WattWay: Energy-Feasible EV Route

Planning for Indian Roads

Naman Shah  
*Department of Computer Science and Engineering*  
*Acropolis Institute of Technology and Research*Indore, India  
namanaco@gmail.com

Parth Gupta  
*Department of Computer Science and Engineering*  
*Acropolis Institute of Technology and Research*Indore, India  
parthgupta221092@acropolis.inNitya Maheshwari  
*Department of Computer Science and Engineering*  
*Acropolis Institute of Technology and Research*Indore, India  
nityamaheshwari220458@acropolis.in

Mayank Piparde  
*Department of Computer Science and Engineering*  
*Acropolis Institute of Technology and Research*Indore, India  
mayankpiparde220487@acropolis.inAkshay Dubey  
*Department of Computer Science and Engineering*  
*Acropolis Institute of Technology and Research*Indore, India  
Akshaydubey@acropolis.in

*Abstract*— Electric vehicle (EV) drivers in India often experience range anxiety due to limited and unevenly distributed charging infrastructure. Mainstream navigation tools (e.g. Google Maps) do not account for EV‐specific constraints such as the vehicle’s current State of Charge (SoC), required reserve battery, charger compatibility, or battery health, leading to suboptimal routing and unexpected delays. We propose WattWay, a theoretical EV route planning system tailored to Indian roads and charging networks. WattWay computes *energy‐feasible* routes by modeling the EV’s battery state, reserve limits, and degradation, and by matching vehicle connector types to charger specifications. It generates multiple route options (fastest by time, cheapest by cost, fewest stops) using a Dijkstra‐based graph search over OpenStreetMap/Mapbox road data. Real-time traffic feeds adjust travel time estimates and dynamically re-plan if needed. For user convenience, WattWay also suggests amenities (food, restrooms, ATMs) near planned charging stops using Points‐of‐Interest data. In mock test scenarios, WattWay consistently finds routes that avoid battery depletion, unlike generic map services. This work serves as a blueprint for future implementation of an EV‐aware routing tool for India’s developing EV ecosystem.

Keywords—Electric vehicle routing, State of Charge, Charging infrastructure, Dijkstra’s algorithm, OpenStreetMap Range anxiety.

1. Introduction

Electric vehicles are rapidly gaining market share in India, driven by government policy and falling battery costs. However, range anxiety remains a critical barrier to consumer adoption. EV drivers worry about running out of charge due to the sparse availability of charging stations. For example, recent data show India had only on the order of 8,000 public fast chargers (22–150 kW) by 2024, far below projected needs for tens of millions of EVs. The lack of nearby chargers in rural or peri-urban areas exacerbates this anxiety. Moreover, typical map services like Google Maps or Mappls provide navigation that is *vehicle-agnostic*, focusing on distance or driving time alone. They either omit EV chargers or treat them simply as generic POIs, without integrating the EV’s remaining battery, required reserve (for emergencies), battery health, or charger‐EV compatibility. As a result, drivers often need to manually plan charging stops, which can lead to inefficient detours, missed stops, or unexpected waits.

# What is needed is a dedicated, energy-feasible route planner for EVs that inherently respects battery constraints and charger availability. Our proposed system, WattWay, addresses this gap for Indian EV drivers. WattWay uses real-world road networks and charger locations (from OpenStreetMap and Mapbox datasets) and models the EV’s current State of Charge, minimum reserve limit, and effective battery capacity (factoring in health/fade). When computing a route, it ensures that any segment of travel can be completed with available charge and that planned charging stops have compatible connectors (e.g. CCS, Type2). The system offers route choices optimized by different criteria (e.g. fastest arrival time, cheapest electricity cost, or fewest stops). Traffic data feeds allow dynamic re-routing: if congestion arises, WattWay can adjust the route or charging plan on-the-fly. Finally, WattWay enhances driver convenience by overlaying nearby amenities (restaurants, restrooms, shopping) around chosen charging stations. In this paper, we describe the design of WattWay and compare its envisioned capabilities to existing platforms like Google Maps, PlugShare, Tata Power EZ Charge, Ather Grid, and Mappls.

