



Journal of Advanced Research in Computing and Applications

Journal homepage:
<https://karyailham.com.my/index.php/arca/index>
ISSN: 2462-1927



Design and Evaluation of a Lightweight Reverse Geocoding API for Malaysian Location-Description Retrieval Using Spatial Indexing

Mohammad Hafiz Ismail^{1,*}, Tajul Rosli Razak², Mohd Faris Mohd Fuzi¹

¹ Faculty of Computer and Mathematical Sciences, Universiti Teknologi MARA, 02600 Arau, Perlis, Malaysia

² Faculty of Computer and Mathematical Sciences, Universiti Teknologi MARA, 40450 Shah Alam, Selangor, Malaysia

ARTICLE INFO

Article history:

Received 17 July 2026

Received in revised form 18 August 2026

Accepted 19 September 2026

Available online 29 September 2026

Keywords:

reverse geocoding; spatial indexing;
BallTree; Web API; edge computing

ABSTRACT

Reverse geocoding converts geographic coordinates into meaningful location descriptions. This study developed a lightweight local reverse geocoding API for Malaysian location-aware and embedded computing applications. The dataset contained more than 54,000 location records, including repeated coordinates associated with multiple descriptions. These records were reorganised into spatial groups so that descriptions sharing the same coordinate could be returned together. Four retrieval methods, namely KD-Tree, BallTree, Geohash, and H3, were implemented and evaluated against a brute-force haversine baseline using 1,000 coordinate queries. The evaluation measured query latency, agreement with the baseline, additional retrieval distance, and index preparation time. KD-Tree and BallTree achieved 100% agreement with the baseline and recorded zero distance difference. BallTree reduced mean retrieval latency from 0.302 ms for brute-force search to 0.135 ms. Although Geohash and H3 achieved slightly lower latency, both produced mismatched retrieval results with kilometre-scale maximum distance differences. BallTree was therefore selected as the preferred method because it provided efficient retrieval while preserving complete agreement with the geographic-distance reference. The proposed API provides a practical local approach for Malaysian coordinate-to-description retrieval in embedded monitoring and location-aware systems.

1. Introduction

Location-aware systems commonly receive geographic coordinates from mobile devices, Global Positioning System (GPS) receivers, embedded controllers, Internet of Things (IoT) gateways, vehicle monitoring systems, field inspection applications, and environmental sensing platforms. However, latitude and longitude are machine-readable representations that do not directly provide meaningful place descriptions for users or downstream applications. Reverse geocoding addresses this requirement by converting geographic coordinates into human-readable location information, which is required in many location-based services and spatial analysis applications [1]. Similarly, raw GPS coordinates have limited semantic meaning unless they are associated with relevant geographic descriptions or contextual information [2]. For example, a coordinate such as 1.463430, 103.754715

* Corresponding author.

E-mail address: mohammadhafiz@uitm.edu.my

is technically valid, but it does not immediately indicate whether the point corresponds to a road segment, building, residential block, village, industrial zone, or another locally meaningful location. For many Malaysian local applications, the required function is not a complete commercial mapping service. Instead, a system may only need a controlled method for translating coordinates into the nearest available local description stored in a prepared reference dataset. This form of location-description retrieval is relevant for embedded monitoring, asset tracking, mobile field reporting, IoT dashboards, local logistics systems, institutional mapping, maintenance records, and community-level service applications. Previous studies have demonstrated the operational use of reverse-geocoded or spatially retrieved location information in mobile Web Geographic Information System (GIS) applications, environmental monitoring infrastructure, public facility identification, and emergency response planning [3], [4], [5], [6].

The need for a local solution is increased by several practical limitations of depending entirely on external reverse geocoding APIs. First, embedded and IoT systems may submit coordinate data continuously or at frequent intervals. When every query is sent to an external provider, API cost can accumulate as the number of monitored devices or reporting events increases. This is unnecessary when an application only needs to match a coordinate against a fixed Malaysian reference dataset. Yu et al. [7] examined an offline reverse geocoding algorithm based on a k-d tree and reported that the spatial indexing approach improved computational efficiency compared with a traditional search procedure. Their work supports the feasibility of implementing reverse geocoding as a local retrieval function rather than requiring an external geocoding service for every query.

A closed-loop embedded computing system should be able to continue basic location interpretation even when external internet access is unavailable, slow, or restricted. In a related edge-computing context, Horvath et al. [8] examined location-based service discovery for mobile-edge computing, showing that location awareness can be used to identify a suitable local or edge service instance. These studies indicate that local or near-source processing of location information is relevant for systems requiring immediate spatial interpretation.

Local deployment is also relevant when coordinate data represents operationally sensitive resources, including institutional equipment, field sensors, maintenance personnel, agricultural monitoring units, public facilities, or local infrastructure. A locally hosted reverse geocoding API reduces repeated transmission of raw coordinate queries to external systems and allows the organisation to control the reference dataset, access mechanism, and deployment environment. In the present study, this concern is treated as an architectural requirement rather than a direct privacy evaluation variable.

Commercial reverse geocoding services are appropriate for systems requiring comprehensive address coverage, road-network interpretation, administrative boundaries, routing services, or global mapping functionality. However, these capabilities may be excessive for a small local application that only requires the nearest available Malaysian location description. Spatial retrieval studies have shown that nearest-location processing can be conducted efficiently through indexed candidate search. Tree-based spatial indexing has been applied to reduce the number of candidate locations requiring further route-distance analysis. In studies involving public school accessibility and emergency trauma centre identification, the BallTree algorithm was used to identify nearby candidate locations before the final route evaluation was conducted through external directions services, such as the Google Maps API [4], [6]. Although both studies depended on external APIs for route-based distance processing, their findings indicate that BallTree is suitable for efficiently narrowing the spatial search space in location retrieval tasks.

