint still needs authentication.
- **Objective 5 (evaluation): partially achieved.** Systematic testing was carried out, but geometry and database tests are not yet automated in the repository and no evaluation with LCC officers or field measurement was possible within the project period.

8.5 Recommendations

- **Pilot with Lusaka City Council.** Run a supervised pilot with enforcement officers, repeat the SUS evaluation with them, and compare screening results with field measurements on a sample of roads.
- **Confirm the regulatory basis.** Have LCC and the road authorities confirm the reserve widths, the OSM-to-regulatory class mapping and the building-line distances (Y) that should be configured, and record the legal reference for each in the configuration.
- **Fix the known defects before deployment.** Correct RO-06 and RA-07, authenticate the GIS endpoint and the proxy route, restrict CORS, remove the diagnostic /api/db-test endpoint, and bring the frontend and the engine response format into agreement.
- **Make the database reproducible.** Add the roads and buildings table definitions, the find_buildings_near_road() function, the manual_reports columns and the storage bucket to the versioned migrations.
- **Improve usability.** Replace raw server errors with clear messages, add a visible search or reset control, and fix road selection on mobile browsers.

- **Keep the wording consistent with the screening principle.** Prefer labels such as "potential issue" over "violation" for automated results, and keep the verification flag visible wherever results are exported or printed.

8.6 Future Work

- Implement side-specific (left and right) setback analysis, which the rule model already represents.
- Measure from the surveyed edge of the road reserve instead of the centreline, using cadastral or survey data.
- Integrate the cadastral (parcel) register and the building-permit system so that parcel- and structure-level exceptions are populated from authoritative sources.
- Supplement OSM with automatically extracted footprints such as Google Open Buildings to improve completeness.
- Add automated geometry, API and end-to-end tests with continuous integration, and measure end-to-end latency under load.
- Provide an administrator interface for building-line distances and exceptions, bulk analysis of several roads, report export, and offline support for field officers.

References

- [1] K. Chama, R. Lifuka, M. Masonga, and D. Phiri, *ZAMBIA: LUSAKA URBAN PROFILE*. 2007. Available: <https://www.ircwash.org/sites/default/files/UN-HABITAT-2007-Zambia.pdf>
- [2] “[No title].” Available: <https://www.parliament.gov.zm/sites/default/files/documents/acts/The%20Urban%20and%20Regional%20Planning%20%20Act%2C%202015.pdf>. [Accessed: Oct. 02, 2026]
- [3] “[No title].” Available: <https://www.parliament.gov.zm/sites/default/files/documents/acts/The%20Urban%20and%20Regional%20Planning%20%20Act%2C%202015.pdf>. [Accessed: Oct. 02, 2026]
- [4] “Website.” Available: <https://www.openstreetmap.org>
- [5] “[No title].” Available: https://media.zambialii.org/media/legislation/35226/source_file/0bf1976686f892b1/zm-act-2002-12-publication-document.pdf. [Accessed: Oct. 18, 2026]
- [6] “The Urban and Regional Planning Act, No. 3 of 2015,” *Zambialii*. Available: <https://zambialii.org/akn/zm/act/2015/3/eng@2015-08-14>
- [7] “Website.” Available: <https://turfjs.org>
- [8] “Website.” Available: <https://www.openstreetmap.org>
- [9] “GeoJSON.” Available: <https://geojson.org>. [Accessed: Oct. 23, 2026]
- [10] “Website.” Available: <https://postgis.net>
- [11] “Leaflet — an open-source JavaScript library for interactive maps.” Available: <https://leafletjs.com>. [Accessed: Oct. 23, 2026]
- [12] “Research Portal.” Available: https://openresearch.surrey.ac.uk/discovery/fulldisplay?vid=44SUR_INST:ResearchRepository&docid=alma99257978302346&context=L. [Accessed: Oct. 23, 2026]
- [13] W. Sirko *et al.*, “Continental-Scale Building Detection from High Resolution Satellite Imagery,” July 26, 2021. Available: <http://arxiv.org/abs/2107.12283>. [Accessed: Sept. 27, 2026]
- [14] “Manifesto for Agile Software Development,” 2001. Available: <https://agilemanifesto.org>. [Accessed: Sept. 27, 2026]
- [15] K. Schwaber and J. Sutherland, *The Scrum Guide. The Definitive Guide to Scrum: The Rules of the Game*. 2020. Available: <https://scrumguides.org>. [Accessed: Sept. 27, 2026]
- [16] “Home.” Available: <https://scrumguides.org>. [Accessed: Oct. 03, 2026]
- [17] “Website.” Available: <https://www.openstreetmap.org>
- [18] J. Brooke, “SUS: A ‘Quick and Dirty’ Usability Scale,” in *Usability Evaluation In Industry*, CRC Press, 1996. Available: <https://api.taylorfrancis.com/content/chapters/edit/download?identifierName=doi&identifierValue=10.1201/9781498710411-35&type=chapterpdf>
- [19] J. Sauro and J. R. Lewis, *Quantifying the User Experience: Practical Statistics for User Research*, 2nd ed. Morgan Kaufmann, 2016.

Appendix A: Individual Contribution Statement (Group Projects)

A.1 Project and Group Information

Table 18: Project and group information

Item	Details
Project Title	Road Setback Management System: A Web-Based GIS Tool for Screening Building Setback Compliance along Roads in Lusaka
Supervisor	Dr Lighton Phiri
Group Size	3
Group Members	Kupalwa Chibuye (2021424278) Elijah Zulu (2021385299) Lubasi Sipangule (2022065786)

A.2 Overall Group Project

The group developed the Road Setback Management System, a web-based GIS tool that helps Lusaka City Council enforcement officers screen the buildings along a selected road for possible road-setback encroachment. The common problem is that setback enforcement is manual, slow, inconsistent and poorly recorded. The system resolves the applicable road-reserve and building-line controls from configuration data and road-specific assignments, retrieves candidate buildings with a PostGIS spatial query, measures their distance to the road with Turf.js, classifies them with an explanation and a verification flag, and records every inspection for administrative review. It consists of a Next.js web client with a Leaflet map, an Express/Node.js GIS service and a Supabase (PostgreSQL/PostGIS) database, and is deployed on Render.

A.3 Individual Contribution Statement: Elijah Zulu

A.3.1 Individual Role and Responsibilities

No.	Area of Responsibility	Description
1	Database design and data management	Design, creation and management of the Supabase (PostgreSQL/PostGIS) database: Lusaka roads and buildings, application tables, constraints, indexes, triggers and migrations.
2	Authentication and validation	Sign-up and sign-in with Supabase Auth, input validation, e-mail confirmation, session handling and redirection of unauthenticated users.
3	Access control	Administrator and user roles in app_metadata, row-level security policies, table grants, the profiles table and the set_user_role() function.
4	Supporting features and mobile experience	Data structures and integration for road rule overrides, candidate audit trail and manual field reports with photo upload; forced-landscape layout and map touch correction for phones.
5	Deployment	Hosting of the application on Render connected to the hosted Supabase project, and configuration of environment variables and keys.

A.3.2 Detailed Individual Contribution

The work I personally undertook falls into the following activities.

- **System and database design.** I designed the relational structure that stores inspections and the building candidates screened in each run, user profiles, road rule overrides, the status-change history and manual reports, together with the relationships, constraints and indexes that connect them (Section 4.7).
- **Data collection and preparation.** I obtained the road and building layers for Lusaka and loaded them into the database so that the system could screen real roads and buildings, and set up the database-side retrieval of buildings near a selected road.
- **Software development.** I developed the authentication flow, the role-based access control, the administrator and user views, and the data side of the supporting features described in Section 5.7.
- **API and external-service integration.** I integrated the Next.js client with Supabase Auth, the PostgreSQL database, the set_user_role() function and Supabase Storage for report photos, using the browser, server and middleware clients provided by @supabase/ssr.