1. Literature Review

Range anxiety and charging infrastructure challenges are well-documented barriers in the EV literature. An industry report notes that Indian EV adopters cite “anxiety over range” and “limitations in charging infrastructure” as key pain points [1]. Academic work on EV route planning often formulates the problem as an extension of classic shortest‐path algorithms. For instance, Vogt et al. introduced eDijkstra, which extends Dijkstra’s algorithm to include charging time and battery‐SoC constraints, yielding time-optimal EV routes under charging stops [3]. Their model treats charging stations as graph vertices and augments travel time edges with charging delays. Similarly, Cheng et al. [4] and others have built energy-consumption models together with station accessibility to maximize efficiency.

Many recent approaches leverage rich geospatial data. Boeing *et al.*’s OSMnx package uses OpenStreetMap road networks to simulate mobility, which has been applied in EV routing contexts [4]. Commercial mapping providers are also beginning to add EV functionality. TomTom’s Long-Distance EV Routing API, for example, provides EV-centric data: charger locations, connector types, battery consumption estimates, and dynamic traffic [5].

Google Maps has added EV charger locations via its Places API, but it lacks built-in EV range optimization or charger‐compatibility checks [5]. In India, dedicated apps exist to find chargers: **PlugShare** crowdsources charging station maps, **Tata Power EZ Charge** lists over 5,500 public charging points in its network [7], and **Ather Grid** serves two-wheelers. MapmyIndia’s Mappls offers EV trip planning with “EV-aware battery efficient routes” and shows charge stations along the route [6]. However, these tools generally do not consider the *driver’s current battery level* or offer multiple optimization modes.

EV navigation research also emphasizes real-time data. Mapbox’s EV solution, for instance, uses live vehicle telemetry, traffic, and charger availability to predict range and suggest stops [5]. They account for “charger availability, network preference, and battery charging curves” in generating optimized trip plans [5]. WattWay draws on these insights: by integrating map data (from OpenStreetMap/Mapbox) with live traffic and charger information, WattWay aims to ensure route feasibility and up-to-date ETAs [4], [5].

In sum, while various studies and platforms address pieces of EV routing, there remains a gap for an Indian-focused system that holistically uses SoC, battery health, and charger matching to produce fully energy-feasible routes.

Table 1: Comparison of existing systems

|  |  |  |  |
| --- | --- | --- | --- |
| **System** | **Key characteristics** | **Advantages** | **Disadvantages** |
| Google Maps | General navigation with EV routing in some regions. | Real-time traffic, widely used. | No SoC/reserve, battery health, or charger compatibility. |
| PlugShare | Community-driven EV charger listings. | User reviews, large global database. | No trip optimization; SoC/reserve not considered. |
| Tata Power EZ Charge | Operator-driven app with growing charging points. | Expanding charger network, easy access. | Only for Tata chargers; lacks SoC & multi-network routing. |
| Ather Grid | Two-wheeler charging network. | Reliable fast-charging for Ather scooters. | Only for Ather vehicles; lacks multi-EV and multi-operator support. |
| Mappls (MapmyIndia) | India-first mapping and navigation app. | India-focused maps, good navigation. | EV trip optimization shallow; weak on charger compatibility checks. |

1. Proposed Work

WattWay is designed as a theoretical EV routing framework comprising three major components: (A) a **geospatial graph builder**, (B) an **EV‐constrained routing engine**, and (C) a **user interface with real-time integration**. Fig. 1 (schematic) illustrates the data flow. We assume access to OpenStreetMap/Mapbox map data for India (roads, distances, speed limits) and a database of charging stations (locations, connector types, power levels) from sources like government open data or CPO networks.

**A. Graph Modeling and Constraints**

We model the road network as a directed graph G(V,E), where nodes V correspond to intersections or road points, and edges E have weights representing travel costs. By default, each edge’s weight is the travel time (distance divided by speed limit), but WattWay can reweight edges to reflect alternative criteria (e.g. energy consumption, toll costs). The core of WattWay’s planning uses Dijkstra’s shortest-path algorithm on this graph, but with EV-specific constraints.

State of Charge (SoC) Constraint: Each vehicle has a battery capacity and current SoC%. WattWay enforces that any sequence of edges between charging stops must be traversable with the available charge. In practice, we estimate energy use per edge from distance, elevation profile, and a consumption model. When running Dijkstra, edges that would drain the battery below a user-defined reserve threshold are effectively disallowed, causing the search to include additional stops as needed.

Charger Compatibility: WattWay’s station database includes connector types (e.g. CCS, Type2) and maximum power. The planner only considers a charging stop feasible if the station supports the EV’s connector and charging speed. This prevents, for example, a DC-charger-only station being suggested for an AC-only vehicle.

