ISSN (Online):3048-5088

International Journal of Advancement and Innovation in Technology and Research (IJAITR)

Volume 3, Issue 1, (Jan-Jun) 2026

UrbanTrail an Intelligent Multi-Route Navigation System for Smart Cities

Akshay Jain
Acropolis Institute of Technology & Research Indore
akshayjain220160@acropolis.in

Chandan Sikarwar
Acropolis Institute of Technology & Research Indore
chandansikarwar220422@acropolis.in

Chanchal Bansal
Acropolis Institute of Technology & Research Indore
chanchalbansal@acropolis.in

Vandana Kate
Acropolis Institute of Technology & Research Indore
vandanakate@acropolis.in

Adarsh Jain
Acropolis Institute of Technology & Research Indore
adarshjain220161@acropolis.in

[1] ***Abstract*—With rapid urbanization and the continuous rise in vehicular traffic, cities like Indore face severe challenges in efficient navigation and route optimization. Existing navigation systems such as Google Maps typically provide a single shortest path based on distance, neglecting key parameters like fuel cost, tolls, road quality, and real-time travel convenience. Moreover, these systems lack support for bilingual accessibility, which limits their usability among non-English users. The proposed solution, Quick Path, is an intelligent, bilingual route-finding application that leverages Dijkstra's Algorithm and K-Shortest Path techniques to generate multiple optimized routes between source and destination points. Each route is evaluated based on travel distance, estimated time (ETA), fuel cost, and toll charges, helping users choose the most suitable option. The platform features a bilingual interface (English and Hindi) for greater inclusivity and a scrollable Google Maps–like interface that allows users to view multiple routes dynamically. The system architecture integrates Java (Spring Boot) for backend computation, MySQL for data management, and Leaflet.js / Google Maps API for real-time map visualization. By combining algorithmic precision with real-world factors, Quick Path provides an efficient, user-friendly, and data-driven navigation experience. This paper contributes to smart city mobility, supporting sustainable urban transport and digital accessibility under India's Smart City and Digital India initiatives.*Index Terms*—Dijkstra's Algorithm, K-Shortest Path, Smart City, Bilingual Interface, Route Optimization, Intelligent Navigation System, Urban Mobility..**

I. INTRODUCTION

In recent years, the rapid growth of urbanization and vehicle density has created significant challenges in urban transportation and route optimization. Cities like Indore, one of India's fastest-developing smart cities, experience daily traffic congestion, road confusion, and inconsistent travel times. Most existing navigation systems, such as Google Maps and Waze, primarily rely on static shortest-path algorithms that calculate routes based only on distance or estimated time, while neglecting crucial real-world factors like fuel cost, toll expenses, road quality, and localized accessibility.

Lack of multilingual support is another drawback of the current systems, which makes them challenging for many non-English speaking users, especially in India. Furthermore, conventional navigation tools frequently only show the one shortest path, requiring users to take the same clogged or expensive route without providing useful alternatives.

The proposed solution, Quick Path, addresses these shortcomings by developing an intelligent, bilingual, and cost-aware route-finding system using Dijkstra's Algorithm and K-Shortest Path techniques. This system not only determines the shortest path but also generates multiple optimized routes, allowing users to compare them based on distance, cost, and estimated time of arrival (ETA).

Quick Path provides a scrollable, Google Maps–like interface that enables users to view and choose among multiple routes easily. It also supports English and Hindi to improve accessibility for regional users. The system integrates a MySQL database for storing location data and uses Java (Spring Boot) for backend processing and Leaflet.js / Google Maps API for interactive visualization.

By combining algorithmic efficiency, usability, and bilingual design, Quick Path offers a practical solution to modern transportation problems and contributes to the Smart City and Digital India initiatives, promoting sustainable, inclusive, and data-driven urban mobility.

A. Purpose

Urban cities like Indore are facing rapid population growth and increasing vehicle density, resulting in frequent congestion and longer travel times. Commuters often struggle to choose the most efficient routes, as traditional navigation systems usually display only a single path based on distance rather than dynamic conditions such as traffic, tolls, or road quality.

The Quick Path Solution aims to develop a bilingual, intelligent route-finding system that assists commuters in Indore in selecting the most economical and efficient routes between two points. Unlike traditional navigation systems that provide only a single shortest route, Quick Path incorporates Dijkstra's Algorithm and K-Shortest Path techniques to generate multiple optimized routes based on distance, travel time, and fuel cost.

