WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

PARKIT – A Parking Assistant System

Gayathri V[#1], Indhu Kavya P[#2], Isaac Daniel S[#3], Ishwarya A K[#4], Jai Vikash A R[#5]

[#1] Assistant Professor, Department of Computer Science and Engineering, Sri Shakthi Institute of Engineering and Technology, India.
[#2] Student, Department of Computer Science and Engineering, Sri Shakthi Institute of Engineering and Technology, India.
[#3] Student, Department of Computer Science and Engineering, Sri Shakthi Institute of Engineering and Technology, India.
[#4] Student, Department of Computer Science and Engineering, Sri Shakthi Institute of Engineering and Technology, India.
[#5] Student, Department of Computer Science and Engineering, Sri Shakthi Institute of Engineering and Technology, India.

ABSTRACT :

Finding a parking space is a daily difficulty for drivers in cities, campuses and commercial areas, yet it is still commonly handled by driving around, asking attendants or relying on informal signs, which leave drivers with little reliable information about where a free space is. This paper presents ParkIt, a parking assistant that brings parking lot, slot, tariff and availability information into one centralized application. Operators maintain lot and slot records, drivers search for nearby lots and view their current availability, and a dedicated map page shows the last reported occupancy of each lot whenever a valid occupancy feed is available. The system follows a modular architecture in which the user interface communicates with backend services that process requests and retrieve stored data, and occupancy information is supplied by sensors, cameras, an attendant's entries or a compatible parking service. The availability display is designed to distinguish a freshly received occupancy update from an outdated one, so that it never presents old data as current. The current development work centres on the parking management application and its availability interface. No performance benchmark or user study has been completed, so this paper makes no quantitative claims; instead it states the limitations of the prototype and proposes an evaluation protocol covering functional testing, occupancy-update verification, usability and performance. Planned extensions include online payment, automated notifications, license plate recognition, availability prediction and enhanced reporting.

Keywords: ParkIt, Parking Assistant, Smart Parking, Parking Availability, Slot Reservation, Parking Management, Web Application, Real-Time Monitoring.

I. INTRODUCTION

Parking is an essential part of urban mobility, since almost every car trip ends with a search for a place to leave the vehicle. Colleges, hospitals, shopping centres and city districts often operate several parking lots of different sizes, and running them requires coordination between operators, attendants and drivers. In a conventional setup, lot capacities, tariffs and slot assignments are kept in registers, spreadsheets or the attendant's memory, and drivers have no way of knowing whether a lot is full before they reach it. They may circle the area, ask passers-by or phone the lot for updates, which leads to wasted time, extra traffic and fuel consumption, and frustration; Shoup [2] reviewed studies reporting that cruising for parking accounts for a considerable share of traffic in congested districts. Operators face similar difficulties when updating tariffs, monitoring several lots and announcing closures.

The problem addressed here is the lack of a centralized and accessible system for managing parking facilities and giving drivers timely information about free spaces. The conventional approach presents seven main difficulties: limited visibility of slot availability; manual upkeep of lot, slot and tariff data; delayed communication of closures or tariff changes; difficulty finding a suitable lot near the destination; dependence on telephone calls or physical inspection for updates; limited centralized access to records; and no common interface for monitoring several lots.

This paper presents **PARKIT – A Parking Assistant System**, a web application that organizes parking information and provides a foundation for live availability monitoring. Its main objectives are:

1) To develop a centralized parking assistant application that organizes lot, slot, tariff and availability information.
2) To provide an interface for searching nearby lots and viewing their details and current availability.
3) To develop a live availability page that displays occupancy information when a valid occupancy feed is available.
4) To improve communication between parking operators and drivers and to reduce the manual effort of maintaining and retrieving records.
5) To provide a modular, maintainable architecture that supports future features such as online payment, notifications and availability prediction.

