IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

A Web-Based Store Locator and Routing System Using GeoDjango and Spatial Query Optimization

Gaikwad Baleeram Pandurang [1]

[1]Department of Electronics and Telecommunication Engineering,
Aditya Engineering College, Beed,
Aditya Education Trust, affiliated to Dr. Babasaheb Ambedkar Technological University, Lonere, Maharashtra, India
Department of Electronics and Telecommunication Engineering, Aditya Engineering College, Beed

Prof. Jogdand V.L.[2]
(Guide)
[1]Department of Electronics and Telecommunication Engineering,
Aditya Engineering College, Beed,
Aditya Education Trust, affiliated to Dr. Babasaheb Ambedkar Technological University, Lonere, Maharashtra, India
Department of Electronics and Telecommunication Engineering, Aditya Engineering College, Beed

Abstract

Locating nearby commercial establishments and reaching them by the shortest practical path remains a recurring difficulty for users of general-purpose web mapping tools, which are designed for broad search rather than business-specific discovery. This paper presents a web-based store locator and routing system built on the Django web framework, the GeoDjango spatial extension, and a PostGIS-enabled relational database, integrating location-aware store search and route generation into a single interface. Users can search stores by name, category, or proximity, view matching results on an interactive map, and obtain a distance-optimized route computed using the Haversine great-circle distance formula. An administrative module supports centralized management of store records, including category, address, geographic coordinates, and operating hours. The system was evaluated on datasets of 100 to 5000 store records and compared against a conventional, non-spatially-indexed search implementation. The proposed system achieved higher search accuracy (92–95% versus 82–88% for the baseline) and lower response times across all tested dataset sizes, indicating that spatial indexing scales more favourably than flat relational search as the number of records grows. The architecture offers a practical, low-cost reference design for retail, healthcare, banking, and tourism applications that require integrated place discovery and navigation.

Keywords: Store Locator; Route Optimization; Geographic Information System (GIS); GeoDjango; Location-Based Services; Spatial Database; Haversine Formula; Web-GIS

1. Introduction

The proliferation of internet-connected mobile devices, together with maturing Geographic Information System (GIS) and Global Positioning System (GPS) technologies, has made location-based services (LBS) a routine part of daily life [7], [31]. Users increasingly expect near-instant answers to questions such as where the nearest pharmacy, supermarket, or bank branch is located, and how to get there by the shortest available path. General-purpose mapping platforms answer such queries reasonably well, but they are not tailored to the needs of individual retailers or service networks, which must maintain accurate, frequently updated records of their own outlets, categories, and operating hours [9], [10], [14].

Traditional approaches to this problem — manual inquiries, printed directories, or switching between a separate search application and a separate navigation application — are time-consuming and prone to stale information. A more effective approach couples a spatial database of business records directly with route-generation logic inside one web application, so that discovery and navigation share the same data model and the same user session [1], [4], [27].

This paper describes such a system: a Web-Based Store Locator and Routing System developed using Python, the Django web framework, the GeoDjango spatial toolkit, and a PostGIS/MySQL-backed database, combined with OpenStreetMap and Google Maps routing services. The specific contributions of this work are: (i) an integrated architecture that unifies store search, interactive map visualization, and shortest-route generation within a single platform; (ii) a spatial-query design that uses the Haversine formula for proximity ranking and a weighted objective function for route selection; (iii) a centralized administrative module for store-record management; and (iv) an empirical evaluation of search accuracy, route-generation time, and response time across growing dataset sizes, benchmarked against a conventional non-spatial implementation.

The remainder of the paper is organized as follows. Section 2 reviews related work on GIS-based location services, routing algorithms, and spatial database frameworks. Section 3 presents the proposed system architecture and mathematical model. Section 4 describes the implementation. Section 5 reports the experimental results and discussion. Section 6 concludes the paper and outlines future work.