- **Testing and debugging.** I diagnosed and fixed the "permission denied" failures that occurred even though row-level security policies existed, by adding the missing grants, and I checked the behaviour of roles, policies and sessions on the deployed system.
- **Deployment and configuration.** I deployed the application on Render, connected it to the hosted Supabase project and configured its environment variables.
- **Documentation.** I documented the database design, access control and deployment in the sections of this report that cover them.

A.3.3 Individual Deliverables

No.	Deliverable	Student's Contribution
1	Supabase database and migrations	Created the database, loaded the Lusaka roads and buildings, and wrote the four migration files.
2	Authentication module	Sign-up and login logic, input validation, session clients and middleware.
3	Role-based access control	Role claims, row-level security policies, grants, profiles table with triggers, and set_user_role().
4	Administrator and user views	Role gate for the administrator panel and separation of administrator and normal-user functions.
5	Supporting feature tables	road_rule_assignments, candidate_status_history, manual_reports and the report-photo storage bucket.
6	Mobile layout support	Forced-landscape hook and Leaflet touch-coordinate correction.
7	Deployed application	Live deployment on Render connected to the hosted Supabase project.

A.3.4 Contribution to the Main System or Research Output

The components of the final system that I developed or completed are the database and its migrations, the authentication and session handling, the access-control layer, and the deployment. Together they form the data, security and hosting foundation on which the GIS logic developed by Lubasi Sipangule and the interface designed by Kupalwa Chibuye run. The administrator, review and reporting features were built jointly: I developed the database structures, policies and integration, and Kupalwa designed how they appear to the user. The screening logic itself, including the rule engine and distance calculations, was developed by Lubasi and is not my work.

A.3.5 Individual Implementation and Technical Work

Database. The migrations in supabase/migrations create the inspections, building_candidates, profiles, road_rule_assignments, candidate_status_history and manual_reports tables. They

use UUID primary keys and cascading foreign keys, indexes on the columns used for lookups, and check constraints that keep data valid: a road rule override must be either uniform or side-specific and must carry the distances that its type requires, there is at most one override per road, and a manual report may only be open, reviewed, resolved or dismissed. A trigger keeps `updated_at` current, and a second trigger writes every change of candidate status to `candidate_status_history`, which the application cannot write to directly.

Authentication and access control. The sign-up page rejects mismatched passwords and passwords shorter than eight characters before calling Supabase Auth. The files `utils/supabase/client.ts`, `server.ts` and `middleware.ts` create the browser, server and middleware clients, and `frontend/middleware.ts` refreshes the session on every request. The dashboard, report and administrator pages check the signed-in user and redirect to the login page when there is none, and the administrator page additionally requires the admin role in `app_metadata`. Row-level security policies read the same claim from the signed token, so the database enforces what the interface only displays. The functions `handle_new_user`, `handle_user_update`, `log_candidate_status_change` and `set_user_role` run as SECURITY DEFINER with a fixed search path; `set_user_role` checks that the caller is an administrator and that the new role is admin or user before it writes to `auth.users` and `profiles`. Manual reports can only be inserted with `reported_by` equal to the caller's own user identifier.

Mobile and deployment. The files `utils/useForceLandscape.ts` and `utils/leafletRotationFix.ts` support a landscape layout on phones held upright and correct Leaflet's touch coordinates so that panning and zooming follow the finger. The application is deployed at <https://road-setback-1.onrender.com/> using the environment variables in Table 19.

A.3.6 Individual Testing and Evaluation Contribution

My testing work concentrated on the components I built. I diagnosed the permission failures that occurred despite existing policies and fixed them with the missing grants. I checked sign-up, sign-in, role changes and the visibility of the administrator panel on the deployed system, and I recorded that a changed role takes effect only after the user signs in again because the role is held in the signed token (Section 5.6). The results relating to accounts, stored inspections, candidate review, role management and field reports are reported in Section 6.8. The automated rule-engine tests exercise the screening logic developed by Lubasi Sipangule, and the System Usability Scale evaluation was carried out by Kupalwa Chibuye; I did not author either of them.

A.3.7 Collaboration with Other Group Members

My work depends on and is used by the work of both team members. The GIS service developed by Lubasi Sipangule reads the roads table and calls the spatial retrieval function in my database, and reads the road rule overrides that administrators store in my road_rule_assignments table. The dashboard and administrator pages designed by Kupalwa Chibuye save and read records through my tables under my security policies, and show or hide functions according to the role I defined. The three of us agreed the data contract between these parts, and defects at the boundaries, such as differences between the fields the dashboard reads and those the assessment engine returns (Table 23), had to be resolved jointly.

A.3.8 Contribution Summary

Work primarily completed by me:

- the database, its tables, constraints, triggers and migrations, and the loading of the Lusaka road and building data;
- authentication, input validation and session handling;
- roles, row-level security policies, grants and the set_user_role() function;
- deployment on Render and the environment configuration.

Work completed jointly with other group members:

- the administrator, review and reporting features, where I built the data structures and integration and Kupalwa Chibuye designed the interface;
- the mobile experience, where I adapted the layout and map handling to the interface designed by Kupalwa Chibuye;
- the agreed contracts between the interface, the GIS service and the database.

Work primarily completed by others but used in my work:

- the rule engine, setback configuration, geometry and assessment logic (Lubasi Sipangule);
- the interface design and the usability evaluation (Kupalwa Chibuye).

A.3.9 Estimated Contribution

Project Activity	Individual Contribution	Collaborative Contribution
Problem Analysis	30%	70%
Literature Review	25%	75%
Requirements Analysis	30%	70%

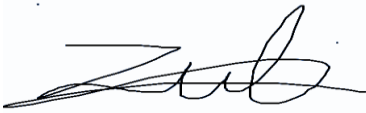
Project Activity	Individual Contribution	Collaborative Contribution
System/Research Design	35%	65%
Implementation / Development	35%	65%
Testing and Evaluation	30%	70%
Documentation	33%	67%
Other (deployment and configuration)	70%	20%

A.3.10 Statement of Authorship and Contribution

I confirm that the contribution described in this statement accurately represents the work that I personally undertook as part of the group project. I have identified collaborative work and have not knowingly claimed the work of another group member as my own.

Student Name: **Elijah Zulu**

Computer Number: **2021385299**

Signature:  _____

Date: 22nd October 2026

A.3.11 Supervisor Verification

I confirm that, to the best of my knowledge, the contribution described above is consistent with the student's participation in the group project.

Supervisor Name: Dr Lighton Phiri _____

Signature: _____

Date: 22nd October 2026

A.4 Individual Contribution Statement: Lubasi Sipangule

A.4.1 Individual Role and Responsibilities

No.	Area of Responsibility	Description
1	Rule engine and configuration	Configuration of road classes, reserve widths, building lines and sources; classification of roads; standard road controls.

No.	Area of Responsibility	Description
2	Assignments, overrides and precedence	Finding and applying road-wide assignments and subject-specific overrides, with deterministic precedence and ambiguity detection.
3	Distance measurement	Shortest distance between the road centreline and building points, lines and polygons, and creation of the setback zone.
4	Candidate assessment and screening workflow	Orchestration of a road screening, retrieval of candidates, and classification of each candidate with a reason and verification flag.
5	GIS service	Express API for analysis, road listing and health check, including input validation and loading of administrator overrides.
6	Testing of the logic	Automated rule-engine tests, performance timing and analysis of the defects found.

A.4.2 Detailed Individual Contribution

The work I personally undertook falls into the following activities.