The system is organized around three user groups. The administrator (parking operator) maintains lot, slot and tariff records according to the permissions implemented in the application. The driver searches for lots, views availability and opens the map and navigation interface and, if reservation is included in the implemented version, books a slot for a limited time. The attendant is associated with the operational information of a lot and, if attendant login and manual occupancy entry are included in the implemented version, supports occupancy reporting. The available permissions and features must correspond to the completed application. The principal contribution of the project is the integration of parking information management with a dedicated availability and navigation interface in one environment, which can be extended without redesigning the application.

The remainder of the paper is organized as follows. Section II reviews related work. Section III describes the methodology and architecture. Section IV details the implementation of each module. Section V summarizes the advantages of the approach. Section VI reports the development status, limitations and evaluation protocol. Sections VII and VIII present the conclusion and future work.

II. LITERATURE SURVEY

Smart parking, intelligent transportation systems, vehicle positioning and web-based information systems have been studied extensively, and this body of work provides the foundation for a centralized parking assistant application.

Intelligent Transportation Systems. Intelligent Transportation Systems (ITS) use information technology, communication systems and data processing to improve transportation operations, including vehicle monitoring, traffic management, route planning and traveller information services. Sussman [1] discusses the role of ITS in modern transportation and the integration of information systems with transportation infrastructure. ParkIt applies this general concept to parking by concentrating on lot information, slot availability and access to navigation information.

Parking and Traffic Congestion. Shoup [2] reviewed studies of cruising for parking and found that drivers searching for a free space can account for a considerable share of traffic in congested districts, and Shoup [3] argued that how parking is priced and supplied shapes driver behaviour and urban traffic. Although these works address urban policy rather than software, they show why timely information about where space is available matters: the less time drivers spend searching, the less they add to congestion. A single lot is much smaller than a city district, but similar principles apply, and a structured representation of lots, slots and tariffs makes information easier to maintain and retrieve.

Smart Parking Systems. Idris et al. [4], Lin et al. [5] and Barriga et al. [6] review smart parking solutions from a technological perspective, covering sensing technologies, communication networks and driver-facing services such as availability information, guidance and reservation. Typical sensing options include in-ground magnetic or infrared sensors, ultrasonic sensors, cameras with image processing and manual counting at entrances, and the choice involves a trade-off between cost, accuracy and installation effort. In ParkIt, any of these sources can supply an occupancy count or slot status to the backend, and the application displays the most recent available state. A sensor reading alone does not guarantee a correct display, because calibration, communication availability and update frequency also influence the service.

Positioning and Navigation. The Global Positioning System (GPS) lets compatible devices estimate their position from satellite signals [7]. When a driver permits location access, the application can sort lots by distance and show the driver's position relative to a lot on a digital map. Route-planning algorithms [8], [9] can then estimate a path from the current position to the chosen lot, although the application may also hand navigation over to an external map service. GPS accuracy degrades near tall buildings and inside covered or underground lots, so slot-level guidance inside a structure may require other positioning aids.

Digital Maps and Geolocation. Geographic Information Systems (GIS) represent geographical features and support spatial visualization. In a parking application, a map can show the position of lots, their availability status and their relationship to the driver's destination. Open map data such as OpenStreetMap [12] or a commercial platform such as Google Maps [15] is one possible source, and the W3C Geolocation API [11] defines how a browser-based client can obtain a device position. The exact mapping technology is a design choice of the implementation.

Web-Based Information Management. A client–server architecture separates the presentation layer from backend processing and data storage, which simplifies the maintenance of user interfaces, validation rules and records. The REST architectural style [10] is a common basis for such separation, and in ParkIt the same principle serves operator and driver interfaces from a common source of parking information. The OWASP Top 10 [13] lists the most common web application risks and serves as a checklist for access control and input validation.

Gaps Identified. The reviewed work addresses parking policy, sensing technologies, positioning and navigation separately. ParkIt applies related principles in a single assistant, combining record management with an accessible availability and navigation interface. Six design requirements emerge: centralized parking information; clear presentation of lot, slot and tariff details; an availability display connected to a valid source of occupancy data; appropriate access control for each user role; reliable handling of missing, delayed or unavailable information; and a modular design that supports future features.