Therefore, this study proposes the development of a lightweight reverse-geocoding API which loads the Malaysian reference dataset locally, groups location descriptions sharing the same

coordinate, and implements four spatial search methods: KD-Tree, BallTree, Geohash and H3. The methods are compared against a brute-force haversine baseline using a small, practical evaluation procedure. The intended outcome is a low-cost local geospatial service suitable for Malaysian embedded computing and location-aware applications that require nearest available location descriptions rather than a full address resolution.

Existing reverse geocoding research has mainly addressed output quality, geographic correctness, ranking mechanisms, uncertainty propagation, integration with external mapping services, or deployment within broader GIS applications [1], [5], [7]. However, a practical implementation gap remains for a lightweight local API that performs nearest location-description retrieval using a medium-sized Malaysian reference dataset and compares multiple spatial indexing methods under the same deployment procedure. In particular, the present dataset contains more than 54,000 records, precomputed Geohash values, and repeated coordinates associated with different location descriptions. This structure creates a realistic retrieval problem that is not adequately represented by a simple nearest-row search.

The present study therefore focuses on three implementation issues:

- i. How to design a local reverse geocoding API that handles repeated coordinates associated with multiple Malaysian location descriptions.
- ii. How to reduce repeated query cost using spatial indexing while preserving agreement with a brute-force geographic-distance reference.
- iii. How to determine whether KD-Tree, BallTree, Geohash or H3 is the most practical indexing method for lightweight web API deployment.

The rest of this article is organised as follows. Section 2 describes the methodology of the study, including the preparation and grouping of the dataset, the design of the reverse geocoding API, the implementation of KD-Tree, BallTree, Geohash and H3 indexing methods, and the evaluation procedure used to compare their performance. Section 3 presents the experimental results, including dataset characteristics, index preparation time, API query latency, nearest spatial group agreement with the brute-force haversine baseline, and additional retrieval distance. Section 4 discusses the findings, the practical suitability of the proposed service for local embedded computing and Malaysia-based location-aware applications and the limitations of using point-based location-reference data. Finally, Section 5 concludes the article and identifies possible directions for future system improvement.

2. Methodology

This study comprises four stages: dataset preparation, open web API design, spatial indexing method implementation, and evaluation. First, the Malaysian location-reference dataset was cleaned, validated, and reorganised into spatial groups to address repeated coordinates associated with multiple location descriptions. Second, a platform-neutral reverse geocoding API was designed to accept coordinate queries and return the nearest available location-description candidates through standard web responses. Third, four spatial indexing methods, namely KD-Tree, BallTree, Geohash, and H3, were implemented for nearest spatial group retrieval. Finally, the implemented methods were evaluated using query latency, agreement with a brute-force haversine baseline, additional retrieval distance, and index preparation time. Figure 1 summarises the overall methodology

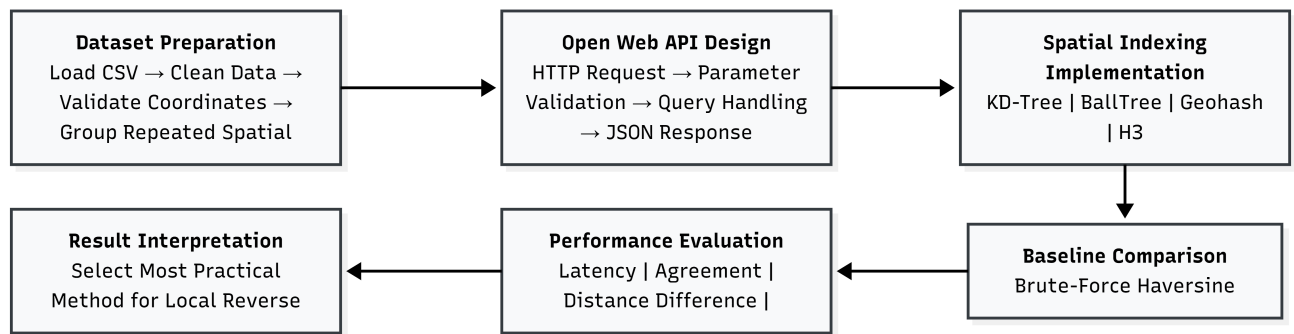


Fig. 1. Proposed methodology

2.1 Dataset Preparation

The dataset used in this study was stored in a comma-separated values file named *daterah-working-set.csv*. The dataset contained 54,357 rows of Malaysian location-reference records. Each record consisted of six fields: States, District, Town, Lat, Lon, and GeoHash. The States field represented the state label of the recorded location, while the District field represented the supplied local area label. The Town field contained the location description to be returned by the reverse geocoding service. The Lat and Lon fields stored the geographic coordinates of the reference record, and the GeoHash field contained an existing five-character spatial encoding of the coordinate.

A sample of the dataset is shown in Table 2. The records indicate that the dataset includes location references from different Malaysian states and local areas, including Sik in Kedah and Petaling Jaya in Selangor.

Table 1

The CSV file dataset structure

Field	Data Type	Description
States	Text	The state in Malaysia
District	Text	The name of the district in the state
Town	Text	Location description, such as town, village, road, housing area, or local place name
Lat	Numeric	Latitude of the reference point in decimal degrees
Lon	Numeric	Longitude of the reference point in decimal degrees
GeoHash	Text	Five-character Geohash value

Table 2

Sample records from the location referenced dataset

States	District	Town	Lat	Lon	Geohash
Kedah	Sik	Sik	5.872871	100.6569	w1p96
Kedah	Sik	Sik	5.983500	100.6944	w1pd5
Selangor	Petaling Jaya	Kelana Jaya	3.113067	101.588063	w281x
Selangor	Petaling Jaya	Padang Golf	3.107094	101.608255	w2838
Johor	Pontian	Taman Nilam	1.576929	103.283876	w232y
Perak	Ulu Kinta	Taman Bagus	4.680784	101.102318	w0zfs

The dataset preparation consisted of four operations: loading the CSV file, cleaning the data, validating the coordinate fields, and grouping repeated spatial points. First, the dataset was loaded into the application environment using a tabular data-processing procedure. The total number of loaded rows was checked against the expected total of 54,357 records to confirm that the input file had been read completely.