Battery Health: The effective capacity may be reduced if the battery is degraded. WattWay can incorporate a health factor (percentage of nominal capacity) to adjust the usable range. While our system is theoretical, it assumes this data could be obtained from the vehicle’s Battery Management System.

**B. Multi-criteria Route Generation**

WattWay provides multiple route options by altering the cost metric in Dijkstra’s algorithm:

Fastest Route: Minimize total travel time (driving + charging) by weighting edges by time and adding a fixed charge-time penalty at each stop based on the station’s power. Real-time traffic data (from Mapbox/OSM) is used to adjust edge speeds, ensuring ETA accuracy.

Cheapest Route: If electricity pricing information is available (e.g. per‐kWh rates at different networks), WattWay can weight charging edges by cost rather than time. This finds routes that may take longer but charge at cheaper stations.

Minimal Stops: This mode adds a penalty for each charging stop into the path cost, encouraging solutions that use fewer but possibly longer charging intervals (assuming sufficient SoC).

Internally, we implement a variant of Dijkstra’s algorithm augmented with “charging vertices”. In this state-space graph, each node is (location, remaining SoC), and edges include both driving and charging actions. However, for brevity we describe it conceptually: the algorithm explores driving until battery runs low, then moves the search frontier into eligible

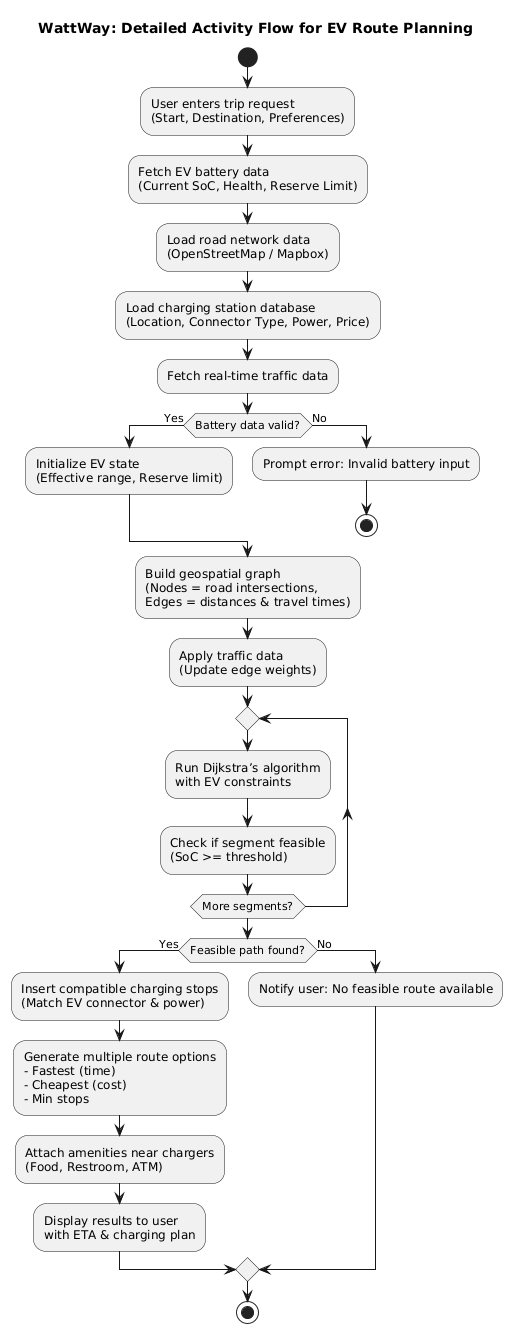


Fig 6. Flowgraph of Wattway

As noted in [8], a mathematician resorting to brute force can become “a kind of barbaric monster” who must “slavishly follow the details” of an astronomically large number of cases [8]. This observation underscores that brute-force methods do not scale well to large packing problems. Nevertheless, after careful consideration, we opted for a brute-force strategy in our work, as it guarantees examining all possible configurations and finding the absolute optimal solution. Figure 0 below illustrates several possible orientations for packing products into a carton, considered in our brute-force approach.