III. PROPOSED METHODOLOGY

ParkIt follows a modular application architecture in which users interact with a frontend interface, requests are processed by backend services, and parking information is retrieved from a database or external service. Live availability additionally requires a source of occupancy data, which may be sensors, cameras, an attendant's entries or another authorized parking service. The system is designed around four activities: maintaining parking records, retrieving and displaying lot and slot information, obtaining and presenting occupancy updates, and validating requests while controlling access to protected information.

System Architecture. The architecture has four layers. The presentation layer contains the screens used by operators and drivers and presents lot information, slot status, navigation controls and the availability map. The application layer processes requests, validates input, applies business rules and coordinates the frontend, database and occupancy source. The data layer stores user accounts, lots, slots, tariffs and reservations, depending on the implementation. The occupancy service, a set of sensors, cameras, manual entries or a compatible parking service, supplies slot status or counts together with timestamps. It is a necessary dependency for genuine live availability: a map showing a fixed or sample status does not by itself demonstrate real-time monitoring.

Workflow. The user opens the application and is shown the screens permitted by the implemented navigation and access rules. An authorized administrator maintains parking records, and the driver searches for a lot near the destination, views its availability and, where reservation is implemented, books a slot. The application requests occupancy information from the configured source, processes the received counts and timestamp, and displays them. The interface states whether availability information is present and sufficiently recent; if the occupancy source is unavailable, it shows an appropriate status instead of implying that old data is current. Authentication, role-specific permissions and attendant-side occupancy reporting depend on those modules being present in the final implementation.

Parking Lot Management. This module maintains the parking facilities handled by the organization. Potential data fields are the lot identifier, name, address and coordinates, total capacity, vehicle types accepted, opening hours, tariff and operational status. The administrator interface can add, view, update and remove lot records, subject to the permissions and validation rules of the application.

Slot Management. This module organizes the individual parking slots within each lot. A slot record may contain a slot identifier, the lot it belongs to, the level or zone, the slot type (for example regular, disabled-access, electric-vehicle or two-wheeler) and, if the database supports it, the current status and any reservation. Structured slot records reduce duplication and make it easier to associate slots with the lots they belong to.

Live Availability Monitoring. An occupancy source determines whether slots are free and transmits the result to the application through a configured communication mechanism, and the availability interface presents it on a digital map and in a lot list. An occupancy record contains the lot or slot identifier, the status (free, occupied or reserved), a count where only totals are available, and the timestamp of the measurement. The frontend can refresh the display periodically or receive updates through a real-time communication mechanism. The system must distinguish newly received data from older stored data and display a stale-data warning when the last update is too old. The update frequency and communication mechanism are to be selected and tested during implementation.

Data Model. The logical data model has seven entities: the user (account and profile information, where accounts are supported), the parking lot (facility information), the slot (an individual space within a lot, if slot-level tracking is implemented), the tariff (pricing rules for a lot or slot type), the vehicle (the driver's registered vehicle details, where supported), the reservation (association of a driver and vehicle with a slot for a time window) and the occupancy update (status or counts with timestamps, when history is required). The actual database structure may differ from this proposal, and the entities should not be read as evidence that every table or collection is already implemented.

Security and Validation. User input is validated before changes to parking records are accepted, administrative operations are restricted to authorized users, and sensitive information such as vehicle registration numbers and booking history is not exposed through public interfaces. Occupancy updates should be associated with an authorized sensor, device or attendant, and validation must also run on the server because frontend checks alone can be bypassed. Secure credential handling, access-control checks and appropriate error handling complete the protections.

IV. SYSTEM IMPLEMENTATION

The methodology is realized as a web application with the following modules.

Development Environment. The application is developed in a JavaScript-based workflow and started through npm project scripts. A JavaScript runtime such as Node.js [14] is a typical basis for the backend, and a relational database such as PostgreSQL [16] is a typical choice for persistence. The roles are as follows: the frontend renders screens and handles interaction through standard client-side web APIs [17]; the backend or API processes requests and applies application rules; the database persists parking records; the mapping service displays lots on a map; and the occupancy source supplies slot status or counts together with timestamps.

