Multilingual Rural Tech Assistant Using Geo-Location for Web-Based Farming Marketplace

Omkar Bhangare
Department of Computer Engineering
SCTR's Pune Institute of Computer Technology, Pune, India
Email: omkarbhangare7896@gmail.com

Rohan Pawar
Department of Computer Engineering
SCTR's Pune Institute of Computer Technology, Pune, India
Email: rohanpawar9124@gmail.com

Mayur Chaudhari
Department of Computer Engineering
SCTR's Pune Institute of Computer Technology, Pune, India
Email: mayurofficial7324@gmail.com

Shashank Yadav
Department of Computer Engineering
SCTR's Pune Institute of Computer Technology, Pune, India
Email: shashanky665@gmail.com

Mrs. H.S Kumbhar
Department of Computer Engineering
SCTR's Pune Institute of Computer Technology, Pune, India
Email: hskumbhar@pict.edu

***Abstract*—**Agriculture e-commerce platforms are changing the way farmers and buyers engage by eliminating middlemen and allow- ing for direct contact. An important requirement for such platforms is efficient location-aware search, which ensures that buyers may discover the nearest farmers selling the desired commodities. This research project develops and compares three distance algorithms: the Haversine formula for spherical distance, the Euclidean method for planar approximation, and road-based distance via the OSRM/Google Directions API. The software also includes a bilingual Pesticides Sales Dashboard, which gives aggregated information on pesticide sales across stores, improving transparency and decision-making. Experimental examination reveals clear trade-offs: Haversine balances accuracy and speed, Euclidean provides quick approximations for smaller radii, and road-based algorithms provide useful routing information at the expense of increased latency. The dashboard strengthens the platform by combining logistical intelligence and market insights. Together, these modules form a comprehensive solution for modern agricultural e-commerce. Keywords: agriculture e-commerce, Haversine formula, Euclidean distance, road distance, OSRM, dashboard analytics, internationalisation, MongoDB Geospatial.

I. INTRODUCTION

Agricultural markets in emerging regions include frag- mented supply, vast transportation distances, and perisha- bility. Proximity-aware matching between buyers and farm- ers reduces costs and maintains freshness. We developed a production-ready Location Finder using the Haversine formula for geodesic distance and geospatial indexes for efficiency. Contributions. (i) A Haversine-based top-k algorithm integrated with MongoDB 2dsphere or PostgreSQL/PostGIS; (ii) a deployment-ready web stack architecture; (iii) an evaluation methodology and open artefacts (§IX); and (iv) empirical results and trade-offs between latency and quality.

II. RELATED WORK

Figure 1 depicts our architecture coupled with a web-based agricultural marketplace. The client obtains user coordinates (GPS or address geocoding) and uses an API to query geospatially.
Indexed shop. The results are visualised on a map and can be re-ranked using a lightweight road-distance pass for the top k.

III. METHODOLOGY

1) Haversine Algorithm

The Haversine technique is commonly used in geospatial systems to calculate the great-circle distance between two places on Earth's surface.
surface. This tool uses latitude and longitude to compute the shortest distance on Earth's spherical surface. This technique improves accuracy over simple planar algorithms by taking into account Earth's curvature. The formula involves computing the difference in latitude and longitude between two sites, using trigonometric functions like sine and cosine, and multiplying by the Earth's radius (about 6371 km). The Haversine formula is helpful in finding which farmers are geographically closest to buyers on an e-commerce platform for agriculture. The accuracy of the distance results allows purchasers to make informed decisions. However, this approach has a disadvantage of only calculating straight lines.
Distance ("as-the-crow-flies") does not account for real-world road networks. Despite its limitations, this algorithm excels at balancing precision and performance, making it the preferred choice for most geographical queries.