- **Requirements analysis.** I translated the setback requirements of the roads legislation into the rule model of the system, including the distinction between the estimated road-reserve offset and the building-line distance (Section 3.3).
- **System and technical design.** I designed the rule-resolution procedure, the precedence scheme and the screening workflow, including the principle that unresolved or ambiguous information is reported for review rather than treated as compliant (Section 4.9).
- **Algorithm development.** I developed the precedence algorithm, the distance measurement for points, lines and polygons, and the classification of a candidate by zone.
- **Software development.** I implemented the rule engine, geometry module, detection modules, orchestrator and GIS service in JavaScript on Node.js and Express.
- **API and external-service integration.** I connected the GIS service to the Supabase database through its JavaScript client to load roads and administrator overrides and to call the spatial retrieval function.
- **Testing and debugging.** I wrote the automated rule-engine tests and the performance timing, analysed the two defects they exposed, and completed the fix for Leaflet's server-side rendering failure that Kupalwa Chibuye had started.
- **Documentation.** I documented the rule model, algorithms and screening workflow in the sections of this report that cover them.

A.4.3 Individual Deliverables

No.	Deliverable	Student's Contribution
1	Rule configuration file	Road classes, reserve widths and offsets, building-line status, sources and OSM class mapping.
2	Rule-engine modules	Classification, standard controls, assignment and override finding, precedence and application.
3	Geometry module	Distance and nearest-point calculation for points, lines and polygons; setback zone creation.
4	Detection modules and orchestrator	Candidate retrieval, candidate assessment and the road screening workflow.
5	GIS service	Express API with analysis, road listing and health endpoints.
6	Automated test suite and timing	31 rule-engine test cases and the performance measurement.

A.4.4 Contribution to the Main System or Research Output

The components of the final system that I developed or completed are the rule engine, the measurement and assessment logic, and the GIS service that exposes them. They are the core of the screening: they decide which distance applies to a road, measure how far each building is from it and classify the result. They depend on the data and security foundation developed by Elijah Zulu and are presented to officers through the interface developed by Kupalwa Chibuye. The administrator-managed overrides that my engine applies were stored by Elijah Zulu, and the dashboard and map that display my results were built by Kupalwa Chibuye. The database, authentication and interface are not my work.

A.4.5 Individual Implementation and Technical Work

Rules and classification. A road is classified from its OSM class, or from its route reference where one matches a configured pattern. The configuration holds the road-reserve width and estimated centreline offset for each class (for example 100 m and 50 m for a trunk road, 60 m and 30 m for a main road, 36 m and 18 m for district and branch roads, and 18 m and 9 m for other roads), the building-line distance, which is unresolved by default, and the statutory source. Pedestrian features such as footways and paths are excluded from screening, and an unknown class is left unresolved instead of being mapped to the most permissive class.

Precedence. Controls and overrides are ranked first by explicit priority and then by scope specificity, in the order road, road side, road segment, parcel and structure. When several controls share the highest rank the result is AMBIGUOUS and carries the list of conflicts, so

that an administrator decides. The control for a building is resolved in eight steps, described in Appendix G.3.

Measurement and assessment. The geometry module converts the road into line strings and returns zero when the road intersects the feature. Otherwise it measures the shortest distance using Turf.js, from the vertices of one geometry to the other in both directions, and returns the nearest point. The orchestrator uses the reserve offset X as the screening distance, or X plus the building-line distance Y when Y is resolved, retrieves the candidates within that distance, and assesses each one in its own try and catch block so that one malformed building yields an unknown result instead of failing the road. Each candidate is reported as lying within the estimated road reserve, within the building-line band, outside the screening zones, or outside X with Y unresolved, with a reason and a verification flag.

GIS service. The `/api/analyze` endpoint returns HTTP 400 when the road ID is not an integer, loads the road, fetches the administrator override for that road from the `road_rule_assignments` table, runs the screening and returns the result. The `/api/roads` endpoint lists road geometries for the map, and `/api/health` reports the status of the service.

A.4.6 Individual Testing and Evaluation Contribution

I wrote the 31 automated rule-engine test cases with Node.js's built-in test runner, covering classification, standard controls, road-wide assignments, precedence and subject-specific overrides. Twenty-nine passed and two exposed genuine defects, RO-06 and RA-07, which I analysed in Table 14. I also timed 1,000 consecutive candidate assessments, which gave about 0.33 ms per building, and checked the geometry module against synthetic roads and buildings placed at known distances. The system-level checks on the deployed application (ST-03 and ST-04) exercised the screening logic end to end. I completed the fix for Leaflet's server-side rendering failure that Kupalwa Chibuye had started. The usability evaluation with the System Usability Scale was carried out by Kupalwa Chibuye, and the checks of authentication, roles and persistence concern the work of Elijah Zulu.

A.4.7 Collaboration with Other Group Members

My work depends on and is used by the work of both team members. The GIS service reads the roads table and the road rule assignments, and calls the spatial retrieval function, in the database developed by Elijah Zulu, so the structure of those tables and the behaviour of the function determine what my logic can use. The dashboard and map developed by Kupalwa Chibuye consume the response of my service, so the fields it returns, such as required distance,

deviation and status, had to be agreed between us, and the differences between the fields the dashboard reads and those the engine returns (Table 23) had to be resolved jointly. Kupalwa Chibuye and I also worked together on loading the map in the browser only.

A.4.8 Contribution Summary

Work primarily completed by me:

- the rule configuration and the classification, precedence, assignment and override logic;
- the distance measurement and setback zone creation;
- candidate assessment, the screening orchestrator and the GIS service;
- the automated rule-engine tests and the performance timing.

Work completed jointly with other group members:

- the loading of the map in the browser only, which Kupalwa Chibuye started and I completed;
- the response contract between the GIS service and the dashboard, agreed with Kupalwa Chibuye;
- the use of administrator overrides and the spatial retrieval function, agreed with Elijah Zulu.

Work primarily completed by others but used in my work:

- the Supabase database, road and building data, authentication, access control and deployment (Elijah Zulu);
- the interface, map visualisation and usability evaluation (Kupalwa Chibuye).

A.4.9 Estimated Contribution

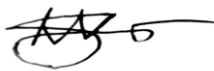
Project Activity	Individual Contribution	Collaborative Contribution
Problem Analysis	30%	70%
Literature Review	25%	75%
Requirements Analysis	35%	65%
System/Research Design	35%	65%
Implementation / Development	40%	60%
Testing and Evaluation	35%	65%
Documentation	33%	67%
Other (GIS service and API)	60%	40%

A.4.10 Statement of Authorship and Contribution

I confirm that the contribution described in this statement accurately represents the work that I personally undertook as part of the group project. I have identified collaborative work and have not knowingly claimed the work of another group member as my own.

Student Name: **Lubasi Sipangule**

Computer Number: **2022065786**



Signature: _____

Date: **22nd October 2026**_____

A.4.11 Supervisor Verification

I confirm that, to the best of my knowledge, the contribution described above is consistent with the student's participation in the group project.

Supervisor Name: _____

Signature: _____

Date: _____

A.5 Individual Contribution Statement: Kupalwa Chibuye**A.5.1 Individual Role and Responsibilities**

No.	Area of Responsibility	Description
1	Interface design	Layout, navigation, colour scheme and colour coding of results for officers without GIS training.
2	Login and sign-up pages	Design and implementation of the pages, including loading, error and success feedback.
3	Dashboard	Road input, road control details, summary counters, results table, inspection history, and the client code that calls the GIS service and saves results.
4	Usability evaluation	System Usability Scale questionnaire, collection of responses, scoring and analysis.

A.5.2 Detailed Individual Contribution

The work I personally undertook falls into the following activities.

- **System and interface design.** I designed the screens an officer uses to sign in and to screen a road, choosing a simple map-plus-panel layout so that a road can be screened in a few actions (Section 4.8).
- **User interface design.** I defined the colour scheme and the colour coding of results, with red for potential issues, green for compliant and amber for buildings needing review.
- **Software development.** I implemented the login, sign-up and dashboard pages as React and TypeScript components in Next.js.
- **Testing and debugging.** I began the fix for the failure of Leaflet during server-side rendering by loading the map only in the browser, which Lubasi Sipangule then took over and completed.
- **Usability evaluation.** I prepared the System Usability Scale questionnaire and instructions, collected seventeen responses, scored them and analysed the comments (Section 6.5 and Appendix E).
- **Documentation.** I documented the interface design and the usability evaluation in the sections of this report that cover them.