User Interface Module. The interface provides navigation between the available features and presents parking information clearly, allowing each user to perform the actions their permissions support. A consistent interface includes meaningful labels, loading indicators, validation messages and suitable responses to failed requests.

Administrative Management Module. This module centralizes the maintenance of lot, slot and tariff information and, where implemented, lets authorized users create or modify records and review parking data. Validation prevents incomplete or inconsistent information from being stored; for example, a reservation must refer to an existing slot and a free time window.

Lot and Slot Information Module. This module retrieves lot details and slot information from the configured data source, displays them in a readable format and reflects updates made through authorized management operations. Its quality depends on the consistency of the stored data and the reliability of communication between the interface and the backend.

Availability and Navigation Module. The availability page gives a visual representation of free and occupied spaces. It identifies the lot to be monitored, requests the latest available occupancy data, validates the returned counts and timestamp, displays the status on the map, refreshes it according to the configured update mechanism, and handles missing, invalid or outdated data. If the occupancy source is temporarily unavailable, an appropriate status is shown. The page communicates the freshness of the displayed availability and does not imply real-time monitoring unless updates are actually being received. Once a lot is chosen, the driver can be guided to it through the mapping service, and entrance or route markers may be added where the map service and lot data support them.

Backend and API Integration. If a separate backend is used, the frontend communicates with it through API requests that retrieve parking records, submit updates and obtain the latest availability [10]. The backend validates request parameters and enforces authorization rules before changing stored information, and handles unavailable services, invalid requests and missing records in a predictable manner.

Database Integration. The database provides persistent storage, and relationships and validation rules preserve consistency between lots, slots, tariffs and reservations. If occupancy history is required, timestamped records are stored; if only the latest status is needed, a lighter storage approach may be used. Reservation handling should prevent two drivers from booking the same slot for overlapping periods, for example through transactions or uniqueness constraints.

Error Handling. The application handles an unavailable backend, records that are not found, invalid input, map or location-service failures, missing occupancy data, delayed occupancy updates, conflicting reservations and network interruptions. Clear status messages tell users whether a problem concerns the application, the network or the availability of occupancy data.

Testing Methodology. The following tests are proposed for the completed system. They are proposed test cases, not claims that all tests have passed.

Open the application: the initial screen loads correctly.

Retrieve lot and slot details: valid lot information and the correct tariff are displayed.

Open the availability page: the availability interface loads.

Receive valid occupancy data: the displayed status reflects the reported occupancy.

Receive no occupancy data: the interface indicates that availability data is unavailable.

Receive an old timestamp: the interface identifies the availability as outdated.

Submit invalid or conflicting input: the request, such as booking an already reserved slot, is rejected with an appropriate message.

Attempt an unauthorized update: the operation is denied.

Interrupt network access: the application handles the failure without crashing.

V. ADVANTAGES

1. Centralized Parking Information. A common digital environment replaces scattered registers and separate communication channels.

2. Improved Availability Visibility. When connected to a functioning occupancy source, the availability interface helps drivers see the last reported number of free spaces before they reach a lot.

3. Reduced Search Time and Manual Coordination. Centralized access to lot and availability information may reduce the time spent circling for a space and the repeated inquiries to attendants about capacity and tariffs.

4. Better Accessibility. A web-based interface makes information available from supported devices without consulting several records.

5. Improved Data Organization. Structured lot, slot and tariff records are easier to retrieve, update and maintain.

6. Extensibility. A modular architecture lets future features be developed independently, provided that the existing interfaces and data structures support integration.

7. Support for Decision-Making. If reliable occupancy history and operational records are collected, the system may later support analysis of peak hours, utilization and demand.

8. Communication Potential. The application is a foundation for notifications about lot closures, tariff changes or reservation reminders. These advantages describe the intended benefits of the design, and their actual impact must be assessed through testing and user feedback.