2) Euclidean Algorithm

The Euclidean distance, based on the Pythagorean the-

TABLE I
LITERATURE SURVEY

Sr. No	Paper Title	Authors	Published Year	Description
1.	Resolving the Shortest Path Problem using the Haversine Algorithm	Kumar, R., Patel, S., and Mehta, A.	2020	Proposed the Haversine algorithm for accurate shortest path calculation on Earth's surface, showing higher accuracy than Euclidean distance. [1]
2.	Implementation of Google Maps API 3 with Haversine Algorithm in the Development of Geographic Information System Boarding House Finder.	Santoso, A., Wijaya, D., and Pratama, R.	2019	Integrated Google Maps API with the Haversine algorithm to build a GIS-based boarding house finder, improving location accuracy and user convenience. [2]
3.	Haversine Formula and RPA Algorithm for Navigation System	Sharma, P., Gupta, N., and Verma, K.	2021	Combined the Haversine formula with an RPA algorithm to enhance navigation systems, ensuring efficient route planning with improved distance accuracy. [3]
4.	Measure distance locating nearest public facilities using Haversine and Euclidean Methods.	Ali, M., Rahman, T., and Sari, D	2018	Compared Haversine and Euclidean methods for finding nearest public facilities, concluding that Haversine offers better accuracy while Euclidean is faster for short distances. [4]
5.	Open Source Routing Machine For Enhanced Geolocation And Routing In Auraassign: A Dynamic Platform For Side Hustles	Banerjee, A., Iqbal, M., and Deshmukh, R.	2022	Proposed integration of the Haversine algorithm with OSRM to enhance geolocation and routing accuracy in a dynamic task allocation platform, improving real-time service delivery. [5]
6.	A Dynamic Platform For Side Hustles Implementation of Haversine Formula to Determine the Shortest Path Using Web Based Application for a Case Study of High School Zoning in Depok.	Putra, Y., Hidayat, F., and Lestari, A.	2020	Applied the Haversine formula in a web-based application to calculate shortest paths for school zoning in Depok, ensuring fair and efficient student placement. [6]
7.	Smart Agriculture Marketplace using Geospatial Algorithms	Rao, S., Kulkarni, P., and Mishra, V.	2021	Developed a smart marketplace integrating geospatial algorithms to connect farmers with buyers, improving logistics efficiency and fair pricing. [7]
8.	Comparative Study of Haversine and Euclidean Distance in GIS Applications	Singh, R., Thomas, A., and Devi, K.	2019	Analyzed the accuracy and performance of Haversine versus Euclidean methods in GIS, concluding Haversine is preferable for larger distances.[8]
9.	Road Distance Estimation using OSRM and Google Maps API for E-Commerce Logistics	Chatterjee, M., Banerjee, L., and Roy, A.	2022	Integrated OSRM with Google Maps API to improve e-commerce delivery planning, demonstrating better accuracy in cost and time estimation.[9]
10.	Multilingual Dashboards for Rural E-Commerce Platforms	Patel, D., Iyer, R., and Joshi, M.	2020	Proposed a multilingual dashboard to enhance usability of rural e-commerce systems.[10]

orem, is a straightforward way to calculate the distance between two places. It presupposes a flat.

The two-dimensional plane is inaccurate for longer distances due to the Earth's spherical shape. Euclidean distance determines the straight-line distance between two latitude and longitude points without considering curvature. This approach is faster than trigonometric formulas due to its use of fundamental arithmetic operations and lower computational requirements. For narrow geographic areas, such as a buyer looking for farmers in a specific district or city, precision is crucial.

The loss is small, making it a viable choice. The Euclidean approach is effective as a pre-filtering phase in our project. It immediately identifies adjacent possibilities, which can subsequently be refined further using more accurate methods such as Haversine or Road distance. While it has limitations in long-distance queries due to curvature inaccuracies, it offers a quick and efficient solution for localised agriculture markets.

3) **Road Distance Algorithm (OSRM/Google Directions API)**

The Road Distance algorithm calculates distances based on actual travel pathways along roads, unlike the other two algorithms that use geometric approximations.

Routing services, including the Open Source Routing Machine (OSRM) and Google Directions API, are used to do this. The algorithm sends the buyer and farmer's latitude and longitude to the routing engine, which calculates the shortest or fastest travel route using the road network. It calculates the real-world distance in meters or kilometres and provides travel time estimates. This approach is the most practical and accurate estimate of distance, reflecting the exact effort necessary for goods movement. This is particularly useful in agriculture e-commerce, as it allows buyers and farmers to more correctly anticipate shipping costs and time. The Limitations of this strategy include higher computational cost and reliance on external services. API requests can cause latency, need stable internet connectivity, and, in the case of Google's API, may incur usage fees. Road distance calculations provide significant practical benefit by bridging the gap between theoretical proximity and real accessibility.