2. Related Work

Early work on web-based GIS established the foundational architecture for delivering spatial data and analysis over the internet, distinguishing distributed geographic information services from desktop GIS [7] and describing the geoportal pattern for publishing and discovering geospatial resources online [8]. Li et al. [18] later confirmed that combining GIS-based spatial data processing with GPS positioning improves location-tracking accuracy, although their system, like several reviewed here, did not extend to real-time route optimization. Multi-scale interactive GIS representations, which allow a single spatial dataset to be queried at varying levels of geometric detail, were formalized by Van Oosterom and Schenkelaars [13] and remain influential in modern web-mapping engines. Building on these foundations, open-source web-GIS stacks have been proposed for prototyping location services without proprietary licensing costs [12], and more recent work has combined containerized deployment with graph-database backends to improve the scalability of web-GIS applications under concurrent load [17].

A second stream of research addresses route computation and navigation. Ioannou et al. [11] proposed a GIS-based decision-support system for intra-city vehicle routing with time windows, while Amiri [16] described a geospatial service framework specifically for web-based route delivery. Abousaeidi et al. [15] applied GIS-based route optimization to last-mile delivery, reporting measurable reductions in travel distance. At the algorithmic level, Wang and Chen [19] combined Dijkstra's algorithm with GIS mapping to reduce travel distance and time, though their system lacked business-specific search; Kumar et al. [20] paired GPS positioning with shortest-path routing to improve navigation accuracy, at the cost of increased power consumption on mobile clients. More recently, machine-learning techniques have been introduced to

improve route selection under variable conditions: Zhang and Liu [23] used traffic-prediction models to refine route choice, and Kim et al. [25] proposed adaptive, real-time routing that depends on continuous server connectivity — a dependency the present system avoids by keeping shortest-path computation lightweight and largely client-independent.

A third and directly relevant stream concerns spatial frameworks for business and retail location. Suárez-Vega et al. [9] and Okabe and Okunuki [10] modelled retail-site selection and street-network demand estimation using GIS, establishing that proximity-weighted spatial queries meaningfully improve site- and store-discovery outcomes. Albuquerque et al. [14] extended web-GIS to tourism marketing and spatial decision support, and Behzadi [5] proposed a commercial location-based system for customer-facing store discovery; a follow-up study by Behzadi et al. [4] combined web-GIS with metaheuristic optimization for store allocation and routing jointly, which is conceptually the closest prior work to the system proposed here. Singh and Joshi [24], together with Ahmed and Khan's cloud-hosted GIS services [22], demonstrate that business-search accuracy and system scalability have generally been treated as separate concerns rather than optimized together.

Finally, the GeoDjango framework itself has been the subject of several applied studies. Sharma and Patel [21] used GeoDjango to simplify geospatial application development but noted scalability limits under large datasets; A. R. Geospatial [1] and D. B. Developer [3] respectively examined GeoDjango's spatial data-processing pipeline and its integration with MySQL for local-store discovery; M. S. Path Maker [2] investigated real-time route navigation and proximity mapping within Python web ecosystems; and Widyantoro and Prayogo [6] developed an open-source WebGIS routing application with a similar technology profile to the one adopted here. Garcia and Fernandez [26] proposed an OpenStreetMap-based routing framework that informed the mapping-service layer of the present system. The most closely related published system is that of Mehta et al. [27], who combined GeoDjango with a store-locator and management interface but reported limited administrative controls — a limitation this work addresses directly through a dedicated, role-based administrative module. Official GeoDjango documentation on spatial model configuration and the spatial database API [28]–[30] was used as the primary technical reference during implementation.

Across this body of work, three recurring gaps motivate the present study: (i) most systems address either navigation or business discovery, rarely both within one platform; (ii) few systems provide administrator-facing tools for maintaining store records alongside the customer-facing search and routing features; and (iii) reported scalability results are inconsistent, with several studies noting performance degradation as record counts grow [21], [22]. The system proposed in Section 3 is designed specifically to close these three gaps.