A.5.3 Individual Deliverables

No.	Deliverable	Student's Contribution
1	Login and sign-up pages	Designed and built the pages and their feedback messages.
2	Dashboard page	Road input, details panel, counters, results table, history list and the code that calls the GIS service and saves results.
3	Usability questionnaire and analysis	SUS questionnaire, instructions, response analysis and score.

A.5.4 Contribution to the Main System or Research Output

The components of the final system that I developed or completed are the login and sign-up pages and the dashboard, which together form the part of the interface an officer uses to sign in, screen a road, read the results and review earlier runs. The interface depends on, and presents the output of, the GIS service developed by Lubasi Sipangule and the database, access control and deployment developed by Elijah Zulu. The sign-in flow was built jointly: I designed the pages and Elijah Zulu developed the authentication logic behind them. The screening logic and the database are not my work.

A.5.5 Individual Implementation and Technical Work

Front-end structure. The interface is a Next.js application written in TypeScript with React. The login, sign-up and dashboard pages are client components that keep their state in React state hooks. The dashboard holds the road input, the screening result, the selected road, the loading state and the inspection history, and it sends the road ID to the GIS service through the Next.js route `/api/analyze`. When a result returns, the dashboard converts it into GeoJSON features for display, maps the engine's "non_compliant" status to the user-facing label, and saves the inspection and one row per candidate under the signed-in user's session.

Results display. The side panel shows the road control details, the summary counters and a results table in which each candidate carries a colour-coded status badge of red, green or amber, with a toggle to show all candidates. The inspection history lists earlier runs, and the navigation bar shows the signed-in user and the links appropriate to the user's role.

Usability evaluation. The evaluation used the 10-item System Usability Scale delivered as an online questionnaire with step-by-step instructions. The responses were summarised by item, scored with the standard SUS rule to give 68.5, and the free-text comments were analysed (Section 6.5).

A.5.6 Individual Testing and Evaluation Contribution

My testing and evaluation work concentrated on the usability of the interface. I planned and carried out the usability evaluation with the System Usability Scale and analysed its results, including its limitations: a small sample, no record of respondents' backgrounds, and probably few or no enforcement officers (Section 6.5). The interface issues identified in the code review, namely that the dashboard reads fields the assessment engine does not return, that raw database errors reach the user, and that the label "violation" reads as a legal determination, are reported in Section 6.8. I also began the fix for Leaflet's server-side rendering failure, which Lubasi Sipangule later took over and completed. The automated rule-engine tests exercise the screening logic developed by Lubasi Sipangule, and I did not author them.

A.5.7 Collaboration with Other Group Members

My work depends on and is used by the work of both team members. The dashboard can only show what the GIS service developed by Lubasi Sipangule returns, so the fields of its response, such as required distance, deviation and status, had to be agreed between us, and differences between the fields the dashboard reads and those the engine returns (Table 23) had to be resolved jointly. The dashboard reads and writes the tables created by Elijah Zulu under the

security policies he defined, and the sign-in pages I designed rely on the authentication logic he developed. Elijah Zulu also deployed the application on which the usability evaluation was run.

A.5.8 Contribution Summary

Work primarily completed by me:

- the interface design, colour scheme and colour coding of results;
- the login, sign-up and dashboard pages;
- the System Usability Scale evaluation and its analysis.

Work completed jointly with other group members:

- the sign-in flow, where I designed the pages and Elijah Zulu built the authentication logic;
- the client-side loading of the map to avoid Leaflet's server-side rendering failure, which I started and Lubasi Sipangule completed;
- the agreed contracts between the interface, the GIS service and the database.

Work primarily completed by others but used in my work:

- the rule engine, setback configuration, geometry and assessment logic (Lubasi Sipangule);
- the database, authentication, access control and deployment (Elijah Zulu).

A.5.9 Estimated Contribution

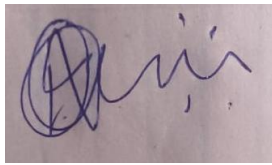
Project Activity	Individual Contribution	Collaborative Contribution
Problem Analysis	30%	70%
Literature Review	25%	75%
Requirements Analysis	30%	70%
System/Research Design	25%	75%
Implementation / Development	20%	80%
Testing and Evaluation	40%	60%
Documentation	33%	67%
Other (usability survey)	70%	30%

A.5.10 Statement of Authorship and Contribution

I confirm that the contribution described in this statement accurately represents the work that I personally undertook as part of the group project. I have identified collaborative work and have not knowingly claimed the work of another group member as my own.

Student Name: **Kupalwa Chibuye**

Computer Number: **2021424278**



Signature: _____

Date: **25th October 2026**

A.5.11 Supervisor Verification

I confirm that, to the best of my knowledge, the contribution described above is consistent with the student's participation in the group project.

Supervisor Name: _____

Signature: _____

Date: _____

Appendix B: User Manual / Installation Guide

B.1 Using the System

The system is available at <https://road-setback-1.onrender.com/>. It runs in any modern web browser and requires no plug-ins.

- **Sign up and log in.** Create an account with an e-mail address and password on the sign-up page, confirm the e-mail address if requested, and log in. Accounts are ordinary users by default; only an administrator can grant the admin role.
- **Select a road.** On the dashboard, click a grey road on the map or type its road ID in the "New Road Inspection" box.
- **Run the screening.** Press Run. The system resolves the road's rule, retrieves buildings near the road, measures each one and shows the result.
- **Read the result.** The selected road is drawn in dark blue and its setback zone as an orange band. Buildings are coloured red (potential issue), green (compliant) or amber (needs review). The "Road Control Details" panel shows the road, regulatory class, configured reserve width, screening distance, rule source and whether verification is required. Counters above the results table show the number of candidates, potential issues, compliant and review cases.
- **Inspect a building.** Click a building on the map to see its measured distance, the required distance, the deviation and the reason for the status.
- **Check the history.** The Inspection History panel lists your recent runs with the area, number of buildings and number of potential issues.
- **Report a violation.** Choose "Report Violation" in the navigation bar to log an observation made in the field without running an analysis. Select the category, report type and requested action, describe what was observed, optionally add the road and candidate IDs, capture your location and attach a photo, then submit.
- **Administer (administrators only).** The Admin Panel lists all users with a role selector, lists recent building candidates with a status selector and an audit-trail viewer, and manages road-specific setback overrides and submitted manual reports. Role changes take effect when the affected user next signs in.

Important: every result is a screening indication. Results marked "Verification required" must be confirmed by field inspection before any enforcement action.

B.2 Installing and Running the System Locally

Prerequisites. Node.js and npm, Git, and a Supabase project with the PostGIS extension enabled.

- Clone the repository (Lubasi-unza/Road-Setback). It contains two independent Node.js applications: frontend/ (Next.js) and the GIS service directory (named Backend in the repository README), plus supabase/migrations/.
- Install dependencies separately in each application with npm install.
- Create the database objects. Load the roads and buildings layers (OSM extract, columns fid, name, fclass, geometry) and create the find_buildings_near_road(road_id_input, distance_meters) function, then run the four migration files in order in the Supabase SQL editor: 20260908_admin_roles_and_candidates.sql, 20260909_grants_and_profiles.sql, 20260910_admin_set_user_role.sql and 20261003_feature_additions.sql. The roads and buildings tables, the spatial function, the manual_reports report columns and the manual-report-photos storage bucket are not created by these migrations and must be set up separately.
- Create the environment files listed in Table 19. They must never be committed to version control.
- Start the GIS service with npm start (port 5000) and the frontend with npm run dev (port 3000), in two terminals, then open <http://localhost:3000>.
- Verify the GIS service with: `curl http://localhost:5000/api/health`. A healthy service returns `{"status":"ok", ...}`.