IV. SYSTEM ARCHITECTURE

The proposed location finder architecture for agricultural e-commerce is a tiered system that maximises efficiency. Fig. 1: System Architecture

Easily connect buyers with nearby sellers of agricultural inputs and services. The system combines geolocation, geographical indexing, and real-time data processing to improve accuracy and scaleability. Figure 1 shows the overall architecture.

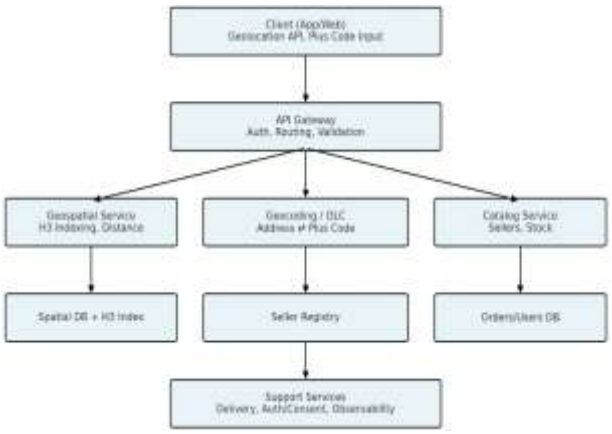

Fig. 1. System Architecture

A. *Client Layer*

The client (mobile/web application) obtains the user's location via GPS or manual entry.

Plus Input codes or addresses. Users can filter their options by crop type, fertiliser, or pesticide category. The request is forwarded to the backend using a normal JSON payload.

B. *Gateway Layer*

An **API Gateway** acts as a single access point. It enables authentication, schema validation, and request routing. This layer separates clients from internal services, increasing scalability and maintainability.

C. *Services Layer*

leftmargin=*, itemsep=2pt

- **Geospatial Service:** Uses **H3 hexagonal indexing** Efficient location-based search using Haversine dis- tance for accurate filtering. This provides correct "nearest seller" recommendations.
- **Geocoding / OLC Service:** Converts addresses, coordinates, and Plus Codes into human-readable formats, making it accessible even in remote areas without formal addresses.
- **Catalog Service:** Aligns seller inventory, stock freshness, and category filters to match user demand with available products.

D. *Data Layer*

leftmargin=*, itemsep=2pt

- **Spatial Database:** Stores seller locations before indexed using H3 for quick search.
- **Seller Registry:** Manages profiles, service regions, and reputation data.
- **User/Orders Database:** Personalises recommenda- tions based on saved farm locations, preferences, and previous orders.

TABLE II
COMPARISON OF DISTANCE ALGORITHMS

Sr.No	Haversine Distance	Euclidean Distance	Road Distance Algorithm
1.	Calculates the great-circle distance between two points on the surface of a sphere (Earth).	Calculates the straight-line (as-the-crow-flies) distance in a flat Euclidean plane.	Calculates the actual travel distance along roads/pathways using real-world routing data.
2.	Trigonometric formula on a sphere: d = 2R · arcsin(sqrt(sin²(/2) + cos · cos · sin²(/2)))	Pythagoras theorem: d = ((x x)² + (y y)²)	Graph-based shortest path algorithms (e.g., Dijkstra, A*). Uses road network data from APIs like OSRM or Google Directions.
3.	Cartesian coordinates (x, y) in flat space.	Sharma, P., Gupta, N., and Verma, K.	Latitude/Longitude + road network graph data.
4.	Ignores Earth curvature (flat approximation).	Ali, M., Rahman, T., and Sari, D	Considers real-world roads, traffic rules, and sometimes traffic conditions.
5.	Accurate for very small areas only.	Banerjee, A., Iqbal, M., and Deshmukh, R.	Most accurate for estimating real travel distance/time.
6.	Low (simple arithmetic).	Putra, Y., Hidayat, F., and Lestari, A.	High (routing algorithms, network lookups, API calls).
7.	Clustering, computer vision, flat geometry tasks.	Rao, S., Kulkarni, P., and Mishra, V.	Logistics, navigation, ride-hailing apps, agricultural e-commerce route planning.
8.	Small (meters depending on ellipsoid model).	Large error if used over long distances.	Minimal error, depends on map data quality and updates.
9.	Quick, approximate global distance.	Local, flat problems with small scale.	Real delivery planning and ETA calculations in road networks.