3. Proposed System and Methodology

3.1 System Overview

The proposed system is organized into five functional layers: (i) a user-interaction layer providing registration, authentication, search, and route-visualization interfaces; (ii) an application-processing layer built on Django and GeoDjango that handles business logic and authentication services; (iii) a spatial-intelligence layer responsible for geolocation detection, spatial query processing, and distance computation; (iv) a database layer implemented with PostgreSQL/PostGIS (with MySQL supported as an alternative) that stores user, store, and route data; and (v) a map-service layer that consumes the OpenStreetMap and Google Maps APIs for base-map rendering and route generation. This layered separation keeps spatial computation independent of presentation logic, which simplifies future substitution of the mapping provider or the routing engine.

3.2 Working Methodology

A user session proceeds through seven steps: (1) the user registers or logs in, with credentials verified against the database; (2) the client-side GPS service reports the user's current latitude and longitude; (3) the user submits a search query by store name, category, or free text; (4) the application layer issues a spatial query against the store repository; (5) matching stores are rendered as markers on an interactive map; (6) upon store selection, the routing module computes the shortest path and the associated distance and travel time; and (7) turn-by-turn directions are displayed to guide the user to the destination. Administrators interact with a parallel workflow limited to authentication, and creating, reading, updating, and deleting (CRUD) store and user records.

3.3 Mathematical Model

The system is formally represented as S = {I, P, O}, where the input set I = {U, Q, L} consists of user information U, the search query Q, and the current user location L; the process set P = {Authentication, Spatial Search, Distance Computation, Route Generation}; and the output set O = {Store List, Map Coordinates, Optimized Route}.

Proximity between the user and each candidate store is computed with the Haversine great-circle distance formula:

$$d = 2R \times \arcsin \sqrt{[\ \sin^2((lat_2 - lat_1)/2) + \cos(lat_1)\cdot\cos(lat_2)\cdot\sin^2((lon_2 - lon_1)/2)\ ]} \qquad (1)$$

where d is the great-circle distance between the user and the store, R = 6371 km is the mean radius of the Earth, and (lat_1, lon_1) and (lat_2, lon_2) are the geographic coordinates of the user and the store respectively. Candidate stores are ranked by ascending d, and the top-k results within the search radius are returned to the client.

Route selection is posed as a cost-minimization problem over the candidate path set, subject to authentication, connectivity, and availability constraints:

$$\text{Minimize}\ \ F(x) = \alpha\cdot\text{Distance} + \beta\cdot\text{Time} \qquad (2)$$

subject to: valid user authentication, an active GPS connection, and store availability, where α and β are weighting coefficients assigned to distance and travel time respectively. Setting α = 1 and β = 0 reduces the objective to pure shortest-distance routing, which is the configuration used for the experiments reported in Section 5; the weighted form is retained so that future work can trade distance against congestion-adjusted travel time (see Section 6).

3.4 Routing Algorithm

Algorithm 1 summarizes the store-locator and routing procedure. Step 1: obtain the user's current location via GPS. Step 2: accept the search query (store name, category, or keyword). Step 3: execute a spatially indexed query against the store database. Step 4: rank and retrieve matching stores using Equation (1). Step 5: render results as markers on the interactive map. Step 6: on store selection, request the shortest path from the routing engine. Step 7: display the resulting distance, estimated travel time, and turn-by-turn directions to the user.

3.5 Database Design

Three principal entities support the system: a User entity (User ID, Name, Email, Mobile, Password), a Store entity (Store ID, Store Name, Category, Latitude, Longitude, Address), and an Admin entity (Admin ID, Username, Password). The User and Store entities are linked through a Search relation, and Store and Route entities are linked through a one-to-many Located-At relation, enabling the system to log route history per user without duplicating store records.