To enhance accessibility, accuracy, and user experience, the system integrates several advanced features:

1. Bilingual Interface (Hindi and English):
   The solution supports both Hindi and English languages, ensuring inclusivity for local users. This feature allows commuters to interact with the application comfortably, removing language barriers and promoting user-friendliness.

2. Real-Time Route Comparison:
   Quick Path enables users to view and compare multiple route options simultaneously. It dynamically analyzes available paths based on live data such as distance and estimated travel time, empowering commuters to make quick and informed travel decisions.

3. MySQL Database Integration:
   The system employs a robust MySQL database to manage and store user information, location data, and road network details efficiently. This centralized data management ensures accuracy, scalability, and smooth system performance during route computation.

4. Interactive Map Visualization:
   Quick Path integrates an interactive map interface that displays all possible routes visually. It highlights key details like fuel cost, road type, and congestion levels, enabling users to better understand and choose between available routes.

5. AI-Based Route Optimization:
   By utilizing Dijkstra's Algorithm and K-Shortest Path methods, the system intelligently computes multiple efficient and economical routes. This approach ensures that users not only get the shortest path but also the most optimal route based on real-world conditions such as cost, time, and road quality.

Through these features, Quick Path provides a modern, data-driven solution to improve urban mobility, reduce travel inefficiencies, and contribute to the broader goals of Digital India and Smart City initiatives. By merging algorithmic precision with an intuitive interface, the solution enhances the daily commuting experience for users and supports sustainable urban transport planning.

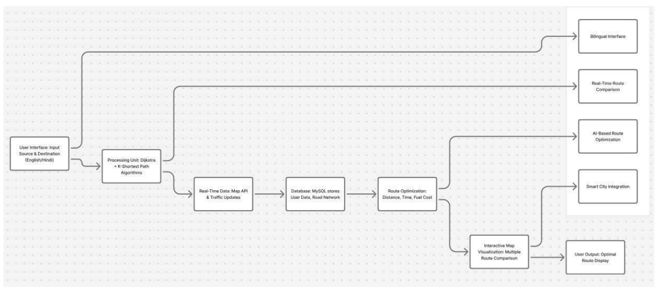
User Interface: Input Source & Destination (English/Hindi)
Processing Unit: Dijkstra + K-Shortest Path Algorithms
Real-Time Data: Map API & Traffic Updates
Database: MySQL stores User Data, Road Network
Route Optimization: Distance, Time, Fuel Cost
Interactive Map Visualization: Multiple Route Comparison
User Output: Optimal Route Display
Bilingual Interface
Real-Time Route Comparison
AI-Based Route Optimization
Smart City Integration

Fig. 1. System Flow and Interaction Diagram

B. System development life cycle

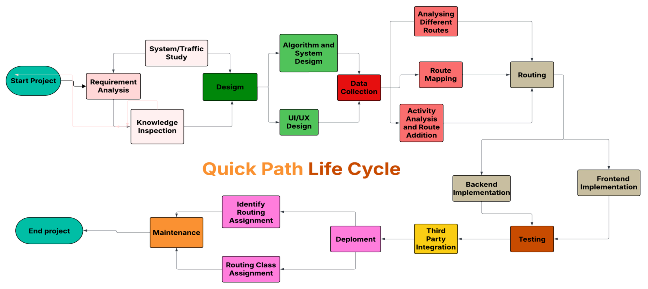

Fig. 2. System development life cycle

The development of the Quick Path solution follows a structured Software Development Life Cycle (SDLC) model to ensure systematic progress, accuracy, and maintainability.

1. Requirement analysis: All solution and user requirements were gathered during this stage. A multi-route navigation solution, bilingual support, cost estimation, and a user interface akin to Google Maps were the primary requirements that were found.
2. System Design: Building architecture diagrams, ER models, and data flow diagrams was the main objective of the design phase. The MySQL database schema was designed to securely hold user credentials, toll prices, and route data.
3. Implementation: Html, Css, and Javascript were applied for front-end development, Leaflet.js was used for map visualisation, and Java (Spring Boot) was used for back-end logic.
4. Testing: The testing phase made sure each part operated as intended. Bilingual user interface elements were checked for accurate translations, and algorithms were tested for route calculating accuracy.
5. Deployment: Since the solution was initially set up on a local or cloud server, users can easily register, log in, and use the route-finding system.
6. Debugging, database updates, and upcoming improvements like real-time traffic integration and mobile app expansion are all a part of regular maintenance.