Second, the textual fields were inspected for missing values, inconsistent spacing, malformed values, and possible label truncation. This step was required because location-reference data may contain inconsistent textual entries. For example, a malformed state label such as “angor” may indicate a truncated representation of “Selangor”. Such records should not be corrected automatically without verification. Instead, suspicious state and area labels should be reviewed against surrounding records or an authoritative reference before standardisation. Leading and trailing whitespace should also be removed from the States, District, Town, and GeoHash fields.

Third, the latitude and longitude values were validated before spatial indexing. The Lat and Lon fields were converted into numeric values, and rows containing missing or non-numeric coordinate values were removed. Geographic range validation was performed to ensure that latitude values remained within -90 to 90 degrees and longitude values remained within -180 to 180 degrees. Since the proposed service is intended for Malaysian location retrieval, records outside the spatial extent represented by the Malaysian dataset should also be inspected as possible data errors. The existing GeoHash value was retained for the Geohash implementation and may be checked against regenerated Geohash values during validation.

Fourth, repeated spatial points were grouped before the construction of the indexing structures. This procedure was necessary because different location descriptions may share the same latitude and longitude. If such records were indexed separately, the API could return one arbitrary textual description even when several descriptions were spatially identical. To avoid this problem, records sharing the same state label, area label, latitude, longitude, and Geohash value were assigned to a single spatial group. Each group was given a generated identifier, referred to as `spatial_group_id`, while all associated Town descriptions were retained as candidate outputs.

2.2 Open Web API Design

The reverse geocoding service was designed as a platform-neutral web API based on standard HTTP communication and structured response representation. The service follows a RESTful retrieval model in which a client submits a geographic coordinate to an endpoint and receives the nearest location-description candidate set from the prepared spatial dataset.

The client may operate on the same host machine as the API, within a local network, or on an edge device that communicates with the local server. This design separates the location retrieval function from the client implementation and allows the same service to be consumed by web dashboards, mobile applications, embedded gateways, IoT controllers or command-line applications. The API processing flow is shown in Figure 2.

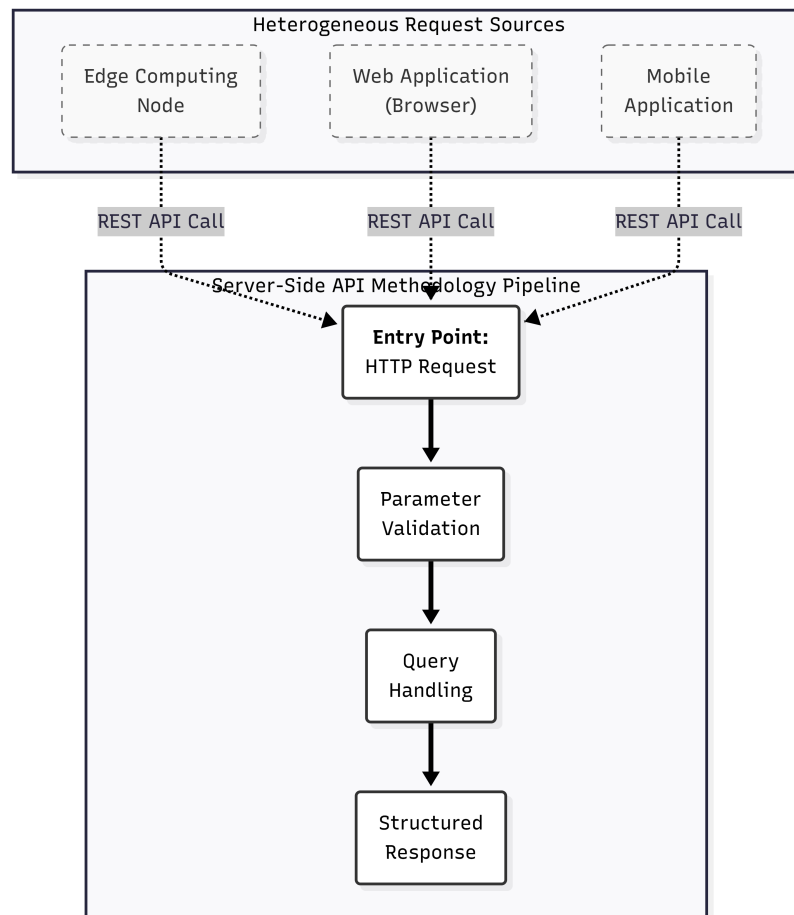


Fig. 2. Interaction model between heterogeneous client sources and the proposed Reverse Geocoding Web Service API

2.2.1 HTTP Request Design

The API exposes a versioned endpoint for reverse geocoding queries. A version identifier is included in the endpoint path to allow future changes to the response structure or retrieval procedure without affecting existing clients.

The proposed endpoint is defined in Figure 3.

```
GET /api/v1/reverse-geocode?lat={latitude}&lon={longitude}&method={method}
```

Fig. 3. Proposed RESTful endpoint for returning location name

The endpoint uses GET method as the operation retrieves location result from existing reference data without modifying the server-side dataset. This is in accordance with stateless communication mechanism that can be accessed by different client platforms through a common request format

The full HTTP request parameters are presented in Table 3.