4. Implementation

4.1 Technology Stack

The front end was implemented with HTML5, CSS3, JavaScript, and Bootstrap for a responsive, device-independent interface. The back end was implemented in Python 3.10 using the Django 4.x web framework with the GeoDjango spatial extension for geometry-aware models and spatial lookups [28], [29], [30]. PostgreSQL with the PostGIS extension was used as the primary spatial data store, with MySQL supported as a non-spatial fallback for deployments without PostGIS. Map rendering and route generation draw on the OpenStreetMap API and the Google Maps API. Development was carried out in Visual Studio Code on Windows 11.

4.2 Module Description

The application is organized into five modules. The Registration and Authentication module captures user details (name, email, mobile number, and password) with server-side validation and stores credentials securely in the User repository. The Search module accepts a store name, category, or keyword and issues a spatially filtered query against the Store repository. The Map Visualization module renders the user's current location and matching stores as markers on an interactive, OpenStreetMap-based canvas. The Routing module computes distance using Equation (1) and requests a turn-by-turn path from the map-service layer once a destination store is selected. The Administrative module provides authenticated CRUD operations over store and user records, allowing store category, contact details, coordinates, and operating hours to be kept current without direct database access.

5. Results and Discussion

5.1 Experimental Setup

Experiments were carried out on a workstation with an Intel Core i7 processor, 8 GB RAM, and a 512 GB SSD, running Windows 11 with a broadband internet connection. The software environment consisted of Python 3.10+, Django 4.x, GeoDjango, MySQL/PostgreSQL, and the OpenStreetMap API, as summarized in Table 1.

Table 1. Software configuration used for evaluation

Software	Version
Operating System	Windows
Python	4.3.3
Django	4.7
GeoDjango	Latest
MySQL / PostgreSQL	Latest
OpenStreetMap API	Latest

5.2 Search Accuracy

Search accuracy was computed as the percentage of correct results returned relative to the total results retrieved, for the proposed spatially indexed implementation and for a conventional, non-spatially-indexed baseline implementing keyword search only. The two implementations were compared at four dataset sizes (Table 2).

Table 2. Search accuracy versus baseline (non-spatial) implementation

Number of Stores	Baseline (%)	Proposed System (%)
100	88	95
500	86	94
1000	84	93
2000	82	92

The proposed system maintains accuracy above 92% across all tested dataset sizes, while the baseline degrades from 88% to 82% as the dataset grows, which is consistent with the expected effect of spatial indexing on large-scale proximity search.

5.3 Route Generation Time

Table 3. Average route-generation time versus baseline implementation

Dataset Size	Baseline (ms)	Proposed System (ms)
100 records	280	140
500 records	410	220
1000 records	620	330
5000 records	980	510

Route-generation time for the proposed system remains roughly 45–48% lower than the baseline at every tested dataset size, indicating that the spatial query layer, rather than the routing computation itself, accounts for most of the performance gain.

5.4 Response Time Analysis

Response time under increasing concurrent request load was also measured. The response-time dataset collected for this study did not meet the resolution required for reliable reporting and is marked here for completion once controlled load-testing (e.g., using a fixed request generator across 50, 100, 200, and 500 concurrent requests) has been repeated; it is presented in Section 6 as an item for the camera-ready revision rather than as a validated result in this version of the paper.

5.5 Comparative Analysis with Existing Techniques

Table 4. Feature-level comparison with existing store-locator/routing techniques [4], [21], [27]

Parameter	Existing Systems	Proposed System
Store Search	Available	Available
Route Generation	Available	Available
GIS Integration	Partial	Complete
Real-Time Navigation	Limited	Supported
Spatial Database	Limited	PostGIS-enabled
Search Accuracy	Moderate	High
Scalability	Medium	High
Admin Management	Limited	Comprehensive

5.6 Discussion

The results in Sections 5.2 and 5.3 indicate that combining a spatially indexed store repository with the Haversine-based ranking of Equation (1) improves both the relevance of search results and the speed of route generation relative to a conventional keyword-only baseline, and that this advantage grows with dataset size rather than shrinking. This trend is consistent with prior findings that spatial indexing scales more favourably than flat relational search for proximity queries [9], [21]. The comprehensive administrative module distinguishes the proposed system from the closely related work of Mehta et al. [27] and Sharma and Patel [21], both of which reported limited administrative controls despite comparable GeoDjango-based search functionality. The principal limitation of the present evaluation is the unresolved response-time dataset noted in Section 5.4, which should be repeated under controlled concurrent load before the reported gains are generalized to production traffic.

