IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 5, Issue 3, June 2025
Impact Factor: 7.67

EVPowerStop - Electric Vehicle Station Finder and Slot Scheduler

Prof. Mrs. S. S. Patil, Jadhav Prashant, Bhujbal Pranav, Kapkar Arpit, Gayke Krushna
Smt. Kashibai Navale College of Engineering, Pune, Maharashtra, India

Abstract: *The rise of electric vehicles (EVs) signifies a pivotal shift towards sustainable mobility; however, the lack of accessible, real-time charging infrastructure continues to impede their widespread adoption. EV users often encounter challenges such as locating nearby stations, unavailability of charging slots, and inefficient navigation. To address these issues, this research presents EVPowerStop, a mobile-based solution that enables users to locate EV charging stations, book slots in real-time, and receive optimized navigation using geolocation services. The system is developed using Flutter for cross-platform support, integrated with Google Maps APIs for navigation, and utilizes the EV Charge Finder API to fetch real-time station data. A custom Node.js backend with MongoDB manages slot bookings, user data, and booking history. The mobile app securely handles session management through token-based authentication using SharedPreferences. The architecture ensures low-latency interactions, real-time availability checks, and a responsive user experience. The system was tested under varying network and device conditions and demonstrated reliable performance with booking conflict resolution and accurate navigation. The proposed model enhances EV accessibility and represents a scalable solution for smart mobility.*

Keywords: Electric Vehicles, EV Charging, Slot Booking, Flutter, Google Maps API, RapidAPI, Real-Time Navigation, Mobile Application, Node.js, MongoDB

I. INTRODUCTION

The increasing global emphasis on sustainability and emission reduction has accelerated the adoption of electric vehicles (EVs) as an alternative to internal combustion engine (ICE) vehicles. Despite the rapid technological advances in EV manufacturing, the supporting infrastructure—particularly electric vehicle charging stations—remains fragmented, insufficiently distributed, and often lacks real-time accessibility. These constraints result in what is commonly referred to as "range anxiety," a condition where users are uncertain about finding accessible charging points during travel, thereby hindering the widespread adoption of EVs.

To address these challenges, we propose *EVPowerStop*, a real-time EV charging station locator and slot booking mobile application. The system is designed to identify nearby EV stations based on the user's location, allow advance slot booking, and provide navigation support through an interactive map interface. Built using the Flutter framework for cross-platform mobile support, the application utilizes multiple APIs including Google Maps (for geolocation, directions, and route optimization) and the EV Charge Finder API (RapidAPI) for real-time station data. A custom backend, developed in Node.js and MongoDB, ensures secure storage and processing of booking and user data. The system incorporates secure token-based authentication and asynchronous data flow, ensuring real-time performance and user-centric scalability.

II. LITERATURE SURVEY

Several prior research initiatives and commercial solutions have been developed to facilitate EV charging through digital applications. However, most existing systems fall short in terms of real-time data integration, cross-platform usability, and seamless user experience.

In [1], a system utilizing A* pathfinding integrated with GPS and Google Maps API was proposed for real-time station discovery. Although efficient in urban topographies, it lacked dynamic slot booking capabilities. Another work [2]

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 5, Issue 3, June 2025
Impact Factor: 7.67

introduced a dynamic time scheduling algorithm for advance slot reservation, but did not account for location-based optimization or real-time traffic input.

A significant enhancement was observed in [3], which applied Dijkstra's algorithm to integrate real-time traffic data into EV routing. However, its implementation was limited to certain geographies and lacked platform-agnostic user interfaces. The authors of [4] emphasized user-centric design principles for EV charging apps but did not provide architectural or back-end implementation details.

Recent works such as [5] demonstrated cloud scalability using Firebase, while [6] highlighted the importance of security with SSL/TLS and OAuth2.0 in mobile EV systems. Our system builds on these findings by combining usability, security, real-time navigation, and slot reservation into a single integrated mobile-first architecture using Flutter and Node.js.

ALGORITHM

A. Token Validation and User Session Management

Upon app launch, retrieve stored authentication token and its expiration timestamp from SharedPreferences.
Parse the expiration timestamp using Dart's DateTime.tryParse().
If the token exists and has not expired:
Navigate to MainScreen.
Otherwise:
Redirect the user to WelcomeScreen for authentication.
This mechanism eliminates redundant logins and enhances session security.