Table 3

Sample records from the location referenced dataset

Parameter	Requirement	Description
lat	Text	Latitude in decimal degrees
lon	Text	Longitude in decimal degrees
method	Text	One of the spatial retrieval method: kdtree, balltree, geohash or h3
format	Numeric	Response format – json or cbor
limit	Numeric	Maximum number of returned location candidates

During the experimental stage, the method parameter allows the same coordinate to be evaluated using all four spatial indexing methods. In an operational deployment, the client does not need to select the search method. Instead, the API may use the method selected from the evaluation as its default internal retrieval procedure.

A client may originate from the localhost itself or from a separate edge device. For example, a dashboard application running on the same server may request locality information through localhost, while a GPS-enabled gateway or embedded monitoring node may submit coordinates through a local network connection. In both cases, the client follows the same endpoint contract and does not need direct access to the CSV dataset or the spatial index.

The service may also perform an operational coverage check to determine whether the query coordinate falls within the general geographic extent represented by the Malaysian dataset. However, this check is not equivalent to official boundary validation because the study uses point-based location records rather than polygon-based national or administrative boundaries.

2.2.2 Response Representation and Error Handling

The API returns the reverse geocoding result in a structured response format. JSON is used as the primary response representation because it is widely supported by web, mobile, server-side and embedded gateway applications. A successful response contains the query coordinate, the spatial retrieval method, the nearest spatial group, the calculated distance and the associated location-description candidates.

An example JSON response is shown in Figure 4. The API returns standard HTTP status responses so that client devices can distinguish successful retrieval from invalid input or service failure. Error responses are represented in JSON during the main implementation. Table 4 outlines the HTTP response status used by the API.

```
{
  "query": {
    "latitude": 5.976500,
    "longitude": 116.115800
  },
  "method": "balltree",
  "nearest_spatial_group": {
    "state": "Sabah",
    "area_label": "Kota Kinabalu",
    "reference_latitude": 5.976474,
    "reference_longitude": 116.115777,
    "distance_km": 0.004
  },
  "location_candidates": [
    "Bangunan Pasar Besar Kota Kinabalu",
    "Jalan Pusat Pembangunan Masyarakat"
  ]
}
```

Fig. 4. Proposed JSON response

Table 4

HTTP response status used by the API

HTTP Status	Condition	Description
200 OK	Valid Query and result retrieved	The nearest spatial group and location candidates are returned
400 Bad Request	Missing or malformed request parameters	The client request is incomplete or invalid
422 Unprocessable Content	Valid coordinate syntax but outside operational coverage	The request cannot be processed within the configured Malaysian dataset scope
500 Internal Server Error	Dataset or spatial index failure	The service cannot complete the retrieval operation

2.3 Spatial Indexing Implementation and Baseline Comparison

Following dataset preparation, the unique spatial groups were used as the search units for reverse geocoding. Each group retained its spatial_group_id, state label, area label, reference coordinate, Geohash value, and associated location-description candidates. This arrangement prevented repeated coordinates from being indexed as separate geographic targets while preserving the textual descriptions linked to each point.

Four spatial indexing methods were implemented: KD-Tree, BallTree, Geohash, and H3. Each method received the same latitude and longitude query and returned the nearest available spatial group. The location descriptions associated with the retrieved group were then included in the API response. A brute-force haversine procedure was implemented separately as the reference method for comparing retrieval correctness.

2.3.1 KD-Tree Retrieval

The KD-Tree method represents the unique spatial groups in a tree structure for nearest-neighbour retrieval. Since raw latitude and longitude values do not directly represent distance on the Earth's surface, each coordinate was converted into a three-dimensional unit-sphere representation before index construction. The transformed coordinate values were defined in Eq (1).

$$\begin{aligned}x &= \cos(\phi)\cos(\lambda) \\ y &= \cos(\phi)\sin(\lambda) \\ z &= \sin(\phi)\end{aligned}\tag{1}$$

where ϕ is latitude in radians and λ is longitude in radians.

For each incoming query, the latitude and longitude values were converted using the same transformation. The KD-Tree then returned the nearest indexed spatial group based on the transformed coordinate space. The final geographic distance between the query and the retrieved reference point was calculated using the haversine formula before the result was returned through the API.

2.3.2 BallTree Retrieval

The BallTree method was implemented using latitude and longitude values converted into radians. Unlike the KD-Tree implementation, BallTree supports nearest-neighbour retrieval using the haversine distance metric directly. Therefore, it provides a suitable point-based method for searching geographic coordinates.

During initialisation, one radian-coordinate pair was prepared for each unique spatial group and stored in the BallTree index. For each query request, the input coordinate was converted into radians and submitted to the index for nearest-neighbour retrieval. The angular distance returned by the haversine search was converted into kilometres by multiplying the result by the mean Earth radius. The matched spatial group was then used to retrieve all location-description candidates associated with the selected coordinate.

BallTree was included in the comparative evaluation because it provides an indexed search method that applies the same spherical-distance basis as the brute-force haversine reference used in this study. This permits direct evaluation of query latency and retrieval agreement without requiring a separate Euclidean-distance approximation during candidate selection. Previous geographic applications have also applied BallTree for nearest-location retrieval, including public facility candidate identification and nearest trauma-centre selection from geographic coordinate data [4], [6].

2.3.3 Geohash Retrieval

The Geohash method used the precalculated five-character GeoHash field stored in the dataset. Geohash converts latitude and longitude coordinates into hierarchical alphanumeric spatial cell identifiers, where the character length determines the spatial precision of the encoded region [9]. During initialisation, the unique spatial groups were organised in a lookup structure according to their stored Geohash values. This design is consistent with previous applications of Geohash codes

as spatial indexing keys for geographic data retrieval and location-aware monitoring records [10], [11].