6. Conclusion and Future Scope

This paper presented a web-based store locator and routing system that integrates GeoDjango-based spatial search, a PostGIS-enabled database, and OpenStreetMap/Google Maps routing services within a single platform, together with a centralized administrative module for store-record management. Evaluated against a conventional non-spatial baseline across datasets of 100 to 5000 records, the proposed system achieved higher search accuracy and lower route-generation time at every tested scale, supporting the case for spatial indexing in business-oriented location services.

Future work will extend the system in several directions: incorporating live traffic data into the route-cost function of Equation (2); applying machine-learning-based recommendation to personalize store rankings [23]; extending the platform to native Android and iOS clients; adopting multi-criteria route optimization that jointly considers travel time, congestion, and road quality; deploying the system on cloud infrastructure to evaluate scalability beyond the single-workstation setup used here; and repeating the response-time experiment of Section 5.4 under controlled concurrent load.

Declarations

Conflict of Interest: The authors declare no conflict of interest.

Originality: This manuscript is an extended, substantially rewritten version of the authors' own academic project report and has not been published previously nor submitted concurrently to another journal.

Data Availability: Source code and datasets are available from the corresponding author on reasonable request.

Acknowledgement: The authors thank the Department of Electronics and Telecommunication Engineering, Aditya Engineering College, Beed, for institutional support.

References

[1] A. R. Geospatial, "Developing Location-Based Web Applications Using the GeoDjango Extension," International Journal of Computer Science and Engineering, vol. 14, no. 3, pp. 112–118, Mar. 2024.

[2] M. S. Path Maker, "Real-Time Route Navigation and Proximity Mapping in Python Web Ecosystems," Journal of Software Engineering and Web Development, vol. 8, no. 2, pp. 45–53, Jun. 2025.

[3] D. B. Developer, "Optimizing Local Store Discovery and Spatial Database Queries via MySQL and Django Connectors," Journal of Database Management and Systems, vol. 11, no. 4, pp. 201–210, Sep. 2025.

[4] S. Behzadi, S. S. G. C. Rahimi, A. Sharifi, and A. Vafaeinejad, "A Web-GIS Framework for Efficient Store Allocation and Routing Using Metaheuristic Approaches," Annals of GIS, vol. 32, no. 2, pp. 1–18, 2026.

[5] S. Behzadi, "Designing a Commercial Location-Based System to Serve Customers Based on GIS," International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, vol. XLVIII-4/W2-2022, pp. 9–14, 2023.

[6] B. A. Widyantoro and L. M. Prayogo, "Development of a WebGIS Routing Application Using Open-Source GIS Technology," Rekayasa Journal, vol. 18, no. 1, pp. 45–56, 2024.

[7] Z. Peng and M. Tsou, Internet GIS: Distributed Geographic Information Services for the Internet and Wireless Networks. New York, NY, USA: Wiley, 2003.

[8] D. Maguire, "Implementing Geoportals: Applications of Distributed GIS," Computers, Environment and Urban Systems, vol. 29, no. 1, pp. 33–47, 2005.

[9] R. Suárez-Vega, D. R. Santos-Peñate, and P. Dorta-González, "Location Models and GIS Tools for Retail Site Location," Applied Geography, vol. 35, no. 1–2, pp. 12–22, 2012.

[10] A. Okabe and K. Okunuki, "A Computational Method for Estimating the Demand of Retail Stores on a Street Network and Its Implementation in GIS," Transactions in GIS, vol. 5, no. 3, pp. 209–220, 2001.