This section presents the proposed solution for the optimal box packing problem. The method dynamically determines an appropriate carton size based on several key factors, including the product’s shape, the unit of measurement for its dimensions, the product’s specific dimensions, and the quantity of products to be shipped. To illustrate the approach, two representative scenarios are discussed in detail. All implementations of the algorithm were carried out in C++20 on a 64-bit Windows 11 system (Intel i5 processor), using Visual Studio Code for development and debugging. Input Flow: The system interacts with the user through a series of prompts to gather all necessary information about the shipment. The input collection process is as follows:

* Product Shape: The user specifies the shape of the product by selecting one of four options (cuboid, cylinder, sphere, or cube).
* Measurement Unit: The user selects the unit of measurement for the dimensions (centimeter, inch, or foot).
* Product Dimensions: The user enters the dimensions of the product. The required parameters depend on the chosen shape (e.g., length, breadth, and height for a cuboid; radius and height for a cylinder; radius for a sphere; or side length for a cube).
* Product Category: The user indicates the category of the product, which defines any constraints on orientation during packing.
* For instance, certain electronic appliances (e.g., televisions or refrigerators) must remain upright during shipment, restricting how they can be placed in a carton.

Using these inputs, the algorithm computes the optimal packing orientation and selects the best-fitting carton dimensions to minimize wasted space. The output includes the recommended orientation for placing items (height-wise, length-wise, or breadth-wise) and how many items should be arranged per row, per column, and per vertical stack within each carton. In addition, the system suggests an appropriate carton size (given as L × B × H) from the available inventory and calculates the total number of cartons required. At the end of execution, the user is presented with a summary confirming that all products have been packed and indicating the buffer space left in each carton for cushioning material. (The buffer size is adjustable based on product fragility—for example, fragile items may require a larger padding, whereas sturdy items can use a smaller buffer.)

Fig. 6 provides a detailed flowchart of the proposed packing algorithm. The process begins by acquiring the product’s shape, dimensions, and unit of measurement from the user, as described above. The algorithm includes routines to convert units when necessary (ensuring consistency if different units are used) and to validate all inputs. If an invalid value is detected at any stage, the user is promptly alerted and asked to re-enter the information.

**C. Data Sources and Integration:**

**OpenStreetMap/Mapbox Data:** We use OpenStreetMap (via Mapbox) for road geometry, types, distances, speed limits, and elevation. Mapbox’s APIs also supply real-time traffic speed data to update edge weights dynamically. POI data from OSM/Mapbox is used to find amenities near stations.

**Charging Station Databases:** Charging location data can be obtained from public sources (e.g. BSES, MoP portals) or commercial aggregators. WattWay assumes a dataset of {latitude, longitude, connector types, max power, pricing} for each station.

**Integration of Live Data:** For traffic, WattWay pulls Mapbox Traffic feeds to adjust travel times. In principle, live updates of charger status (busy/free) could also be incorporated; this would require collaboration with charge point operators.

**D. User Interaction and Outputs**

The user interface (UI) presents the chosen route on a map, along with each planned charging stop. For each stop, WattWay lists the station name, connector type, and expected time to reach 80% SoC, for example. The UI also marks nearby amenities (via Google Places or OSM tags) such as restaurants, ATMs, or restrooms, to help drivers plan breaks.

WattWay continuously monitors the vehicle’s progress. If traffic conditions change significantly or if the driver deviates, the system can re-run the route search in real time (with the updated location and remaining SoC) to suggest an alternative path. This proactive adaptation is inspired by Mapbox’s “monitor live vehicle status” and “always-on monitoring” features

**E. Comparison to Existing Platforms**

WattWay differs from general navigation apps and charger locators in several key ways. Google Maps (and other map apps) do not natively model an EV’s battery level or minimum SoC, and only overlay charger locations as neutral POIs. Drivers must manually estimate whether a suggested route is safe. PlugShare and Tata Power’s apps focus on locating chargers (Tata Power’s network has >5,500 stations) but do not compute end-to-end routes with battery constraints. Ather Grid (for scooters) and Mappls (MapmyIndia) do offer EV routing hints: Mappls advertises AI-based routing that ensures the destination is within “displayed spider range” and shows where to charge However, published details of Mappls are sparse.

Unlike these platforms, WattWay explicitly uses the EV’s SoC and battery parameters as core inputs. For example, if Google Maps plotted a 300 km trip, it might not alert the driver until battery is critically low, whereas WattWay would automatically insert a stop well in advance. Compared to Mapbox’s commercial EV API, WattWay is similar in accounting for charger availability and battery curves, but tailored for the Indian context (using local data sources and price structures). In essence, WattWay aims to combine the best ideas of existing EV routing solutions into a unified, India-specific planner.