Table 19: Environment variables

Application	Variable	Purpose
GIS service	SUPABASE_URL	URL of the Supabase project
GIS service	SUPABASE_SERVICE_ROLE_KEY	Service-role key used to read roads and call the spatial function (secret)
GIS service	PORT (optional)	Listening port; default 5000
Frontend	NEXT_PUBLIC_SUPABASE_URL	URL of the Supabase project
Frontend	NEXT_PUBLIC_SUPABASE_PUBLIC_KEY	Public (publishable) key for browser and server sessions
Frontend	NEXT_PUBLIC_GIS_API_URL	Address of the GIS service; default http://localhost:5000

To run the frontend in production mode, run `npm run build` followed by `npm start`. The `npm start` command fails with "Could not find a production build" if the build has not been run.

B.3 Troubleshooting

- **"Failed to connect to GIS server".** The GIS service is not running or `NEXT_PUBLIC_GIS_API_URL` is wrong. Check the health endpoint.
- **"Failed to retrieve road: Cannot coerce the result..." (HTTP 500).** The road ID does not exist or none was entered. Enter a valid road ID or click a road on the map.
- **"Permission denied" when saving results.** The table grants and row-level security policies from the migrations have not been applied.
- **A new administrator still sees the user view.** Roles are stored in the sign-in token. Sign out and sign in again.

Appendix C: Additional System Designs / Diagrams

C.1 Screening Statuses

For each candidate the engine compares the measured distance D from the road centreline with the estimated road-reserve centreline offset X and, where resolved, the additional building-line distance Y . Table 20 lists the statuses returned by the GIS service.

Table 20: Screening statuses returned by the GIS service

Status	Condition	Meaning
ROAD_RESERVE_CANDIDATE	$D \leq X$	Candidate lies within the estimated road reserve (shown as a potential issue)
BUILDING_LINE_CANDIDATE	$X < D \leq X + Y$ (Y resolved)	Candidate lies outside the reserve but inside the building-line band
OUTSIDE_BUILDING_LINE_ZONE	$D > X + Y$	Candidate lies outside both zones; not a candidate
OUTSIDE_ESTIMATED_ROAD_RESERVE	$D > X$, Y unresolved	Outside the reserve but the building-line distance is not configured; needs review
UNRESOLVED	X is null	The reserve offset could not be resolved (for example an excluded or unknown road class)
ADMIN_REVIEW_REQUIRED / ambiguous	Equal-precedence controls	Several applicable controls tie; an administrator must resolve the conflict
unknown	Error during assessment of one candidate	Candidate could not be measured; the reason is recorded and the rest of the road is still assessed

C.2 GIS Service API Specification

POST /api/analyze screens one road. Request body (JSON): `{"road_id": <integer>}`. A missing or non-integer road_id returns HTTP 400 with `{"error": "A valid road_id is required"}`. Any other failure, including an unknown road ID, returns HTTP 500 with `{"error": <message>}`. A successful response contains:

road rule	GeoJSON feature of the selected road effective road control: regulatoryClass, roadReserve, buildingLine, source, classificationStatus,
--------------	--

```

        verificationRequired, resolutionStatus
screeningDistanceMeters  X, or X + Y when Y is resolved
roadReserve              { estimatedCentrelineOffsetMeters,
                          assessment }
buildingLine             { status, additionalDistanceMeters,
                          assessment }
candidateCount, candidates[]
  candidateId, candidateGeometry, candidateProperties,
  measuredDistanceMeters, rule, roadReserve, buildingLine,
  screeningStatus, featureStatus, reason, requiresVerification,
  screening { included, zone, distanceMeters, thresholds },
  nearestPointOnRoad
screeningStatus          "provisional" | "ambiguous" | "unresolved" (road level)
requiresVerification     true unless every control is resolved

```

GET /api/roads returns a GeoJSON FeatureCollection of all roads (properties fid, name, fclass) for the map. **GET /api/health** returns {"status":"ok","timestamp":...}. A diagnostic endpoint GET /api/db-test exists in the current code and should be removed before operational use.

C.3 Additional Database Tables

Table 8 describes the tables of the first design. The final migration (20261003_feature_additions.sql) added three tables, summarised in Table 21.

Table 21: Database tables added in the final migration

Table	Purpose	Key columns and access control
road_rule_assignments	Administrator-managed setback override for one road	road_id (unique), type (uniform or side_specific), setback_distance_m, left_setback_m, right_setback_m, reference, reason, priority. All authenticated users may read; only administrators may insert, update or delete.
candidate_status_history	Audit trail of changes to a candidate's status	candidate_id, previous_status, new_status, changed_by, note, changed_at. Written only by a SECURITY DEFINER trigger; read-only for authenticated users.
manual_reports	Field reports logged without running an analysis	road_id, road_name, description, latitude, longitude, photo_url, status (open, reviewed, resolved, dismissed), reported_by. The report form also writes category, type, requested action and road-width columns that are not yet in the migration.

The status of each candidate can be changed only by administrators; every change inserts a row in `candidate_status_history` through the trigger `log_candidate_status_change()`, so the history cannot be written or altered by the application directly.

Appendix D: Additional Test Cases and Results

D.1 Rule-Engine Test Cases

The tests below are in backend/test/ruleEngine.test.js and are run with: `node --test test/ruleEngine.test.js` (Node.js v22.22.2). The run reported 31 tests, 29 passed, 2 failed, 0 skipped, total duration about 133 ms.

Table 22: Complete list of rule-engine test cases and results

ID	Test case	Result
RC-01	trunk -> trunk_road (provisional)	Pass
RC-02	primary -> main_road	Pass
RC-03	secondary -> district_road	Pass
RC-04	tertiary -> branch_road	Pass
RC-05	residential/service/unclassified/primary_link use fallback other_road	Pass
RC-06	footway/path/track/steps/cycleway are excluded	Pass
RC-07	unknown OSM class is unresolved, not silently other_road	Pass
RC-08	trunk route reference T2 is detected; A1 is not	Pass
RC-09	missing fclass/highway does not throw and is unresolved	Pass
SC-01	configured reserve widths and centreline offsets	Pass
SC-02	building line is unresolved by default for every class	Pass
SC-03	unknown class yields null offset and unresolved status	Pass
SC-04	null class (excluded road) yields null offset	Pass
RA-01	no assignment -> class default, status NONE	Pass
RA-02	uniform assignment replaces the default offset	Pass
RA-03	assignment with building-line distance resolves Y	Pass
RA-04	assignment for another road is ignored	Pass
RA-05	two equal-priority assignments for one road are AMBIGUOUS	Pass
RA-06	higher-priority assignment wins	Pass
RA-07	side-specific assignment is unsupported: offset stays unresolved (null)	Fail
RP-01	empty list -> NONE	Pass
RP-02	structure outranks parcel at equal priority	Pass

ID	Test case	Result
RP-03	explicit priority outranks scope	Pass
RP-04	equal priority and scope -> AMBIGUOUS with both conflicts	Pass
RO-01	building override matches by structure id only	Pass
RO-02	parcel override matches by parcel id	Pass
RO-03	applies_to restricts an override by feature type	Pass
RO-04	building override replaces the offset for that building only	Pass
RO-05	two equally specific overrides for one building are AMBIGUOUS	Pass
RO-06	an override with scope "structure" is applied (scope is accepted by the finder and precedence table)	Fail
RO-07	missing road geometry/feature is rejected	Pass

D.2 Code Review Findings

Table 23: Defects and risks identified by code review

No.	Finding	Severity	Recommended action
1	GIS endpoint POST /api/analyze and the Next.js proxy route perform no authentication, and the GIS service uses the service-role key with open CORS. The Next.js middleware refreshes sessions but does not block unauthenticated requests, so page protection is client-side only.	High	Verify the Supabase session in the proxy and pass a signed token to the GIS service; restrict CORS to the frontend origin.
2	Diagnostic endpoint /api/db-test returns sample road and building data for a hard-coded road.	Medium	Remove before deployment.
3	Side-specific road assignments silently use the class default (test RA-07).	Medium	Return an unresolved status until directional analysis exists.
4	Overrides with scope "structure" throw an error (test RO-06).	Medium	Add "structure" to the supported scopes.
5	Dashboard reads fields (requiredSetbackMeters, deviationMeters, status "non_compliant") that the current assessment engine does not return, so deviation and required distance may be blank and statuses may not map.	Medium	Align the response contract and add a contract test.
6	The function find_buildings_near_road(), the roads and buildings tables, and the report-form columns and storage bucket are not in the versioned migrations.	Medium	Add them to the migrations so the database can be recreated.
7	An unknown road ID returns a raw database message (HTTP 500: "Cannot coerce the result to a single JSON object").	Medium	Return HTTP 404 with a clear message and validate the input in the interface.