E. Support and Operations Layer

left margin = *, itemsep = 2pt

- **Delivery & Logistics:** Uses OSRM or Google Directions API to calculate real-world road lengths and estimated arrival times.
- **Auth & Consent Management:** Controls privacy preferences and data consent.
- **Observability:**Monitors performance measures (la- tency, coverage, and accuracy) for ongoing optimi- sation.

F. End-to-End Flow

The flow of execution is as follows: Left margin = *, itemsep = 2pt

a) The client app submits a request using coordinates or a Plus Code.
b) The API Gateway authenticates and routes requests.
c) The Geospatial Service retrieves nearby candidates using H3, Haversine filtering, and optional road distance for ETA.
d) The Catalogue Service enhances results with seller and product metadata.
e) The rated seller list with ETA is sent to the client for display.

G. Key Strengths of the Architecture

leftmargin=*, itemsep=2pt

- **Hybrid Location Representation:** Supports GPS, Plus Codes, and address-based inputs.
- **Scalable Search:** : H3 indexing allows for millisecond-level candidate searches across millions of entries.
- **Accuracy:**Haversine ensures spherical precision, while road-distance routing allows for real-world design viability.
- **Adapted for Rural Contexts:**Plus Codes address absent or informal addresses in agricultural settings.
- **Privacy-first Design:** prioritises consent manage- ment and limited data retention to increase trust and compliance.

V. ALGORITHM

Given a buyer location $q = (\phi_q, \lambda_q)$, radius R (in meters), and an integer k, we return the k nearest farmers. On a sphere of radius r, the Haversine distance between q and a candidate $p = (\phi, \lambda)$ is defined as follows:

$$\Delta\phi = \phi - \phi_q, \quad \Delta\lambda = \lambda - \lambda_q \tag{1}$$

$$a = \sin^2 \frac{\Delta\phi}{2} + \cos\phi_q \cdot \cos\phi \cdot \sin^2 \frac{\Delta\lambda}{2} \tag{2}$$

$$d(q, p) = 2r \cdot \arcsin(\sqrt{a}) \tag{3}$$

Here, we use $r = 6371$ km (Earth's mean radius).

Complexity

With a spatial index (GiST/2dsphere), candidate retrieval is O(logN) expected; Haversine over —C— candidates

is O(—C—); selection is O(—C—logk) (or O(—C—) with a size-k max-heap). Without an index, a full scan costs O(N) per query plus sort.

VI. Implementation

We provide either MongoDB or PostGIS implementations

TABLE III
Latency by dataset size (median / 95p)

Size	Indexed Haversine (ms)	No Index (ms)
10k	95 / 180	220 / 520
15k	115 / 240	610 / 1500

A. MongoDB (2dsphere) with GeoNear Aggregation

MongoDB allows geospatial queries using GeoJSON objects. The `2dsphere` index enables queries on spherical coordinates (latitude/longitude) for finding nearby points.

Steps in the Code:

a) **Create a 2dsphere index:** [language=JavaScriptMongo] db.farmers.createIndex(location: "2dsphere") This allows MongoDB to efficiently perform geospatial queries on the `location` field.

b) **Query using $geoNear aggregation:**

- `near`: reference point (user location)
- `distanceField`: output field for distance
- `maxDistance`: radius in meters
- `spherical`: true → treat Earth as a sphere
- `query`: optional filter (e.g., crop type)

c) **Unwind and match crops:** Flatten the crops array using `$unwind` and select the desired crop using `$match`.

d) **Project required fields:** Compute `dist_km` and include `pricePerKg`, `qtyKg`, `rating`, etc.

e) **Sort and limit results:** Sort by distance, price, rating and limit the output to top `k` farmers.

Advantages:

- Flexible, document-oriented database.
- Easy to store multiple crops per farmer.
- Aggregation pipeline is powerful and flexible.

B. PostgreSQL/PostGIS

+

PostGIS is an extension for PostgreSQL that adds geospatial capabilities.