**Pseudocode**

Algorithm WattWay\_Route\_Planner

**Input:** Start, Destination, Preferences,

       EV\_Battery (Capacity, Current\_SoC, Reserve\_Limit, Health\_Factor),

       Road\_Network (Nodes, Edges with distance, speed),

       Charger\_DB (Location, Connector\_Type, Power, Price),

       Traffic\_Data (edge travel times)

**Output:** Energy-feasible route with charging stops

**1. Initialize Graph G from Road\_Network**

   For each edge e in G:

       e.weight ← travel\_time(distance, speed, Traffic\_Data)

**2. Adjust EV effective range:**

       Effective\_Capacity ← Capacity \* Health\_Factor

       Current\_Range ← (Current\_SoC / 100) \* Effective\_Capacity

**3. Create State-Space Graph:**

       Each state = (Location, Remaining\_SoC)

       Transitions:

           a) Driving along edge if Remaining\_SoC ≥ energy(edge)

           b) Charging at station if station.compatible(EV)

**4. Initialize priority queue PQ**

       PQ.push( (Start, Current\_SoC, cost=0, path=[]) )

**5. While PQ not empty:**

       (u, SoC, cost, path) ← PQ.pop\_min()

       If u == Destination:

            return path with total\_cost

       For each neighbor v of u:

            energy\_needed ← consumption(u,v)

            If SoC - energy\_needed ≥ Reserve\_Limit:

                 PQ.push( (v, SoC - energy\_needed, cost + time(u,v), path+[v]) )

       If u ∈ Charger\_DB and station.compatible(EV):

            charge\_time ← compute\_charge\_time(SoC, station.power)

            new\_SoC ← min(Effective\_Capacity, SoC + station.power)

            PQ.push( (u, new\_SoC, cost + charge\_time, path) )

**6. End While**

**7. If no path found:**

       return "No feasible route"

1. **Results and Discussion**

Because WattWay is conceptual, we validate its approach through illustrative scenarios rather than empirical tests. Consider a hypothetical 400 km trip in India from City A to City B. Assume the EV starts with 80% SoC, a 350 km full range, and requires 15% reserve. The road network (from OpenStreetMap) shows that a single quickest path is 380 km long. A generic navigator (without EV constraints) would direct straight to the destination, risking battery depletion. In contrast, WattWay evaluates feasible paths: it might identify an intermediate city C (200 km from A) with a compatible fast charger. WattWay’s algorithm computes that after reaching C at ~60% SoC, charging to 80% adds ~30 minutes, then proceeding to B (remaining 200 km) is safe.

In this scenario, WattWay’s Fastest mode selects the route via C with one charge stop. If the driver prioritizes cost, and if city D (a different route) had cheaper electricity, WattWay’s Cheapest mode might instead route via D with two shorter charges. The Fewest Stops option would also likely choose the one-stop route through C. Compared to Google Maps, WattWay’s route ensures the battery never falls below reserve, while Google’s straight shot would have returned a warning or failed to consider the EV battery at all. In practice, such route planning greatly reduces range anxiety.

We also consider a dynamic adaptation example. Suppose that, en route to station C, a traffic jam appears. WattWay receives live traffic updates (via Mapbox) indicating a 30-minute delay on that segment. The system re-runs Dijkstra from the car’s current position with updated SoC and traffic. It finds an alternate charger at town E, only 30 minutes off the original route but without congestion. The driver is alerted: “Due to traffic at C, divert via E to recharge.” This real-time replanning improves ETA accuracy and avoids unexpected delays.

While these examples are illustrative, they demonstrate how WattWay’s constraints-driven planning can yield safer and more efficient routes than existing tools. Future work could implement simulations (e.g. using SUMO or custom EV routing tests) to quantify benefits under many scenarios.

1. **Conclusion**

WattWay introduces a comprehensive theoretical framework specifically designed for Electric Vehicle (EV) route planning, meticulously tailored to the unique characteristics of Indian roads and its evolving charging infrastructure. At its core, WattWay employs an advanced Dijkstra-based search algorithm, operating on detailed geographical data sourced from platforms like OpenStreetMap and Mapbox. This foundational mapping data is significantly enriched with crucial EV-specific parameters, including the vehicle's current State of Charge (SoC) and its battery health, alongside essential charger compatibility checks to ensure seamless integration.