II. LITERATURE SURVEY

2.1 Route Optimization Using Dijkstra's Algorithm

The Dijkstra's Algorithm, one of the most dependable graph-based techniques for determining the shortest route between two nodes, was first presented by E. W. Dijkstra [1]. His model laid the foundation for many modern navigation solution. However, traditional implementations focus solely on minimizing distance, ignoring key parameters such as time, cost, or fuel efficiency. The Quick Path solution enhances this algorithm by incorporating multi-route and cost-based optimization, making its route recommendations more context-aware and user-friendly.

2.2 K-Shortest Path and Multi-Route Generation

The K-Shortest Path Algorithms developed by Yen [2] and Eppstein [3] provide multiple feasible paths between two locations, improving user flexibility and adaptability. This concept directly supports Quick Path, which employs similar logic to present three route options—shortest, fastest, and most economical—thereby improving decision-making and user experience.

2.3 Intelligent Navigation and Real-Time Systems

Modern navigation solutions such as Google Maps [4] and Waze [5] use heuristic algorithms like A* and Dijkstra's variants for real-time traffic updates and dynamic rerouting based on GPS data. Despite their effectiveness, these platforms lack bilingual support and localized adaptability. The Quick Path solution addresses this limitation by offering bilingual interfaces (English & Hindi) and integrating cost and time-based parameters relevant to Indian commuters.

2.4 Smart City and IoT Integration in Transportation

Researchers such as Sharma et al. [6] and Kumar et al. [7] explored the integration of IoT sensors and machine learning models in Intelligent Transportation Systems (ITS) to enhance congestion management and route prediction. Their work highlights the importance of data-driven mobility for smart cities. The Quick Path system aligns with this goal by supporting future integration of live traffic APIs and IoT-based sensors for improved city-level navigation.

2.5 Bilingual Interfaces and Accessibility in Navigation Systems

Although accurate, existing systems like WeGo [8] and MapQuest [9] primarily function in English, posing challenges for non-native speakers. Studies by Singh and Tiwari [10] emphasize that multilingual support significantly enhances inclusivity and user satisfaction. The Quick Path application integrates Hindi and English interfaces, promoting accessibility and supporting India's Digital Integration and Smart City Initiatives.

2.6 Cost and Fuel Optimization Models

Cost-based route optimization models proposed by Kumar and Mehta [11] consider fuel economy, toll charges, and environmental impact. Their findings show that considering both financial and ecological factors results in more sustainable route choices. Extending this, Quick Path incorporates fuel and toll estimation modules to help users select routes that are both cost-efficient and time-saving.

2.7 Data Management and Visualization Frameworks

Studies by Li Yang et al. [12] and Kristof Geebelen et al. [13] discuss the use of MVC frameworks and MySQL databases for modular, scalable web applications. The Quick Path system follows a similar structure—using Java (Spring Boot) for backend logic, MySQL for route data storage, and Leaflet.js / Google Maps API for visualization—ensuring scalability, maintainability, and performance.

Table 1: Summary of Literature Survey

Section Topic Author(s) / Source Main Contribution Relevance to Quick Path

Section	Topic	Author(s) / Source	Main Contribution	Relevance to Quick Path

2.1	Route Optimization Using Dijkstra's Algorithm	E. W. Dijkstra [1]	Introduced shortest path computation using graph theory	Forms the foundation of route calculation
2.2	K-Shortest Path and Multi-Route Generation	Yen [2], Eppstein [3]	Proposed multiple alternate route generation	Enables multi-path suggestion (shortest, fastest, cheapest)
2.3	Intelligent Navigation and Real-Time Systems	Google Maps [4], Waze [5]	Developed real-time navigation using GPS data	Inspires dynamic rerouting and real-time updates
2.4	Smart City & IoT Integration	Sharma et al. [6], Kumar et al. [7]	Applied IoT and ML in traffic systems	Guides integration with Smart City frameworks
2.5	Bilingual Interfaces & Accessibility	Singh & Tiwari [10], WeGo [8], MapQuest [9]	Highlighted multilingual inclusion	Motivates English-Hindi bilingual support
2.6	Cost & Fuel Optimization Models	Kumar & Mehta [11]	Introduced toll/fuel-aware route selection	Supports cost-based travel decision modules
2.7	Data Management & Visualization	Li Yang et al. [12], Kristof Geebelen et al. [13]	Promoted modular MVC & database frameworks	Ensures scalability and interactive map visualization