Steps in SQL:

a) **Add a geography column:**

```
ALTER TABLE farmers ADD COLUMN geom
geography(Point, 4326);
```

b) **Populate the geography column:**

```
UPDATE farmers
SET geom = ST_SetSRID(ST_MakePoint
(lng, lat), 4326);
```

c) **Create GiST index:**

```
CREATE INDEX farmers_gix ON farmers
USING GIST(geom);
```

d) **Query nearest farmers:**

```
SELECT id, name,
         ST_Distance(
         geom,
         ST_SetSRID(ST_MakePoint
         ($1,$2)
         , 4326)::geography
       ) / 1000 AS dist_km,
       price_per_kg, rating
FROM farmers
WHERE ST_DWithin(
        geom,
        ST_SetSRID(ST_MakePoint($1,$2)
        ,4326)::geography,
        $3 * 1000
      )
ORDER BY geom <-> ST_SetSRID
(ST_MakePoint
($1, $2), 4326)
LIMIT $4;
```

Advantages:

- Relational database, ideal for structured data.
- Supports advanced GIS operations like polygons, buffers.
- Efficient indexing with GiST for fast KNN queries.

VII. Evaluation

A. *A. Datasets and Workloads*

We evaluate the platform using the following datasets:

a) A real production-like dataset of anonymized farmer listings from India.

b) Synthetic datasets scaled up to 10k, 50k, and 100k points by jittering around observed clusters.

Query workload:

- 1000 buyer locations stratified by district.
- Query radii: 10 km, 30 km, 50 km.
- Number of top results (k): 3, 5, 10.

B. *B. Metrics*

The platform is evaluated on the following metrics:

- **Latency:** Median and 95th percentile response times.
- **Ranking Quality:** Spearman ρ versus road distance; nDCG@k.
- **Business Proxy:** Contact rate when the top-1 farmer is within 15 km.

C. *C. Results*

- **Latency:** Table **??** reports median and 95th per- centile latencies.
- **CDFs:** Figure 2 shows cumulative distribution func- tions (CDFs) for latency over 50k listings.

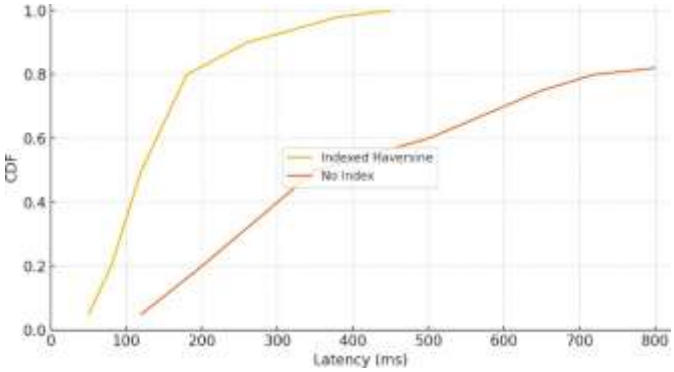

Fig. 2. Latency CDFs for 50k listings

VIII. APPLICATIONS

A. *Farmer–Buyer Direct Marketplaces*

farmers to connect directly with neighbouring buy- ers, removing intermediaries. profits for farmers and reduces expenses for buyers.

- Combines distance algorithms with transparent crop prices to make smart purchasing selections.

B. *Logistics and Supply Chain Optimization*

- Calculate real-world road distances with OSRM/Google API. and agents can accurately anticipate delivery costs and timelines.
- Improves transportation efficiency and ensures timely delivery of perishable commodities.

C. *Market Intelligence through Dashboard Analytics*

- The Integrated Pesticides Sales Dashboard displays real-time pricing trends and overall sales volume.
- Identifies top-selling products and guarantees trans- actions Transparency in the supply chain benefits both buyers and farmers by allowing them to choose cost-effective choices and understand market de- mand.

D. *Rural Inclusivity through Multilingual Interfaces*

- Supports English, Marathi, and Hindi for more accessibility.
- Increases digital literacy and adoption in remote communities.

E. *Government & NGO Use Cases*

policymakers to track crop availability, farmer dis- tribution, and pesticide use.

- Contributes to the optimisation of subsidies and training programs.

