PROJECT ABSTRACT

Parking Spot Saver Android App

SpotHold is an Android app for the most annoying part of city parking: the spot you saw is gone by the time you circle back. Tap a free bay on the live lot map and the app holds it for 15 minutes while you arrive.

Project type: Software (Android application)

Availability: Built to order

Suitable branches: Computer / IT, Android

Technologies: Kotlin, Android Studio, Firebase Realtime DB, Firebase Authentication, Material Design 3, MVVM, Cloud Messaging alerts, Git

Prepared by Projectech · projectech.in — for academic project reference.

2. Introduction / Background

In mall and office parking lots the failure mode is always the same: a driver spots a free bay on the lot display, drives toward it, and finds a car physically closer has taken it. The map and the ground truth diverge in the two minutes that matter. Existing parking apps show availability but cannot reserve — they report the world without changing it.

SpotHold closes that gap with a hold, not just a map: tapping a free spot locks it for 15 minutes, visible to every other user as “held” so nobody is routed to it. The countdown is honest — arrive before it ends or the spot releases automatically. Check-in on arrival converts the hold into a parked session, and a single 10-minute extension covers traffic-light delays. Zones are tiered by walking distance with live free counts, and fairness is structural: one active hold per user, holds are free.

3. Problem Statement

Availability displays without reservation create a race: every driver sees the same “free” bay and the closest car wins, which punishes anyone arriving from further away and generates circling traffic inside the lot. Paid advance booking solves the wrong problem — nobody wants to pay to reserve a mall bay an hour ahead; the need is a free, short bridge between seeing the spot and reaching it.

That bridge must be fair and self-cleaning. Without per-user limits, drivers would hold bays all day; without automatic expiry, forgotten holds would freeze bays permanently. This project builds the hold as an atomic

backend transaction with countdown, one-time extension, auto-release and a fairness rule of one active hold per user, all visible on a live zone map.

4. Objectives

- Show a live zone-wise lot map with free, held and occupied bays updating in real time.
- Let drivers hold a free bay for 15 minutes while they drive in, free of charge.
- Convert the hold to a parked session with one-tap check-in on arrival.
- Allow a single 10-minute extension per hold, then auto-release at zero.
- Enforce one active hold per user as a structural fairness rule.
- Keep a hold history (completed, released, expired) with timestamps.

5. Proposed Methodology

The lot is modelled as zones containing bays, each bay carrying a live state (free/held/occupied) in a Firebase Realtime Database. The hold is the critical operation: it is implemented as an atomic check-and-lock transaction on the bay record, so two drivers tapping the same free bay cannot both succeed — only the first write wins and the other sees it flip to held.

The Kotlin app (MVVM, Material Design 3) renders the zone map, spot detail with the hold timer, and the My Holds screen with history. A countdown drives the hold lifecycle client-side against the server timestamp; expiry warnings fire via Cloud Messaging at 5 and 1 minute remaining; at zero the bay auto-releases. The fairness rule (one active hold per user) is enforced in backend transactions, not in the UI. Testing covers the concurrent-hold race, expiry timing, extension limits and the check-in state transitions.

6. System Architecture / Working

The architecture is mobile + realtime backend. The Android client (Kotlin, MVVM) shows the lot map, spot detail with countdown, and hold history. Firebase Realtime Database holds bay states, holds and history with transaction rules enforcing atomicity and the one-active-hold limit; Firebase Authentication signs drivers in and Cloud Messaging delivers expiry warnings.

Working flow: (1) the operator maps bays into zones with walking-distance tiers; (2) the driver opens the zone map and sees live free counts; (3) tapping a free bay creates the hold as an atomic transaction and the bay flips to held for everyone; (4) the 15:00 countdown ticks on the spot screen while the driver navigates over; (5) on parking, one tap checks in and the hold becomes a parked session; (6) if delayed, the single 10-minute extension applies, otherwise the bay auto-releases at zero; (7) the history screen logs the outcome and blocks a second concurrent hold.

7. Hardware / Software Requirements

- Device: Android 8.0 (API 26) or above; connectivity required to create or release holds.
- Backend: Firebase project (Realtime Database, Authentication, Cloud Messaging).

- Data: the lot's bay layout and zone mapping from the operator.
- Development: Android Studio, Kotlin, Git.

8. Expected Outcomes

- Drivers can lock a bay for the drive-in gap instead of racing for it.
- Every user sees the same live truth: free, held and occupied bays.
- Forgotten holds self-clean at zero with no manual intervention.
- The fairness rule keeps the system usable at peak hours.
- Complete documentation: project report, presentation and viva Q&A.

9. Applications

- Mall and multiplex parking lots.
- Office and tech-park visitor parking.
- Airport and railway station pick-up zones.
- Hospitals with constrained patient parking.

10. Future Scope

- Multi-lot city networks under one account.
- Sensor or camera-based automatic occupancy detection.
- Turn-by-turn guidance to the held bay inside the lot.
- EV-charging bay holds with charger reservations.
- Dynamic hold windows priced by demand.