For each query, the input coordinate was encoded using the same five-character Geohash precision as the prepared dataset. Candidate spatial groups were first retrieved from the matching Geohash cell. Adjacent cells were also included because geographically close points may be assigned to different Geohash cells when they occur near a cell boundary [12]. When the initial candidate area contained no spatial group, the search area was expanded until at least one candidate group was available. This expansion procedure is an implementation decision of the present study and is evaluated through retrieval agreement and query-latency measurements.

After candidate generation, the retrieved spatial groups were ranked using haversine distance, and the nearest group was selected. This refinement step is necessary because Geohash matching identifies spatially bounded candidates but does not by itself guarantee the geographically nearest coordinate. The use of Geohash for candidate filtering followed by a distance-based refinement stage is consistent with previous Geohash-based nearest-neighbour and spatial filtering procedures [12], [13]. Haversine distance was used as the final geographic-distance calculation because it measures great-circle separation between latitude and longitude coordinate pairs on a spherical Earth model [14].

2.3.4 H3 Retrieval

In the H3 retrieval method, each unique spatial group was assigned a hierarchical hexagonal cell identifier. The latitude and longitude of each spatial group were converted into an H3 cell at a fixed resolution, and the resulting cell identifiers were used to organise the groups in a lookup structure [15].

For an incoming coordinate, the API generated the corresponding H3 cell and retrieved candidate spatial groups from the query cell and its surrounding cells. When no candidate was found, the neighbourhood search was expanded. The retrieved candidates were then ranked using haversine distance to identify the nearest spatial group [14], [15].

H3 was included because its hierarchical cell representation can support future area-based summaries, monitoring zones, and local geographic aggregation [15], [16]. The selected H3 resolution is reported as part of the experimental settings. H3 was included because its cell-based representation may support later extension of the service for area-based summaries, monitoring zones or local geographic aggregation. The H3 resolution used in the implementation should be reported together with the experimental settings.

2.3.5 Brute-Force Haversine Baseline

A brute-force haversine search was implemented as the reference procedure for evaluation. Unlike the four indexed methods, the baseline did not reduce the candidate search space. For every query coordinate, it calculated the haversine distance to all unique spatial groups and returned the group with the minimum distance.

The haversine distance between a query point and a reference point is shown in Eq. 2

$$d = 2R \arcsin \left(\sqrt{\sin^2 \left(\frac{\Delta \phi}{2} \right) + \cos(\phi_1) \cos(\phi_2) \sin^2 \left(\frac{\Delta \lambda}{2} \right)} \right) \quad (2)$$

where d is the geographic distance, R is the Earth radius, ϕ_1 and ϕ_2 are the latitudes in radians, and $\Delta\phi$ and $\Delta\lambda$ are the differences in latitude and longitude.

The Brute-force Haversine served as the reference output for determining whether the indexed methods retrieved the same nearest spatial group and whether any alternative result introduced additional retrieval distance.

2.3.6 Common Retrieval Output

To ensure a consistent comparison, all spatial indexing methods returned the same result structure. The retrieved output consisted of the nearest spatial_group_id, state label, area label, reference latitude and longitude, calculated distance, and the list of associated location-description candidates. Only the spatial search procedure differed between methods.

Table 5

Summary of implemented retrieval methods

Method	Search Representation	Candidate Retrieval Procedure
KD-Tree	Three-dimensional unit coordinates	Nearest-neighbour tree query
BallTree	Latitude/longitude in radians	Haversine-based tree query
Geohash	Precalculated five-character spatial cells	String-based neighbouring cell lookup
H3	Hierarchical hex cells	Neighboring cell lookup
Brute-force Baseline	All unique spatial points	Exhaustive calculation and comparison

Each method was subsequently compared with the brute-force baseline using query latency, nearest spatial group agreement, additional retrieval distance, and index preparation time. This was used to determine which retrieval method was most suitable for the proposed local reverse geocoding web service.

2.4 Performance Evaluation

The implemented spatial retrieval methods were evaluated using a common set of coordinate queries and the same web API workflow. The evaluation compared KD-Tree, BallTree, Geohash and H3 against the brute-force haversine procedure. Since repeated textual descriptions may share the same coordinate, the evaluation was conducted at the spatial_group_id level rather than by comparing individual Town values.

Three performance measures were used: query latency, nearest spatial group agreement and distance difference. Query latency measures the practical response speed of the API. Agreement measures whether an indexed method retrieves the same nearest spatial group as the exhaustive baseline. Distance difference measures the geographic effect when an indexed method retrieves a different spatial group from the baseline.

A query set of 1,000 coordinates was prepared to represent different retrieval conditions. The query coordinates consisted of existing coordinates obtained from the prepared spatial groups, coordinates with small positional disturbance, and randomly generated coordinates within the spatial extent of the dataset. The composition of the query set used in evaluation is shown in Table 6.

Table 6
Composition of the evaluation query set

Query Category	Number of Queries	Purpose
Existing spatial group coordinates	400	To verify retrieval for locations already represented in the dataset
Noisy coordinates near existing spatial groups	400	To simulate coordinate variation from GPS-enabled or embedded devices
Random coordinates within dataset extent	200	To assess general nearest-location retrieval behaviour

2.4.2 Evaluation Metrics

The performance of KD-Tree, BallTree, Geohash and H3 was evaluated against the brute-force haversine baseline using three metrics: query latency, nearest spatial group agreement and distance difference. The brute-force method was treated as the reference because it calculates the distance from each query coordinate to all available spatial groups before selecting the nearest result.