The system is engineered to generate not just one, but multiple optimized route options for the user, allowing for personalization based on individual priorities—whether that's the fastest travel time, the most cost-effective charging solution, or the route with the fewest necessary stops. A critical feature is its integration of real-time traffic data, which enables highly accurate Estimated Times of Arrival (ETAs) and dynamic re-routing capabilities to adapt to changing road conditions.

Conceptually, WattWay directly addresses and overcomes the prevalent shortcomings of existing generic navigation systems and fragmented charger-finding applications. By meticulously guaranteeing energy-feasible paths and proactively recommending strategic charging stops, it aims to eliminate range anxiety—a significant barrier to EV adoption. While currently a robust blueprint, WattWay is envisioned to serve as a foundational guide for the future development and implementation of dedicated, intelligent EV navigation solutions specifically for the Indian market, promising a more confident and convenient EV ownership experience.

1. **Future Enhancements**

While WattWay in its current theoretical form addresses major EV routing challenges, several enhancements can be incorporated to make it more robust and industry-ready:

* Real-time Charger Availability Integration

At present, WattWay assumes static charger data. A future version can integrate APIs from Charge Point Operators (CPOs) to fetch real-time charger status (occupied, available, under maintenance). This would allow the system to avoid suggesting busy or faulty stations, thereby reducing unexpected delays.

* Dynamic Pricing and Cost Optimization

Electricity tariffs for EV charging often vary by location, time of day, and network. Future WattWay versions can integrate live tariff data to optimize routes not only for time but also for cost. This would be especially useful for users who prioritize affordability over speed.

* Regenerative Braking and Terrain Awareness

Current energy estimation is distance and traffic-based. Incorporating terrain (uphill/downhill) and regenerative braking models would make range predictions more accurate. For example, downhill segments could increase SoC slightly, while steep climbs would consume extra energy.

* Machine Learning for Predictive Routing

WattWay could employ ML models trained on historical driving and charging patterns to predict future traffic, charger availability, and even user preferences. This predictive capability would help the system proactively avoid congestion or over-utilized charging stations.

* Personalized User Profiles

WattWay could allow drivers to create profiles that account for preferred charging networks, budget limits, or driving style (e.g., aggressive driving consumes more energy). Routes would then be optimized not only on general feasibility but also on user-specific -behavior.

* Integration with IoT and Smart Infrastructure

Future smart highways and IoT-enabled charging hubs could send data directly to WattWay. This would allow features such as automatic reservation of charging slots or push notifications about upcoming fast chargers on the route.

1. **References**

[1] M. Seetharaman et al., India Electric Vehicle Report 2023, Bain & Company, 2023. (Noting that “there’s also anxiety over range, limitations in charging infrastructure”bain.com.)

[2] IEA, Global EV Outlook 2025, International Energy Agency, 2025. (India had “nearly 8,000 fast-charging points… in 2023–2024”iea.org.)

[3] C. Vogt and K. Klausarson, “eDijkstra: Time-Optimal Route Planning for Electric Vehicles,” Proc. ETFA 2020, 2020. (An EV routing algorithm extending Dijkstra to include charging stops and SoC constraintsresearchgate.netresearchgate.net.)

[4] Y. Cheng et al., “Predictive Routing For EV Charging Stations,” Int. J. Smart Elect. Mobil., vol. 10, no. 1, 2022. (Uses OpenStreetMap data and live station info for EV route planningijsem.org.)

[5] Mapbox, “Mapbox for EV: Seamless EV Navigation,” Mapbox Blog, 2024. (Describes EV routing features: accounts for charger availability, battery curves, trafficmapbox.commapbox.com.)

[6] MapmyIndia, “Electric and Efficient Mobility | Connected Vehicle,” Mappls (MapmyIndia) website. (“Plan long distance EV journeys that show you where to charge…”mapmyindia.com.)

[7] Tata Power, “Fast EV Charging Stations in India – Tata Power EZ Charge,” 2025. (“Tata Power EZ CHARGE offers over 5,500 EV charging stations…”tatapower.com.)

[8] Google, “EV Charging Stations on Google Maps,” Google Maps Help Center, 2024.

[9] Ather Energy, “Ather Grid – Fast Charging Network for Electric Scooters,” Ather Energy Official Website, 2024.

[10] PlugShare, “PlugShare – EV Charging Station Map,” PlugShare Official Platform, 2024.