VI. RESULTS AND ANALYSIS

This section reports the development status, the limitations and the proposed evaluation. No benchmark, occupancy log or user study has been completed, so no numerical performance results are claimed and no claim is made that end-to-end real-time availability monitoring has been demonstrated.

A. Development Status. A parking assistant application is under development, and recent work has concentrated on slot management and the availability interface. A verified feature inventory, test logs and screenshots are still to be recorded.

B. Functional Analysis. Functional testing should verify that each implemented module meets its requirements. Lot and slot information should be consistent across screens, and updates should appear correctly once saved. A status shown on the map is valid evidence of availability monitoring only when it corresponds to a real occupancy record: a static status or demonstration data can validate the interface but cannot establish that live monitoring works end to end.

C. Live-Availability Analysis. The availability module should be assessed with actual occupancy data from a configured sensor, device or service. The key factors are update delay (time between measuring and displaying an occupancy change), count accuracy (agreement with a manual count of free spaces), update continuity under normal network conditions, data freshness, and failure recovery after a temporary interruption. These measures distinguish a functioning availability service from a page that only displays map elements.

D. Performance Analysis. Response times should be examined under normal use and under increasing request loads, identifying whether delays arise in the frontend, backend, database, network or external mapping service. Measurements should come from repeatable tests documented with their conditions.

E. Limitations. (i) Genuine live availability depends on an operational occupancy source and a reliable update mechanism. (ii) Sensing accuracy varies with the device, environment and installation conditions, and GPS positioning is unreliable inside covered or underground lots. (iii) Network interruptions can delay or prevent updates. (iv) A map interface alone is no evidence of availability prediction or optimal slot allocation. (v) The completeness of lot, slot and tariff information depends on the data maintained by the operator. (vi) Security and access control must be validated against the actual implementation. (vii) The effectiveness of the system cannot be quantified without test results and user evaluation.

F. Proposed Evaluation Protocol. Each component should be evaluated against a defined reference, as listed below.

Application functionality: number of functional test cases passed.

Lot and slot management: successful creation, retrieval and updating of valid records.

Live availability: successful receipt and display of actual occupancy updates.

Availability freshness: time elapsed between occupancy measurement and display.

API performance: response time for representative requests.

Error handling: correct handling of network and service failures.

Access control: results of authorized and unauthorized operation tests, following the OWASP Top 10 [13].

Usability: feedback from representative users.

G. Discussion. ParkIt addresses a practical parking-management problem by combining structured lot information with a dedicated availability interface, which can improve access to parking information and provide a foundation for real-time monitoring. The quality of the final system depends on the integration of the frontend, data services and occupancy source, and the availability interface must distinguish current information from unavailable or outdated data. The project should therefore be evaluated as an integrated system rather than as a collection of screens, and functional testing, occupancy verification, security checks and usability assessment are required before conclusions are drawn about its effectiveness.

VII. CONCLUSION

This paper presented PARKIT – A Parking Assistant System, an application that brings lot details, slot information and a live availability interface into one digital platform. It aims to reduce the time drivers spend searching for parking, simplify the management of parking records and give drivers a convenient way to access information about available spaces. Its modular architecture leaves room for additional parking features in later versions.

The live-availability component is the most important part of the system, since it can provide useful information when connected to a reliable occupancy

source. Its effectiveness depends on the accuracy, freshness and availability of the received data and on correct handling of network and service failures. The project is a developing solution rather than a validated product: the completed feature set and measured performance still have to be verified through systematic testing, and we have therefore stated its limitations and proposed an evaluation protocol.

VIII. FUTURE WORK

Enhanced Live Availability. Reliability can be improved with dedicated sensors or camera-based detection, together with displayed timestamps, update status and warnings when the reported availability becomes outdated.

Availability Prediction. Occupancy at a given time can be predicted from historical records, the time of day, the day of the week and local events, and should be validated against observed availability.