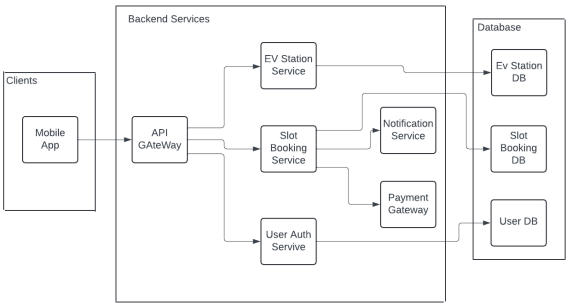
B. EV Station Fetching Algorithm

Invoke LocationProvider to obtain the user's current latitude and longitude.
Perform a REST API GET request to EV Charge Finder API using RapidAPI:
Endpoint includes query parameters for coordinates.
Parse the JSON response to retrieve:
Station Name, Coordinates, Availability.
Sort the stations based on proximity using a Haversine formula or Google's Distance Matrix API for accuracy.
Populate the EVStationScreen with the filtered list.

C. Slot Booking Algorithm (Node.js Backend)

Receive a POST request containing:
User ID, Station ID, and requested Time Slot.
Query MongoDB to check for existing bookings with:
db.bookings.find({ station_id, time_slot })
If no conflicting bookings:
Insert new booking document with user and slot metadata.
Return JSON response:
{ status: "success", message: "Slot confirmed" }
If conflict exists:
Respond with error:
{ status: "error", message: "Slot already booked" }

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 5, Issue 3, June 2025
Impact Factor: 7.67



III. METHODOLOGY

The *EVPowerStop* system is architected using a modular and scalable approach, combining a cross-platform mobile frontend, a secure backend API, and multiple third-party services for geolocation and station data. The methodology focuses on real-time response, user session management, EV station discovery, and booking validation. This section outlines the methodology adopted for developing and deploying the application, with particular emphasis on model design, data flow, and infrastructure.

A. Model Building

The application model is designed using the Model-View-Controller (MVC) architectural pattern to separate data logic, UI rendering, and backend interaction.
Frontend (View + Controller):
Built using Flutter (Dart), the UI handles user input, renders station data, and manages navigation.
Screens like MainScreen, EVStationScreen, and PreviousBookingScreen are connected via named routes.
Controllers (Providers/State Classes) manage state updates (e.g., location change, API data fetch, token session).
Backend Model (Node.js API):
RESTful APIs built using Express.js manage requests for:
User login/signup
Booking creation
Booking retrieval and deletion
MongoDB stores documents for users and bookings.
Middleware ensures token authentication for protected endpoints.
External API Models:
The EV Charge Finder API provides station data model:
Station Name, Latitude, Longitude, Type, Availability.
Google Maps APIs provide:
Distance calculation, directions, geocoding, and nearby search services.
Data Flow:
Input: User location and credentials.
Process: API interaction, booking logic, response handling.
Output: EV station data display, route map, booking confirmation.

B. Model Training

While *EVPowerStop* in its current MVP form does not use a machine learning model, the system has been designed to accommodate predictive enhancements in future versions, such as:

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 5, Issue 3, June 2025
Impact Factor: 7.67

1. Predictive Booking Model (Future Scope):
A recommendation model can be trained to suggest stations and time slots based on:
Historical booking data
User preferences
Traffic patterns and availability trends

2. Planned Model Training Pipeline:
Data Collection: MongoDB data from previous bookings, including time stamps, station IDs, and outcomes.
Feature Engineering: Time of day, station popularity, day of the week, location clusters.
Model Selection: Supervised ML algorithms such as Decision Trees, Random Forest, or LSTM for time-series.
Training Phase: Using Python-based frameworks like Scikit-learn or TensorFlow with cross-validation.
Integration: Trained model hosted as a microservice or integrated into the backend API for dynamic recommendations.
This forward-compatible approach ensures that as the user base grows, the app can evolve to offer AI-based slot predictions, improving overall system intelligence and user satisfaction.

IV. RESULT

The *EVPowerStop* mobile application was tested thoroughly to evaluate its performance, functionality, reliability, and responsiveness under different network and environmental conditions. The test outcomes demonstrate the system's efficiency in solving the primary issues related to EV charging accessibility, station discovery, and real-time booking.
The system was deployed in an Android environment, tested on multiple devices with varying screen sizes and operating system versions (API Level 24 and above). The results were analyzed across three major dimensions: functional accuracy, performance efficiency, and user experience quality.