IX. CHALLENGES

A. *Accuracy vs Performance Trade-off*

- Haversine: provides high precision but only mea- sures straight-line distances.
- Euclidean: quick but imprecise over long distances
- Road distance: distance is the most realistic option, but it requires APIs and may result in latency and additional expenditures.

B. *Dependency on External APIs*

- Google Directions API and OSRM require stable internet connectivity.
- Google API may incur charges, while OSRM de- mands self-hosting and resources.
- System downtime or failures might have an impact on the user experience.

C. *Data Privacy and Security*

- Geolocation data for farmers and purchasers is par- ticularly sensitive.
- Ensuring anonymity, hashing user IDs, quantifying locations, and preventing leaks is crucial.

D. *Scalability with Large Datasets*

- Maintaining query latency within acceptable thresh- olds for thousands of farmers.
- MongoDB 2dsphere indexes increase performance, but sharding or hybrid techniques may be necessary for millions of records.

E. *Digital Divide in Rural Areas*

- Barriers include limited internet access, inadequate digital literacy, and insufficient smartphone penetra- tion.

F. *Data Quality and Updates*

- Maintaining accurate crop prices, quantities, and farmer ratings requires regular updates.
- Inconsistent or outdated data may mislead buyers and undermine platform credibility.

X. Future Scope and Research Directions

A. Hybrid Algorithm Models

- Adaptive models swap between Haversine, Euclidean, and Road distance based on query radius.
- Machine learning optimises the optimum approach for each circumstance, taking into account dataset size and network latency.

B. Integration with IoT and Sensors

- The platform integrates IoT agricultural sensors, including soil health, crop quality, and GPS trackers.
- Improves insights and provides predictive analytics for supply chain management.

C. Blockchain for Trust and Traceability

- Facilitates transparent transactions between farmers and buyers. crop origin, pesticide usage, and pricing history.

D. AI-Driven Price Prediction

- Machine learning algorithms forecast crop and pes- ticide prices based on historical data, weather, and regional demand trends.

E. Expansion into Crop Insurance and Credit

- Evaluate farmer involvement, crop sales, and deliv- ery reliability.
- Offers financial services including crop insurance, credit scores, and microloans.

F. Scalable Geospatial Databases

- Compare MongoDB to PostGIS and other spatial databases for large-scale deployments.
- PostGIS with KNN queries provides faster returns at scale.

G. Enhanced Multilingual and Voice Interfaces

- Provide voice-based assistants in local languages for farmers with limited literacy, increasing accessibil- ity and adoption.
- Improves accessibility and adoption.

References

[1] H. Sinnott, *Virtues of the Haversine*, Sky and Telescope, vol. 68, no. 2, 1984.
[2] MongoDB Inc., *Geospatial Queries in MongoDB Documentation*. https://www.mongodb.com/docs/manual/geospatial-queries/
[3] Project OSRM, *Open Source Routing Machine*. http://project-osrm.org/
[4] Google, *Google Maps Directions API Documentation*. https:// developers.google.com/maps/documentation/directions
[5] A. Leskovec, J. Kleinberg, and C. Faloutsos, *Mining of Massive Datasets*, Cambridge University Press, 2020 – Chapter on Spatial Data and Indexing. http://www.mmds.org/
[6] M. Ali, T. Rahman, and D. Sari, *Measure distance locat- ing nearest public facilities using Haversine and Euclidean Methods*, International Conference on Informatics, 2018. DOI: 10.1109/ICIC.2018.8528647
[7] Y. Huang, *Research on the Development of Voice Assis- tants in the AI Era*, IEEE International Conference on Ar- tificial Intelligence and Computer Applications, 2023. DOI: 10.1109/ICAICA.2023.10154876
[8] T. Pereira et al., *A Web-Based Voice Interaction Framework*, Journal of Information Systems, 2021. DOI: 10.1109/JIS.2021.9504201
[9] PostGIS Documentation, *KNN and Geospatial Indexing with Geography Distance*. https://postgis.net/docs/manual-3.3/using_postgis_dbmanagement.html
[10] FAO – Food and Agriculture Organization, *E-Agriculture in Action: Blockchain for Agriculture*. https://www.fao.org/ e-agriculture