Query latency measures the processing time required for the API to retrieve a location result. The timing began after request validation and ended after the nearest spatial group and its associated candidate descriptions were prepared for response. Before the recorded test, warm-up queries were performed to ensure that the dataset and indexes were already loaded into memory. Latency was reported using mean, median and 95th percentile processing time in milliseconds.

Nearest spatial group agreement measures whether an indexed method returns the same spatial_group_id as the brute-force baseline. Evaluation was performed at the spatial group level because several location descriptions may share the same coordinate. The agreement percentage was calculated as in Eq. (3)

$$Agreement(\%) = \frac{No. of matching spatial groups}{Total queries} \times 100 \quad (3)$$

Distance difference measures the geographic deviation when an indexed method returns a different spatial group from the baseline. For each query, the distance returned by the indexed method was compared with the nearest distance produced by the brute-force method. The formula is shown in Eq. (4).

$$\Delta d_i = d_{i,method} - d_{i,baseline} \quad (4)$$

The result was reported in metres. A value of zero indicates that the indexed method returned a spatial group at the same nearest distance as the baseline, while a larger positive value indicates less accurate spatial retrieval.

For each of the 1,000 query coordinates, the brute-force baseline was executed first, followed by KD-Tree, BallTree, Geohash and H3. The returned spatial group, distance and query latency were recorded for each method. The indexed results were then compared with the baseline to calculate agreement and distance difference. This evaluation procedure measures whether the indexed methods provide faster retrieval while maintaining results close to exhaustive geographic-distance search. A summary of the evaluation metrics is shown in Table 7.

Table 7

Summary of the evaluation metrics used in the study

Query Category	No. of Samples	Purpose
Query Latency	400	To verify retrieval for locations already represented in the dataset
Nearest Spatial Group agreement	400	To simulate coordinate variation from GPS-enabled or embedded devices
Distance Differences	200	To assess general nearest-location retrieval behaviour

3. Results and Discussions

The retrieval performance of the proposed reverse geocoding methods was evaluated using 1,000 coordinate queries. The evaluation set consisted of 400 existing reference coordinates, 400 noisy coordinates derived from known spatial points, and 200 nearby-random coordinates. The brute-force haversine method was used as the baseline for determining the nearest spatial group. KD-Tree, BallTree, Geohash and H3 were then compared against the baseline using retrieval latency, spatial group agreement and distance difference.

3.1 Overall Retrieval Performance

Table 8 presents the overall performance of the five retrieval procedures. All four indexed methods reduced the mean retrieval latency compared with brute-force haversine search.

Table 8

Overall retrieval performance for 1,000 coordinate queries

Method	Mean Latency (ms)	Median Latency (ms)	95th Percentile Latency (ms)	Agreement (%)	Mean Distance Diff. (m)	Max Distance Diff. (m)
Baseline	0.302	0.296	0.335	100	0	0
KD-Tree	0.145	0.142	0.166	100	0	0
BallTree	0.135	0.132	0.156	100	0	0
Geohash	0.11	0.103	0.149	96.0	22.854	3693.121
H3	0.115	0.105	0.158	97.8	11.828	4588.811

Geohash recorded the lowest mean latency at 0.111 ms, followed by H3 at 0.115 ms, BallTree at 0.135 ms and KD-Tree at 0.145 ms. The brute-force baseline recorded a mean latency of 0.302 ms.

Geohash and H3 produced the lowest mean latency values. Geohash reduced mean latency by 63.4%, while H3 reduced it by 62.0% compared with brute-force search. However, both cell-based methods returned non-matching nearest spatial groups for some queries. Geohash achieved 97.0% agreement, corresponding to 30 non-matching results from 1,000 queries. H3 achieved 97.8% agreement, corresponding to 22 non-matching results.

The latency advantage of Geohash and H3 over BallTree was small in absolute terms. Geohash was only 0.025 ms faster than BallTree on average, while H3 was only 0.021 ms faster. In contrast, both methods introduced measurable spatial deviation. Geohash recorded a maximum distance difference of 3693.121 m, while H3 recorded a maximum difference of 4588.811 m. For nearest location-description retrieval, deviations above 3 km or 4 km are not acceptable when a fully consistent method is available with only a small latency increase.

3.2 Performance According to Query Type

Table 9 and Table 10 presents the results according to the three query categories. The category-level comparison shows that the retrieval behaviour differed according to whether the query matched an existing coordinate or nearby coordinates not stored in the dataset.

Table 9
Retrieval performance according to query category

Method	Mean Latency (ms)		
	Existing	Noisy	Nearby random
Baseline	0.302	0.301	0.299
KD-Tree	0.147	0.146	0.142
BallTree	0.138	0.136	0.131
Geohash	0.111	0.112	0.107
H3	0.115	0.116	0.111

Table 9
Retrieval performance according to query category

Method	Agreement (%)			Mean Distance Diff. (m)			Max Distance Diff. (m)		
	Existing	Noisy	Nearby random	Existing	Noisy	Nearby random	Existing	Noisy	Nearby random
Baseline	100	100	100	0	0	0	0	0	0
KD-Tree	100	100	100	0	0	0	0	0	0
BallTree	100	100	100	0	0	0	0	0	0
Geohash	100	97	97.5	0	30.277	53.716	0	3551.12	3693.12
H3	100	91	94	0	4.248	50.645	0	494.59	4588.81

For the 400 existing coordinate queries, all methods achieved 100% agreement and zero distance difference. This indicates that each method can retrieve the expected spatial group when the query coordinate is already represented in the indexed dataset. Under this condition, Geohash produced the lowest mean latency at 0.111 ms, while H3 recorded 0.115 ms.