No.	Finding	Severity	Recommended action
8	Row-level policies allow any authenticated user to insert any inspection or candidate; the administrator e-mail is hard-coded in a migration.	Low	Constrain inserts to the signed-in user and move administrator seeding out of the migrations.
9	Obsolete src/index.js uses CommonJS require() in an ES-module package and refers to a module that no longer exists.	Low	Delete or rewrite it.
10	test/resolveRoadRule.test.js is JSON data (and not valid JSON), not an executable test; test-assess scripts need a live database.	Low	Replace with automated tests such as ruleEngine.test.js.
11	JSON body limit of 50 MB on the GIS service is far above what the API needs.	Low	Reduce to a few kilobytes.
12	Documentation and dependency versions differ (README names Next.js 14, package.json requires 16; Turf 6.5 and 7.x are mixed; the README names the backend folder Backend).	Low	Standardise versions and update the README.
13	Automated results are labelled "violation" in the interface and database, which reads as a legal determination.	Medium	Use "potential issue" for screening results.

Appendix E: Data Collection Instruments

E.1 System Usability Scale Questionnaire

BUILDING SETBACK ENFORCEMENT

About the System

The **Road Setback Management System** is a web-based tool designed to help Lusaka City Council enforcement officers screen buildings along a road for possible road setback violations. It does not replace field inspection – it helps officers decide **where to inspect first**.

Instructions to the participant:

For each of the following statements, please tick **one** box to indicate how much you agree or disagree, based on your experience of using the Road Setback Management System today. Please respond to every statement. There are no right or wrong answers – we are interested in your honest opinion.

Click on the link to access the system ----- <https://road-setback-1.onrender.com/>

How You Use It (in simple steps)

Step Action

STEP 1 Log in with your official email and password.

STEP 2 On the dashboard, **select a grey road** by clicking it on the map or typing its road ID in the side panel.

STEP 3 Click **Run**. The system analyses the buildings near that road.

STEP 4 View the results – the selected road is highlighted, the setback zone is shown as an orange band, and buildings are coloured red (violation), green (compliant), or amber (needs review).

STEP 5 Click a **building** to see its measured distance, required setback and deviation.

STEP 6 Check the **Inspection History** to see previous screenings.

Scale: 1 = Strongly Disagree · 2 = Disagree · 3 = Neutral · 4 = Agree · 5 = Strongly Agree

* Indicates required question

1. 1. I think that I would like to use this system frequently. *

Tick all that apply.

- ☐ Strongly Disagree
- ☐ Disagree
- ☐ Neutral
- ☐ Agree
- ☐ Strongly Agree

Figure 10: SUS Questionnaire

2. 2. I found the system unnecessarily complex. *

Tick all that apply.

- ☐ Strongly Disagree
☐ Disagree
☐ Neutral
☐ Agree
☐ Strongly Agree

3. 3. I thought the system was easy to use. *

Tick all that apply.

- ☐ Strongly Disagree
☐ Disagree
☐ Neutral
☐ Agree
☐ Strongly Agree

4. 4. I think that I would need the support of a technical person to be able to use this system. *

Tick all that apply.

- ☐ Strongly Disagree
☐ Disagree
☐ Neutral
☐ Agree
☐ Strongly Agree

5. 5. I found the various functions in this system were well integrated. *

Tick all that apply.

- ☐ Strongly Disagree
☐ Disagree
☐ Neutral
☐ Agree
☐ Strongly Agree

Figure 11: SUS Questions 1

6. 6. I thought there was too much inconsistency in this system. *

Tick all that apply.

- ☐ Strongly Disagree
☐ Disagree
☐ Neutral
☐ Agree
☐ Strongly Agree

7. 7. I would imagine that most people would learn to use this system very quickly. *

Tick all that apply.

- ☐ Strongly Disagree
☐ Disagree
☐ Neutral
☐ Agree
☐ Strongly Agree

8. 8. I found the system very cumbersome to use. *

Tick all that apply.

- ☐ Strongly Disagree
☐ Disagree
☐ Neutral
☐ Agree
☐ Strongly Agree

9. 9. I felt very confident using the system. *

Tick all that apply.

- ☐ Strongly Disagree
☐ Disagree
☐ Neutral
☐ Agree
☐ Strongly Agree

Figure 12: SUS Questions 2

10. 10. I needed to learn a lot of things before I could get going with this system. *

Tick all that apply.

- ☐ Strongly Disagree
☐ Disagree
☐ Neutral
☐ Agree
☐ Strongly Agree

11. Leave a comment about the system, anything to be added, removed or improved?

This content is neither created nor endorsed by Google.

Google Forms

Figure 13: SUS Questions 3

E.2 System Usability Scale Results

The questionnaire was completed online by 17 respondents. The charts below are the response summaries exported from the survey form; Table 16 and Figure 9 in Chapter 6 reproduce the item counts and the computed SUS score of 68.5. The four free-text comments are summarised in Section 6.5.

BUILDING SETBACK ENFORCEMENT

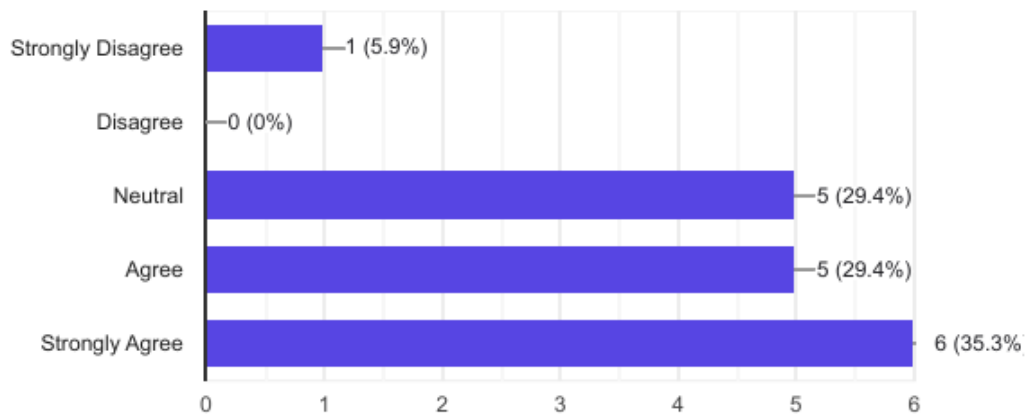
17 responses

[Publish analytics](#)

1. I think that I would like to use this system frequently.

[Copy chart](#)

17 responses



2. I found the system unnecessarily complex

[Copy chart](#)