Push Notifications. Notifications can inform drivers about reservation expiry, lot closures and tariff changes, and should be connected to verified operational events.

Route and Slot Guidance. Guidance to the nearest suitable lot and slot can be investigated using road networks, the driver's position, slot type and walking distance to the destination [8], [9], which requires an appropriate routing model and reliable input data.

Online Payment. Payment for reserved or used parking time could be integrated through an authorized payment gateway, with secure handling of transaction records and controlled access to billing information.

License Plate Recognition. Entry and exit events could be recorded through camera-based plate recognition, QR codes or RFID tags, allowing automatic billing and overstay detection, provided that the organization approves the mechanism and protects the data collected.

Historical Occupancy Analytics. Storing timestamped occupancy records would let administrators study peak hours, utilization and demand patterns.

Security and Privacy. Further work should include access-control testing, secure communication, appropriate retention of vehicle and booking records, and clear restrictions on who can view driver and vehicle information.

Automated Testing, Deployment and Scalability. Automated tests can cover the principal workflows, API validation, database operations and tracking-state transitions, with integration and load testing before deployment. The application can then be prepared for institutiona1 use through production hosting, monitoring, backups and recovery procedures.

IX. REFERENCES

[1]. Sussman, J. M. (2005). Perspectives on Intelligent Transportation Systems (ITS). Springer. https://doi.org/10.1007/b136068

[2]. Shoup, D. C. (2006). Cruising for parking. Transport Policy, 13(6), 479–486. https://doi.org/10.1016/j.tranpol.2006.05.005

[3]. Shoup, D. C. (2005). The High Cost of Free Parking. APA Planners Press.

[4]. Idris, M. Y. I., Leng, Y. Y., Tamil, E. M., Noor, N. M., & Razak, Z. (2009). Car park system: A review of smart parking system and its technology. Information Technology Journal, 8(2), 101–113.

[5]. Lin, T., Rivano, H., & Le Mouël, F. (2017). A survey of smart parking solutions. IEEE Transactions on Intelligent Transportation Systems, 18(12), 3229–3253. https://doi.org/10.1109/TITS.2017.2685143

[6]. Barriga, J. J., Sulca, J., León, J. L., Ulloa, A., Portero, D., Andrade, R., & Yoo, S. G. (2019). Smart parking: A literature review from the technological perspective. Applied Sciences, 9(21), 4569. https://doi.org/10.3390/app9214569

[7]. Kaplan, E. D., & Hegarty, C. J. (Eds.). (2017). Understanding GPS/GNSS: Principles and Applications (3rd ed.). Artech House.

[8]. Delling, D., Sanders, P., Schultes, D., & Wagner, D. (2009). Engineering route planning algorithms. In J. Lerner, D. Wagner, & K. A. Zweig (Eds.), Algorithmics of Large and Complex Networks (pp. 117–139). Springer.

[9]. Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2022). Introduction to Algorithms (4th ed.). MIT Press.

[10]. Fielding, R. T. (2000). Architectural Styles and the Design of Network-Based Software Architectures (Doctoral dissertation). University of California, Irvine.

[11]. World Wide Web Consortium (W3C). (2024). Geolocation API Specification. https://www.w3.org/TR/geolocation/

[12]. OpenStreetMap Contributors. (n.d.). OpenStreetMap Documentation. https://wiki.openstreetmap.org/

[13]. Open Worldwide Application Security Project (OWASP). (2021). OWASP Top 10:2021 — The Ten Most Critical Web Application Security Risks. https://owasp.org/Top10/

[14]. OpenJS Foundation. (n.d.). Node.js Documentation. https://nodejs.org/docs/latest/api/

[15]. Google. (n.d.). Google Maps Platform Documentation. https://developers.google.com/maps/documentation

[16]. PostgreSQL Global Development Group. (n.d.). PostgreSQL Documentation. https://www.postgresql.org/docs/

[17]. Mozilla. (n.d.). MDN Web Docs: Client-side Web APIs. https://developer.mozilla.org/en-US/docs/Web/API