The difference between methods became apparent when the query coordinate did not directly correspond to an indexed point. For the 400 noisy queries, KD-Tree and BallTree maintained 100% agreement with the brute-force baseline. In contrast, Geohash agreement decreased to 97.0%, with a mean distance difference of 30.277 m and a maximum difference of 3551.123 m. H3 achieved 97.5% agreement, with a lower mean distance difference of 4.248 m and a maximum difference of 494.587 m. Therefore, H3 handled the noisy-coordinate condition better than Geohash, although it did not match the complete agreement obtained by the tree-based methods.

The weakest results for the cell-based methods occurred in the nearby-random query category. Geohash agreement decreased to 91.0%, meaning that 18 of the 200 queries returned a spatial group different from the brute-force baseline. Its mean distance difference increased to 53.716 m, with a maximum deviation of 3693.121 m. H3 performed better than Geohash with 94.0% agreement, but it still returned 12 non-matching results. Its mean distance difference was 50.645 m, while its maximum difference reached 4588.811 m.

These findings show that Geohash and H3 perform correctly for stored coordinates but become less reliable when the query falls near spatial cell boundaries or outside the immediate

candidate region represented by the query cell and its searched neighbours. The use of cell-based indexing therefore introduces a trade-off between very low latency and nearest-point reliability.

3.3 Comparison Between Tree-Based and Cell-Based Retrieval Algorithm

The results separate the implemented methods into two practical groups. KD-Tree and BallTree behaved as accurate nearest-neighbour retrieval methods, while Geohash and H3 behaved as faster but less consistent cell-based candidate retrieval methods.

KD-Tree achieved complete agreement with the brute-force baseline. The coordinate transformation into unit-sphere Cartesian values allowed the tree to preserve the nearest spatial group for all evaluated queries. However, KD-Tree recorded a higher mean latency than BallTree and requires an additional coordinate transformation step before index construction and querying.

BallTree achieved the strongest overall result for direct reverse geocoding retrieval. It recorded 100% agreement with the brute-force baseline, zero mean and maximum distance difference, and a mean retrieval latency of only 0.135 ms. Since BallTree operates using haversine distance on latitude and longitude values converted into radians, its search procedure is directly aligned with the geographic distance measure used by the baseline.

Geohash was the fastest method overall, recording a mean latency of 0.111 ms. Its stored five-character cell identifiers are also convenient because the dataset already contains Geohash values. However, its reduced agreement for noisy and nearby-random queries indicates that the Geohash cell retrieval procedure is not sufficiently reliable for final nearest-location selection under the evaluated configuration. The maximum deviation of more than 3.6 km is particularly problematic for a service intended to return the nearest available location-description candidate.

H3 performed slightly slower than Geohash but achieved better overall agreement and lower mean distance difference. It was more stable than Geohash for noisy coordinates, where its maximum deviation remained below 500 m. Nevertheless, its behaviour for nearby-random coordinates remained unsuitable for direct nearest-point retrieval because its maximum deviation exceeded 4.5 km. H3 may still be appropriate for later functions involving spatial zones, local aggregation or monitoring-region summaries, but it is not the strongest method for the primary reverse geocoding operation in this study.

3.4 Practical Implications for Local Reverse Geocoding

The results confirm that the proposed CSV-based reverse geocoding approach is computationally feasible for local deployment. Even the brute-force baseline completed a retrieval in approximately 0.302 ms on average under the evaluation environment. This indicates that the prepared location-reference dataset is manageable for in-memory search. Nevertheless, BallTree reduced the retrieval time by more than half while preserving the same output as exhaustive haversine search.

For local Malaysian applications, the selected method enables incoming coordinates to be translated into the nearest available location-description candidate set without repeated calls to an external reverse geocoding provider. This is relevant for embedded gateways, local IoT monitoring, institutional asset systems, agricultural field applications, local reporting systems and dashboard services.

The candidate-based response design remains necessary. The dataset contains several textual location descriptions associated with identical coordinates. Therefore, even when BallTree identifies the correct nearest spatial group, the system must return the descriptions linked to that group rather

than claim that a single exact address has been identified. This response behaviour is consistent with the actual structure of the location-reference data.

The proposed system also has practical value in a local closed-loop configuration. A device or edge gateway can submit a coordinate to a locally hosted service, retrieve a readable location reference and use the result for display, event logging or local decision processing. This limits dependence on external service availability and avoids recurring third-party lookup charges for repeated coordinate translation.

4. Conclusion, Recommendations and Future Works

This study addressed three implementation objectives for a local reverse geocoding service using Malaysian location-reference data. First, the issue of repeated coordinates with multiple location descriptions was addressed by grouping identical spatial points and returning their associated candidate descriptions. This approach allows the system to represent the available data correctly without claiming exact address identification.

The study examined whether spatial indexing could reduce retrieval cost while preserving agreement with brute-force haversine search. Based on 1,000 coordinate queries, KD-Tree and BallTree achieved 100% agreement with the baseline and recorded zero distance difference. BallTree reduced mean retrieval latency from 0.302 ms for brute-force search to 0.135 ms.

The KD-Tree, BallTree, Geohash and H3 has been compared to identify the most practical method for the retrieval component of a lightweight web API. Although Geohash and H3 recorded slightly lower latency, both produced non-matching results and kilometre-scale maximum distance differences. BallTree was therefore selected as the preferred method because it combined low retrieval latency with complete agreement against the geographic-distance baseline.

Overall, the study provides a workable design for translating Malaysian coordinates into nearest location-description candidate sets. The findings support the use of spatial grouping for ambiguous coordinate records and BallTree for reliable local retrieval within the proposed reverse geocoding service.

Future work should evaluate the complete containerised API under realistic request loads and local edge-device use. The dataset should also be checked against authoritative location or boundary data to improve geographic validity. Real GPS coordinates collected in Malaysian urban and less dense environments should be used to confirm retrieval behaviour beyond generated test queries. Geohash and H3 may also be retained for later area-based grouping or monitoring functions. Despite its limitations, the study provides a workable local reverse geocoding design for embedded monitoring and location-aware systems in Malaysia.