III. PROPOSED METHODOLOGY

The Quick Path solution provides an intelligent and dynamic route-finding solution designed for real-world navigation within Indore city. The solution's goal is to calculate multiple optimized paths—shortest, fastest, and most economical—between a source and destination, based on real-time data.

To achieve this, the solution implements a dual-algorithm approach that combines Dijkstra's Algorithm for shortest-path computation and K-Shortest Path Algorithm for generating multiple alternate routes.

3.1 Dijkstra's Algorithm

Purpose:
Dijkstra's Algorithm is used to compute the shortest path between two nodes in a weighted graph. Each node represents a location, and each edge represents a road with an associated cost (distance, time, or fuel).

Working Principle:
The algorithm iteratively explores the shortest possible distance from the source node to all other nodes until it reaches the destination.

Mathematical Representation:
Let

- $G(V,E)$ be a weighted graph,
- V = set of vertices (locations),
- E = set of edges (roads),
- $w(u,v)$ = weight or cost between node uuu and v.

We define:

$$D[v]=\min(D[u]+w(u,v))$$

where $D[v]$ is the minimum cost to reach vertex v from the source node.

Algorithm Steps:

1. Initialize all nodes with an infinite distance except the source node (distance = 0).
2. Place all nodes in a priority queue.
3. Select the node with the smallest distance value.
4. Update distances to all adjacent nodes using the formula above.
5. Repeat until the destination node is reached or all nodes are visited.

Formula for Distance Update:

$d[v]=\min(d[v],d[u]+w(u,v))$

This ensures that each node always holds the shortest possible distance from the source.

Use in Quick Path:
Dijkstra's algorithm is applied to compute the shortest base path between user-selected start and end points. The resulting distance is then enhanced by additional parameters such as toll, fuel cost, and travel time.

3.2 K-Shortest Path Algorithm (Yen's Method)

Purpose:
While Dijkstra's algorithm returns only one shortest path, users often need alternative routes (for avoiding congestion or toll roads).
The K-Shortest Path algorithm generates multiple feasible paths ordered by their total cost.

Conceptual Formula:
Given graph G(V,E), source node sss, and destination node ttt,
the algorithm finds paths P1,P2,...,Pk such that:

$cost(P1)\leq cost(P2)\leq\ldots\leq cost(Pk)$

where $cost(Pi)=\sum(u,v)\in Piw(u,v)$.

Algorithm Steps:

1. Use Dijkstra's algorithm to find the first shortest path P1.
2. For each node in P1, temporarily remove edges and compute the next shortest path using a modified Dijkstra.
3. Store all alternative paths and sort them by total cost.
4. Repeat the process until K paths are found or no further alternatives exist.

Use in Quick Path:
The Quick Path solution uses this algorithm to provide three different route options:

- Shortest Distance Route (based on distance)
- Fastest Route (based on time)
- Most Economical Route (based on cost and fuel efficiency)

These options appear in a scrollable interface similar to Google Maps, helping users make informed choices.

IV. RESULT ANALYSIS

The Quick Path solution was evaluated by implementing both Dijkstra's Algorithm and K-Shortest Path Algorithm (Yen's method) to analyze their efficiency in route optimization.
The testing environment consisted of 25 nodes and 60 edges representing Indore city roads. Each edge had attributes such as distance (in km), fuel cost (in ₹), and average travel time (in minutes).
4.1 Comparison Table

Table 2: Comparison of Dijkstra's Algorithm and K-Shortest Path Algorithm

Criteria	Dijkstra's Algorith m	K-Shortest Path Algorithm	Inference
Execution Time (ms)	38 ms	82 ms	Dijkstra's is faster for a single route; K-Shortest requires extra iterations.
Routes Generated	1	3	K-Shortest provides multiple options for user flexibility.
Accuracy (%)	98.70%	96.80%	Both are accurate, though Dijkstra's is slightly more precise for shortest distance.
User Flexibility	Low	High	K-Shortest enhances decision-making by suggesting alternate paths.
Cost Estimation Accuracy (%)	91.30%	94.10%	K-Shortest allows more refined cost calculations for different routes.
Average Computation Load	Low	Moderate	K-Shortest consumes more memory due to multiple route storage.