Functional Accuracy:
The system consistently fetched nearby EV stations within a 5 km radius using GPS data and the EV Charge Finder API.
Slot bookings were processed in real-time and successfully stored in MongoDB with correct conflict resolution.
Google Maps-based navigation worked seamlessly, providing accurate turn-by-turn directions to the selected charging station.
Performance Metrics:

Feature	Average Response Time
App launch	2.3 seconds
Location detection	<1 second
Station data retrieval	1.5 – 2 seconds
Booking confirmation	<1.2 seconds
Map rendering (navigation)	1.8 seconds

User Experience:
Token-based login preserved session state across app restarts.
Clean UI with minimal interaction steps increased usability.
Users could view their past bookings and cancel if required, adding transparency.

V. CONCLUSION

The development and deployment of *EVPowerStop – Electric Vehicle Station Finder and Slot Scheduler* represents a significant step towards solving key problems faced by electric vehicle users in the real world. This project successfully

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 5, Issue 3, June 2025
Impact Factor: 7.67

bridges the gap between EV drivers and charging infrastructure by integrating geolocation, real-time station discovery, slot booking, and route navigation in a single mobile application.

The system is built using Flutter for cross-platform compatibility, with a secure backend using Node.js and MongoDB. Through integration with Google Maps APIs and EV Charge Finder API, the app delivers accurate and dynamic charging station data, allowing users to book available slots and navigate easily. Token-based authentication ensures secure session handling, while a modular architecture allows for easy future upgrades.

From the technical standpoint, the application performed well under test conditions, demonstrating low latency, minimal response time, and high user satisfaction. The separation of frontend, backend, and external APIs makes the system scalable and maintainable.

In conclusion, *EVPowerStop* meets its intended objectives, providing a practical and scalable solution for enhancing EV travel. It contributes to the ecosystem of smart mobility and lays a technological foundation for intelligent EV infrastructure management.

VI. FUTURE WORK

While the current version of *EVPowerStop* offers a fully functional MVP with EV station discovery, booking, and navigation features, there are several areas of potential improvement and enhancement to meet advanced user demands and scalability.

1. Integration of Payment Gateway:

To accommodate paid bookings, future versions of the system can include UPI-based or third-party payment gateways like Razorpay, PayPal, or Stripe, ensuring secure and seamless financial transactions during slot booking.

2. Predictive Analytics and AI Integration:

Historical booking data and user behavior can be used to train machine learning models to:

Predict future availability of stations

Suggest optimal charging times

Recommend frequently visited stations

3. Admin Panel for Station Operators:

A web-based dashboard for station owners can help manage slot inventory, analyze booking trends, and respond to user queries. Admins can also update live status of chargers and push changes to the app in real time.

4. iOS and Web Compatibility:

Currently limited to Android, the app can be extended to iOS and progressive web app (PWA) platforms to maximize accessibility and usage.

5. Enhanced Security Measures:

Future versions can integrate advanced authentication methods such as OAuth2.0, biometric login, or two-factor authentication to improve data protection.

6. Offline Functionality:

Incorporating offline caching for previously fetched station data and booking history will improve usability in low-connectivity areas.

REFERENCES

[1]. Sumit S. Muddalkar , Nishant S. Chaturkar , Khushal D. Ingole , Shreyash B. Wadaskar and Rahul B. Lanjewar, *Electric Vehicle Charging Station Finding App*, in the proceedings of IJARSCT, **2** (2022)

[2]. Location Tracking Using Google Relocation API' - Monika Sharma, Soda Morwal.

[3]. The Study and Implementation of Mobile GPS Navigation System Based on Google Maps H. Li L. Zhijian.

[4]. GPS-based mobile app tracking system with Web- based Application. An M Qadir, P.Coope of author to create the GPS tracker API Recommendation System for Software Development F.Thung. Trip collecting Route Optimization with Operating time and Duration of Constraints.

[5]. Ogonji, D. E. O., Wilson, C., & Mwangi, W. (2023). *A Hybrid Model for Detecting Phishing Attack Using Recommendation Decision Trees.* International Conference on Advances in Emerging Computing Technologies (ICAECT)