Acknowledgement

The authors would like to thank the colleagues and administrative staff of Universiti Teknologi MARA (UiTM), Malaysia, for their support and assistance during the preparation of this study. This research did not receive any specific grant from public, commercial or non-profit funding agencies.

Conflict of Interest Statement

The authors declare that there is no conflict of interest regarding the publication of this paper. No financial support, grants, or other forms of compensation were received that could have influenced the outcomes of this work. The research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest”

Author Contributions Statement

Mohammad Hafiz bin Ismail: Conceptualization, Methodology, Software, Formal analysis, Investigation, Project administration, Writing, original draft

Tajul Rosli Razak: Conceptualization, Methodology, Supervision, Writing, review & editing.

Mohd Faris Mohd Fuzi: Investigation, Validation, Data curation, Writing

Ethics Statement

This study was conducted in accordance with the ethical standards of the institutional research committee. The research involved the development and evaluation of a reverse geocoding retrieval system using location-reference data. It did not involve human participants, animal subjects, human biological materials, or questionnaire instruments involving human respondents. Therefore, informed consent was not applicable, and ethical approval was not required for the scope of this study.

References

- [1] Z. Yin, D. W. Goldberg, T. A. Hammond, C. Zhang, A. Ma, and X. Li, "A probabilistic framework for improving reverse geocoding output," *Transactions in GIS*, vol. 24, no. 3, pp. 656–680, 2020.
- [2] Y. Yin *et al.*, "Learning multi-context aware location representations from large-scale geotagged images," in *Proceedings of the 29th ACM International Conference on Multimedia*, 2021, pp. 899–907.
- [3] A. Di Lorenzo, L. Di Renzo, C. Profico, D. Profico, V. Olivieri, and S. Guccione, "GeoCetus: A Multi-Decadal Open Geospatial Infrastructure for the Continuous Monitoring of Marine Strandings in Italy," *Animals*, vol. 16, no. 9, 2026, doi: 10.3390/ani16091323.
- [4] B. Hanterdsith, "Utilizing machine learning and geographic analysis to improve Post-crash traffic injury management and emergency response systems," *Int. J. Inj. Contr. Saf. Promot.*, vol. 32, no. 1, pp. 108–117, 2025, doi: 10.1080/17457300.2025.2487632.
- [5] M. Hysenaj, "From 'here' to Nearby: A Python-Spring Boot Pipeline for Smartphone Geolocation, Reverse Geocoding, and Real-Time Place Discovery in Web GIS," in *2025 6th International Conference on Communications, Information, Electronic and Energy Systems, CIEES 2025 - Conference Proceedings*, Institute of Electrical and Electronics Engineers Inc., 2025. doi: 10.1109/CIEES66347.2025.11300157.
- [6] H. Ramadhan, "Data Collection for Nearest Public Facility Using Ball Tree Algorithm and Google Maps API," in *Proceedings of The International Conference on Data Science and Official Statistics*, 2025, pp. 1187–1199.
- [7] J. Yu, W. Zhang, and R. Chen, "Research on Offline Reverse Geocoding Algorithm Based on k-d Tree," in *Proceedings - 2021 International Conference on Computer Information Science and Artificial Intelligence, CISAI 2021*, Institute of Electrical and Electronics Engineers Inc., 2021, pp. 548–552. doi: 10.1109/CISAI54367.2021.00111.
- [8] K. Horvath, H. Wöllik, U. Christoph, and V. Egger, "Location-Based Service Discovery for Mobile-Edge Computing Using DNS," *Lecture Notes in Networks and Systems*, vol. 447, pp. 379–388, 2023, doi: 10.1007/978-981-19-1607-6_33.
- [9] Consumer Technology Association, "Fast and Readable Geographical Hashing," 2023, *Consumer Technology Association*.
- [10] L. Yan, K. Dongho, and B.-S. Shin, "Geohashed Spatial Index Method for a Location-Aware WBAN Data Monitoring System Based on NoSQL," *Journal of Information Processing Systems*, vol. 12, Jun. 2016, doi: 10.3745/JIPS.04.0025.
- [11] Jiajun Liu, Haoran Li, Yong Gao, Hao Yu, and Dan Jiang, "A geohash-based index for spatial data management in distributed memory," in *2014 22nd International Conference on Geoinformatics*, IEEE, Jun. 2014, pp. 1–4. doi: 10.1109/GEOINFORMATICS.2014.6950819.
- [12] Lihao Ni, Yanshen Liu, and Yi Liu, "Privacy Protection Model for Location-Based Services," *Journal of Information Processing Systems*, vol. 16, no. 1, pp. 96–112, Feb. 2020.

- [13] V. M, A. P. Kesarkar, and D. Lopez, "High-performance implementation of a two-bit geohash coding technique for nearest neighbor search," *Concurr. Comput.*, vol. 33, no. 5, Mar. 2021, doi: 10.1002/cpe.6029.
- [14] R. Sinnott, "Virtues of the Haversine," *Sky Telescope*, vol. 68, no. 1, Aug. 1984.
- [15] I. Brodsky, "H3: Uber's hexagonal hierarchical spatial index," Uber Engineering. Accessed: May 26, 2026. [Online]. Available: <https://www.uber.com/us/en/blog/h3/>
- [16] J. Bousquin, "Discrete Global Grid Systems as scalable geospatial frameworks for characterizing coastal environments," *Environmental Modelling & Software*, vol. 146, p. 105210, Dec. 2021, doi: 10.1016/j.envsoft.2021.105210.