17 responses

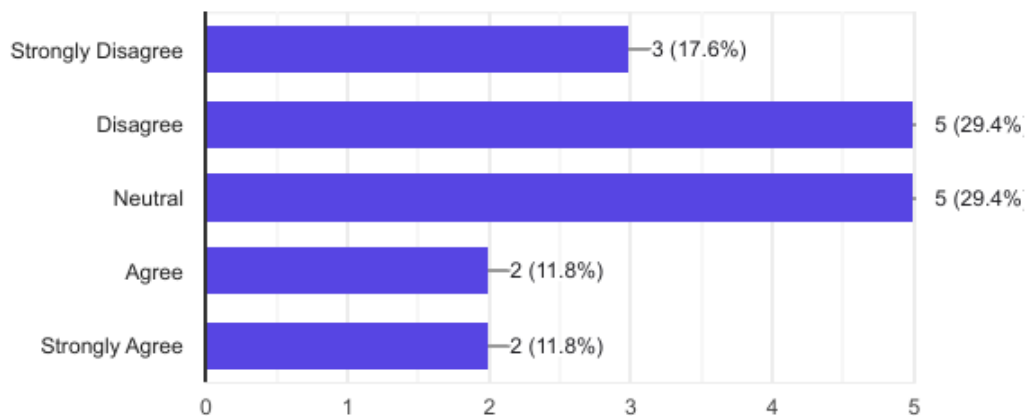


Figure 14: SUS Responses 1

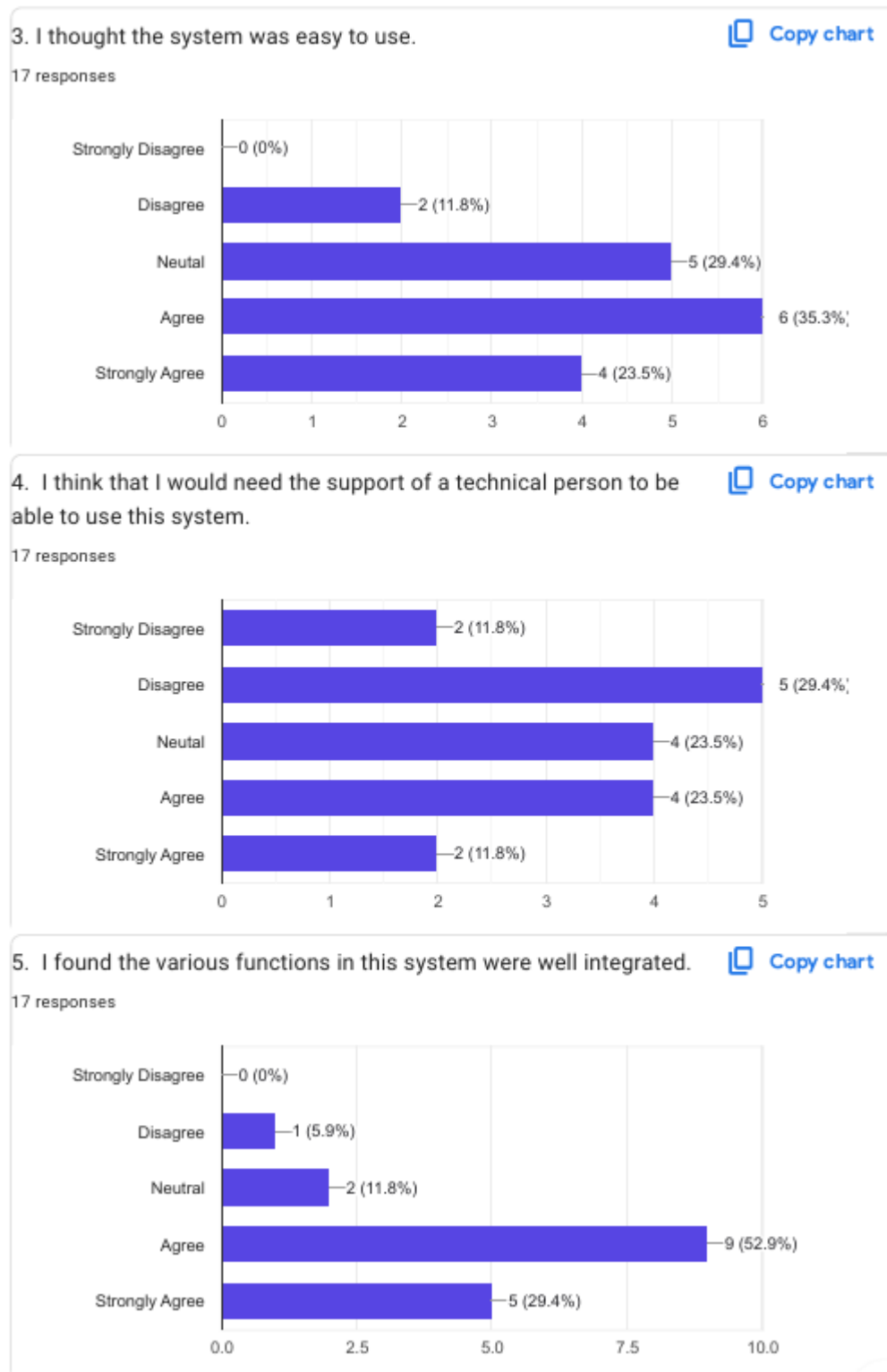
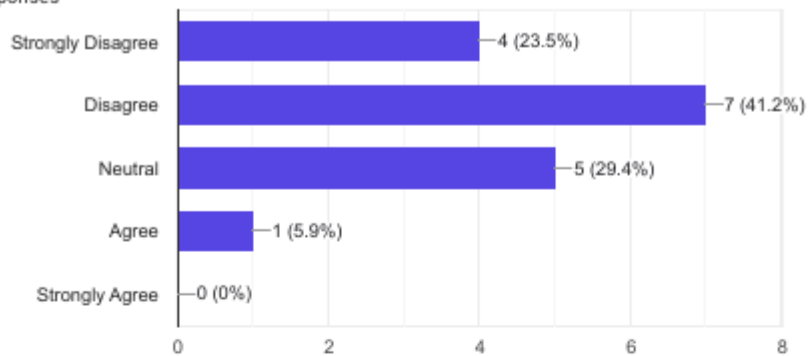


Figure 15: SUS Responses 2

6. I thought there was too much inconsistency in this system.

[Copy chart](#)

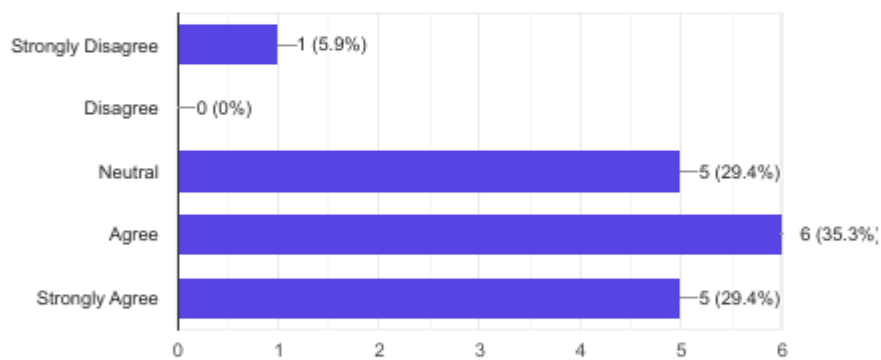
17 responses



7. I would imagine that most people would learn to use this system very quickly.

[Copy chart](#)

17 responses



8. I found the system very cumbersome to use.

[Copy chart](#)