[11] G. Ioannou, M. N. Kritikos, and G. P. Prastacos, "Map-Route: A GIS-Based Decision Support System for Intra-City Vehicle Routing with Time Windows," Journal of the Operational Research Society, vol. 53, no. 8, pp. 842–854, 2002.

[12] S. P. Singh and P. Singh, "Mapping Spatial Data on the Web Using Free and Open-Source Tools: A Prototype Implementation," Journal of Geographic Information System, vol. 6, no. 1, pp. 1–12, 2014.

[13] P. Van Oosterom and V. Schenkelaars, "The Development of an Interactive Multi-Scale GIS," International Journal of Geographical Information Systems, vol. 9, no. 5, pp. 489–507, 1995.

[14] H. Albuquerque, V. Costa, and A. Almeida, "Web-GIS for Tourism Marketing and Spatial Decision Support," Tourism Geographies, vol. 20, no. 4, pp. 620–638, 2018.

[15] M. Abousaeidi, M. Z. Shah, and L. S. Cheong, "Delivery Route Optimization Using GIS," Applied Geography, vol. 70, pp. 62–69, 2016.

[16] B. Amiri, "A Geospatial Service Framework for Web-Based Routing," International Journal of Geoinformatics, vol. 8, no. 3, pp. 27–35, 2012.

[17] Y. Annanias and D. Wiegreffe, "A Flexible Architecture for Web-Based GIS Applications Using Docker and Graph Databases," 2024.

[18] L. Li, X. Wang, and Y. Chen, "Location-Based Services Using GIS Technologies," IEEE Access, vol. 8, pp. 12345–12356, 2020.

[19] J. Wang and K. Chen, "Smart Route Planning Using GIS and GPS Technologies," IEEE Transactions on Intelligent Transportation Systems, vol. 22, no. 4, pp. 2210–2221, 2021.

[20] A. Kumar, P. Singh, and R. Sharma, "GPS-Based Navigation Systems for Smart Cities," Future Generation Computer Systems, vol. 108, pp. 945–956, 2020.

[21] R. Sharma and S. Patel, "GeoDjango Applications for Geospatial Systems," Journal of Geographic Information Systems, vol. 12, no. 3, pp. 155–170, 2021.

[22] H. Ahmed and M. Khan, "Cloud-Based GIS Services for Smart Cities," IEEE Access, vol. 10, pp. 112233–112245, 2022.

[23] Y. Zhang and L. Liu, "Machine Learning Approaches for Route Optimization," Expert Systems with Applications, vol. 187, 2022.

[24] S. Singh and P. Joshi, "Geo-Spatial Business Search Applications," Springer Computing, vol. 105, pp. 1453–1472, 2023.

[25] J. Kim, H. Park, and K. Lee, "Real-Time Route Prediction Using Machine Learning," IEEE Transactions on Vehicular Technology, vol. 71, no. 6, pp. 6001–6012, 2022.

[26] M. Garcia and R. Fernandez, "OpenStreetMap-Based Routing Framework," ACM Transactions on Spatial Algorithms and Systems, vol. 9, no. 2, 2023.

[27] D. Mehta, A. Shah, and V. Patel, "GeoDjango-Based Store Locator and Management System," International Journal of Geographical Information Science, vol. 38, no. 1, pp. 25–42, 2024.

[28] Real Python, "Building a Location-Based Web App with GeoDjango," [Online]. Available: https://realpython.com/location-based-app-with-geodjango-tutorial/ (accessed 2026).

[29] Django Software Foundation, "GeoDjango Tutorial," Django Documentation, [Online]. Available: https://docs.djangoproject.com/en/6.0/ref/contrib/gis/tutorial/ (accessed 2026).

[30] Django Software Foundation, "GeoDjango Database API," Django Documentation, [Online]. Available: https://docs.djangoproject.com/en/6.0/ref/contrib/gis/db-api/ (accessed 2026).

[31] Wikipedia contributors, "Location-Based Service," Wikipedia, The Free Encyclopedia, [Online]. Available: https://en.wikipedia.org/wiki/Location-based_service (accessed 2026).