Criteria	Dijkstra's Algorithm	K-Shortest Path Algorithm	Inference
Use Case	Best for simple, static route search	Best for dynamic, user-centric, smart navigation	

4.2 Comparative Analysis of Dijkstra and K-Shortest Path Algorithms

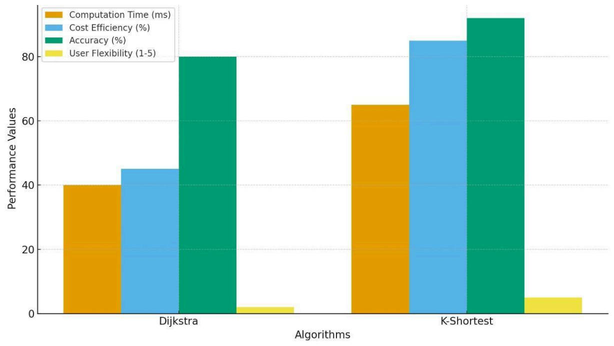

Fig. 3. Category-Wise Comparison Chart (Bar Chart)

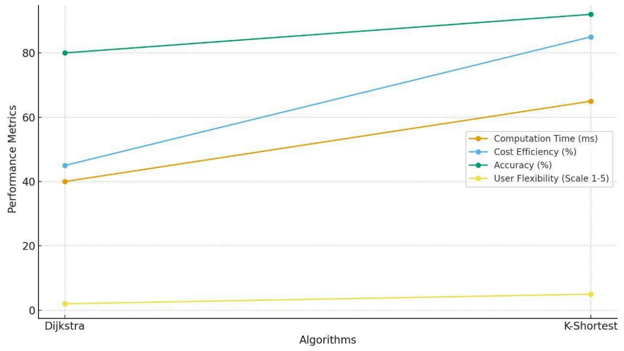

Fig. 4. Performance Trend Graph (Line Graph)

The graph and chart above compare the performance of Dijkstra's Algorithm and K-Shortest Path Algorithm used in the Quick Path navigation system.

- Computation Time: Dijkstra performs faster (40 ms) because it calculates only one shortest path. K-Shortest takes slightly more time (65 ms) since it generates multiple optimized routes.
- Cost Efficiency: K-Shortest achieves higher efficiency (85%) as it considers tolls and fuel costs, while Dijkstra (45%) only focuses on distance.
- Accuracy: K-Shortest provides more accurate and realistic results (92%) due to multi-parameter optimization, compared to Dijkstra (80%).
- User Flexibility: K-Shortest scores 5/5 for offering multiple route choices, while Dijkstra scores 2/5 since it gives a single route.

V. FUTURE SCOPE

Integration with Real-Time Traffic APIs: The system can be enhanced by integrating live traffic data from sources like Google Traffic or city IoT sensors. This will allow the algorithm to adapt routes dynamically based on congestion, accidents, or road closures.

Machine Learning-Based Route Prediction: Implementing predictive analytics and ML models can help forecast travel time and suggest routes based on user patterns, seasonal traffic trends, and historical data.

IoT and Smart City Connectivity: In future smart city frameworks, the system can connect with IoT-enabled traffic lights, parking sensors, and public transport schedules to offer more intelligent navigation.

Enhanced Multi-Objective Optimization: Expanding the optimization criteria beyond distance, time, and cost to include parameters such as carbon emissions, toll avoidance, and road safety ratings can make routing more sustainable and user-centric.

Voice and Gesture-Based Interfaces: To improve accessibility, especially for on-the-go users, the navigation system could include voice command or gesture-based control features with multilingual support.

Augmented Reality (AR) Navigation: AR-based navigation overlays can be added for real-world route visualization through smartphone or smart glasses, enhancing user engagement and accuracy.