17 responses

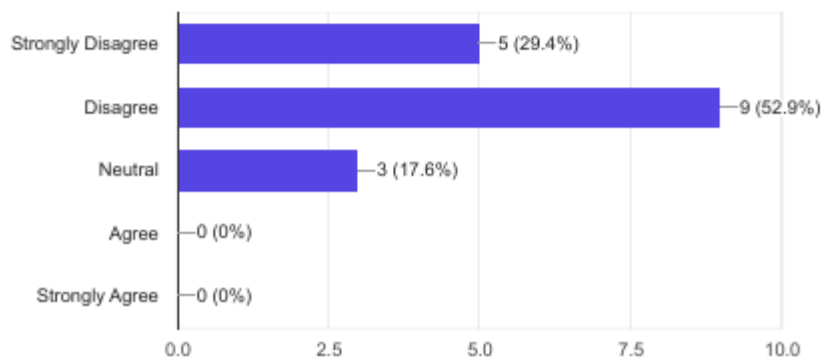


Figure 16: SUS Responses 3

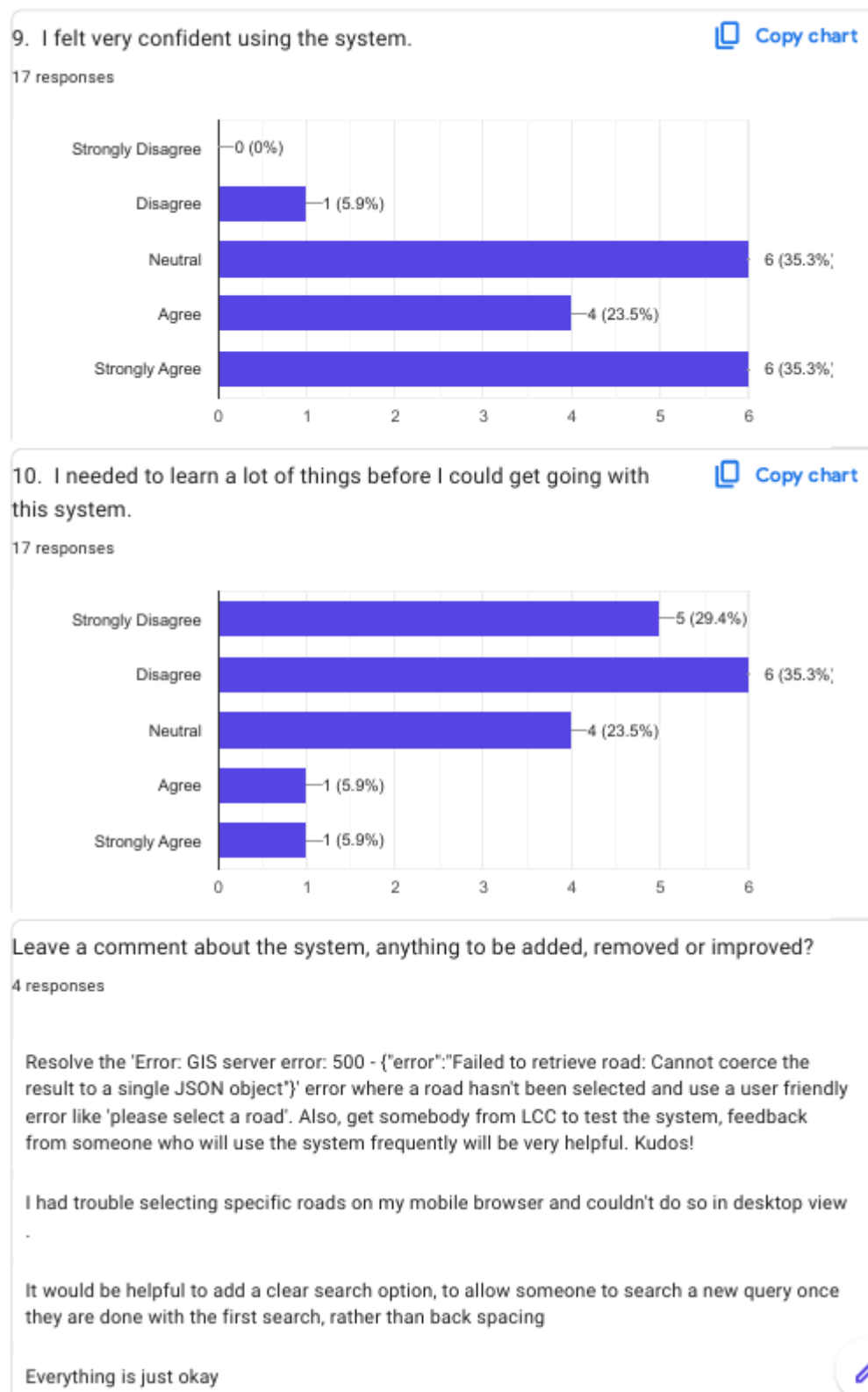


Figure 17: SUS Responses 4

Appendix F: Additional Code / Configuration

F.1 Standard Road Controls (config/setback-rules.json, excerpt)

```
"standard_road_rules": {
  "trunk_road":    reserve width 100 m, centreline offset 50 m
  "main_road":     reserve width 60 m, centreline offset 30 m
  "district_road": reserve width 36 m, centreline offset 18 m
  "branch_road":   reserve width 36 m, centreline offset 18 m
  "urban_road":    reserve width 36 m, centreline offset 18 m
  "park_road":     reserve width 18 m, centreline offset 9 m
  "other_road":    reserve width 18 m, centreline offset 9 m
  // every class: "buildingLine": { "status": "unresolved",
  //                               "additionalDistanceMeters": null },
  // "source": { "name": "Public Roads Act 2002", "section": "21" },
  // "localAuthorityApplicability": "requires_verification"
},
"regulatory_classification": {
  "osm_fclass": { "trunk": "trunk_road", "primary": "main_road",
    "secondary": "district_road", "tertiary": "branch_road" },
  "fallback": { "regulatoryClass": "other_road",
    "status": "provisional" },
  "screening_eligibility": {
    "fallback_screenable": ["primary_link", "secondary_link",
      "tertiary_link", "residential", "service", "unclassified"],
    "excluded": ["path", "footway", "track", "steps", "cycleway"] }
}
```

F.2 Zone Assessment of a Candidate (src/detection/assessCandidate.js, simplified)

```
if (control.resolutionStatus === "AMBIGUOUS") -> ADMIN_REVIEW_REQUIRED
if (X == null)                                -> UNRESOLVED
if (distance <= X)                            -> ROAD_RESERVE_CANDIDATE
if (Y != null && distance <= X + Y)           -> BUILDING_LINE_CANDIDATE
if (Y != null)                                -> OUTSIDE_BUILDING_LINE_ZONE
else                                           -> OUTSIDE_ESTIMATED_ROAD_RESERVE
// every result also carries requiresVerification: true
```

F.3 Database Policies and Functions (excerpts)

```
-- Administrators manage road-specific setback overrides
create policy "admins can insert road overrides"
  on public.road_rule_assignments for insert to authenticated
  with check ((auth.jwt() -> 'app_metadata' ->> 'role') = 'admin');

-- Role changes go through a SECURITY DEFINER function
create or replace function public.set_user_role(target_user_id uuid, new_role text)
returns void language plpgsql security definer set search_path = public as $$
```

```
begin
  if (auth.jwt() -> 'app_metadata' ->> 'role') is distinct from 'admin' then
    raise exception 'Only admins can change user roles'; end if;
  if new_role not in ('admin','user') then raise exception 'Invalid role: %', new_role; end if;
  update auth.users set raw_app_meta_data =
    coalesce(raw_app_meta_data, '{} '::jsonb) || jsonb_build_object('role', new_role)
    where id = target_user_id;
  update public.profiles set role = new_role where id = target_user_id;
end; $$;

-- Audit trail written by trigger on every status change
create trigger building_candidates_log_status_change
  after update on public.building_candidates
  for each row execute function public.log_candidate_status_change();
```

Appendix G: Other Supporting Material

G.1 Deployment and Repository

- Live application: <https://road-setback-1.onrender.com/> (hosted on Render).
- Database and authentication: Supabase (hosted PostgreSQL with PostGIS).
- Source code: GitHub repository [Lubasi-unza/Road-Setback](#).

G.2 Data Sources and Licensing

Road centrelines and building footprints are OpenStreetMap data (Geofabrik-style extract with fields fid, name and fclass) used under the Open Database License, with attribution displayed on the map. Map tiles are OpenStreetMap tiles. All software libraries are open source.

G.3 Rule Resolution Order

The engine resolves the control for a building in eight steps: (1) classify the road from its OSM class or route reference; (2) look up the standard road control for that class; (3) find road-wide assignments for the road; (4) resolve their precedence (explicit priority, then scope); (5) apply the selected assignment; (6) find subject-specific overrides for the building, parcel or road segment; (7) resolve their precedence; (8) apply the selected override. A tie at step 4 or step 7 yields an ambiguous result that is passed to an administrator rather than resolved arbitrarily.