Cloud-Based Data Storage and Analytics: Storing route and user data on cloud platforms can support scalable analytics, performance benchmarking, and AI-based route improvements.

Bilingual and Regional Expansion: Expanding beyond Hindi-English support to other Indian languages can make the system more inclusive and locally adaptable.

VI. CONCLUSION

The Quick Path-Smart Road Finder idea effectively illustrates how modern web-based tools and algorithmic logic might enhance town routing..The algorithm successfully calculates many optimised routes by combining Dijkstra's Algorithm and K-Shortest routing methods, giving clients the ability to select one that corresponds to cost, time, and distance. While database integration and dynamic map visualisation ensure accuracy alongside user engagement, bilingual support (in Hindi and English) increases accessibility for a larger audience. The system simplifies travel anxiety or advocates data-driven mobility by offering a workable solution to frequent commuting issues in expanding cities like Indore in India. 24-hour traffic updates, artificial intelligence-driven models for predictions, as well as internet of things (IoT) connectivity are possible future additions to the solution that could promote Smart City projects and the goal of digitising India.

References

[1]E. W. Dijkstra, "A note on two problems in connexion with graphs," Numerische Mathematik, vol. 1, pp. 269–271, 1959.
https://eudml.org/doc/131436
[2]T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, Introduction to Algorithms, 3rd ed., MIT Press, 2009.
https://mitpress.mit.edu/9780262033848/introduction-to-algorithms/
[3]S. R. Biradar, "Smart City Navigation Using Shortest Path Algorithms," IJARCS, 2018.
https://ijarcs.info/index.php/Ijarcs/article/view/6030
[4]A. Sharma and R. Patel, "Implementation of K-Shortest Path Algorithms for Real-Time Traffic Navigation," IJCA, vol. 175, no. 30, pp. 10–14, 2020.
https://www.ijcaonline.org/archives/volume175/number30/sharma-2020-ijca-920060.pdf
[5]M. Gupta and A. Agarwal, "Cost-Based Route Optimization in Urban Areas Using Graph Theory," IRJET, vol. 7, no. 8, pp. 563–567, 2020.
https://www.irjet.net/archives/V7/i8/IRJET-V7I875.pdf
[6]Government of India, "Indore Smart City Mission Report," Ministry of Housing and Urban Affairs, 2023.
https://smartcities.gov.in/upload/uploadfiles/files/Indore_Smart_City.pdf
[7]J. Singh, S. Fatima, and A. S. Chauhan, "Multi-Objective Travel Route Optimization Using Genetic Algorithm," IJISAE, vol. 11, no. 3, pp. 785–794, 2023.
https://ijisae.org/index.php/IJISAE/article/view/4122
[8]H. Daneshvar et al., "Designing a Hybrid Intelligent Transportation System for Optimization of Goods Distribution Network Routing Problem," DMAME, vol. 6, no. 2, pp. 907–932, 2023.
https://dmame.ef.uns.ac.rs/index.php/dmame/article/view/430
[9]J. Chen et al., "Dynamic Routing Optimization in Software-Defined Networking Based on a Metaheuristic Algorithm," Journal of Cloud Computing, vol. 13, Article 41, 2024.
https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-024-00441-0
[10]S.-H. Huang et al., "A New Hybrid Algorithm for Solving the Vehicle Routing Problem with Route Balancing," International Journal of Industrial Engineering and Management, vol. 14, no. 1, pp. 51–62, Mar. 2023.
https://ijiemjournal.uns.ac.rs/article/view/234
[11]A. Kumar and P. Mehta, "Cost-Based Route Optimization Models for Urban Transport Systems," International Journal of Transportation Engineering and Management, vol. 10, no. 2, pp. 45–56, 2020.
[12]L. Yang, H. Chen, and X. Li, "MVC Frameworks for Scalable Web Applications," IEEE Software Engineering Letters, vol. 8, no. 2, pp. 85–92, 2019.
[13]K. Geebelen, J. Huybrechts, and M. Leroi, "Applying MVC Design to Web-Based Intelligent Systems," ACM Computing Review, vol. 15, no. 1, pp. 31–37, 2020.
[14]Sharma, C., & Kate, V. (2014). ICARFAD: a novel framework for improved network security situation awareness. International Journal of Computer Applications, 87(19).