ParkVision 2.0 Requirements

OptiCore

October 2025

Core Additions/Upgrades

AI/CV	YOLO-based car detection	Replace OpenCV background subtraction with a YOLOv8/YOLOv9 model for reliable detection in diverse conditions (lighting, weather, perspective).
AI/CV	Licence Plate Recognition	Integrate LPR (e.g., EasyOCR or OpenALPR + fine-tuned model) to capture entry/exit times per vehicle.
AI/CV	Spot association logic	Link YOLO bounding boxes with known parking spot coordinates for occupancy tracking.
Backend	Car session tracking	Track when each vehicle enters/leaves the lot and calculate time spent for analytics or metered pricing.
Backend	Analytics API	Provide endpoints for statistics: average occupancy, peak times, most-used spots, revenue projection, etc.
Frontend	Admin dashboard overhaul	Add an interactive parking lot schematic (SVG or grid overlay), occupancy visualization, and analytics charts.
Frontend	Spot type configuration	Allow tagging spots as ”reserved”, ”metered”, ”disabled”, etc, via UI.
Frontend	Authentication and roles	Implement login for Admins / Managers (JWT or OAuth)
General	Deployment and CI/CD	Containerized YOLO inference and FastAPI backend; use Docker Compose for unified startups.
Stretch Goal	Notification system	Alert admins if lot reaches high occupancy or if vehicles overstay.

System Architecture

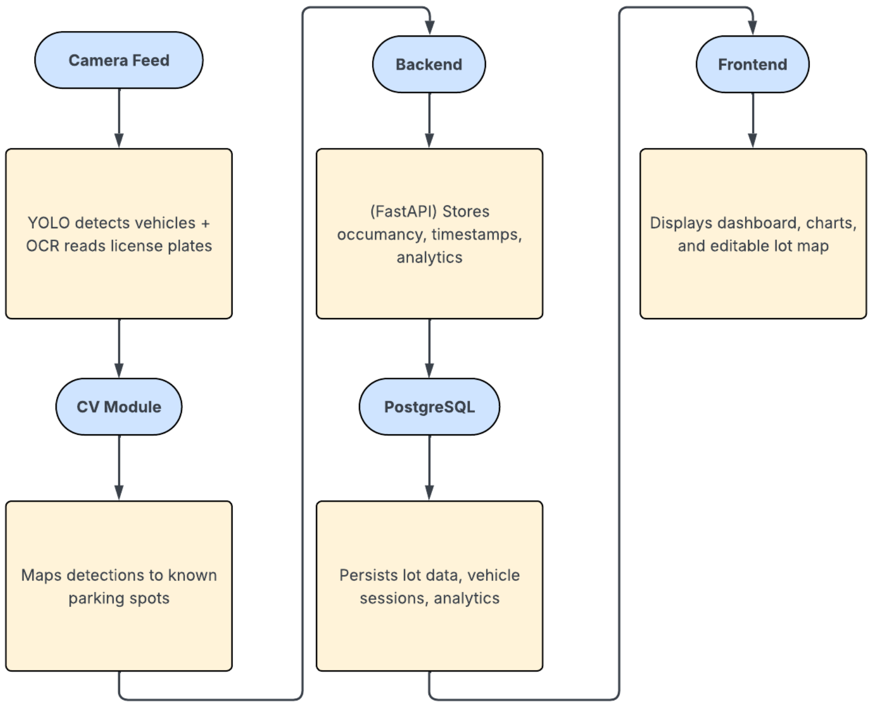
Camera Feed
YOLO detects vehicles + OCR reads license plates
CV Module
Maps detections to known parking spots
Backend
(FastAPI) Stores occumancy, timestamps, analytics
PostgreSQL
Persists lot data, vehicle sessions, analytics
Frontend
Displays dashboard, charts, and editable lot map

Figure 1: The suggested high-level flow of ParkVision 2.0

Team Roles

1. AI/CV

Goals:

- Train/deploy YOLOv8 model for car + parking space detection.
- Implement license plate recognition (EasyOCR or OpenALPR).
- Integrate vehicle tracking to persist vehicle sessions.
- Export real-time occupancy to backend via API.

Tasks:

☐ Collect + annotate sample data for fine-tuning YOLO

- ☐ Train YOLOv8 using Roboflow or local GPU (or find pre-trained models)
- ☐ Develop Python scripts for detection pipeline.
- ☐ Integrate LPR model and link license plate to vehicle's session
- ☐ Test under different lighting/weather conditions, etc.

2. Backend

Goals:

- Create endpoints for lots, occupancy, analytics, and authentication.
- Develop logic for time tracking and vehicle session management.
- Build analytics layer (average occupancy, duration, revenue estimates, etc.).
- Coordinate data flow between AI module and database.

Tasks:

- ☐ Design database schema v2
- ☐ Implement `/api/analytics`, `/api/auth`, and `/api/lot` routes.
- ☐ Write services to process and aggregate data.
- ☐ Unit test endpoints and database operations.
- ☐ Containerize backend (Docker + Compose).

3. Frontend

Goals:

- Build an admin dashboard with visual lot overlay.
- Display real-time occupancy, vehicle info, and analytics.
- Enable configuration of spot types and lot metadata.
- Add authentication + session persistence.

Tasks:

- ☐ Build interactive SVG lot map (I'm thinking drag-and-drop spot editing).
- ☐ Integrate REST API endpoints for occupancy and analytics.
- ☐ Add charts (likely Charts.js) for trends.
- ☐ Implement login + admin roles (like JWT tokens).
- ☐ Polish UI/UX.

Sprint Schedule

Sprint	Timeline	Focus	Deliverables
Q1	Now - 10/8	Project setup, update schema	Updated DB schema, base API routes, YOLO test pipeline
Q2	10/9 - 10/21	Core detection and backend integration	Car/plate detection integrated with backend
Q3	10/22 - 11/5	Admin dashboard and analytics	Frontend dashboard with live occupancy charts
Q4	11/6 - 11/19	Final integration and polish	Full demo (AI $\rightarrow$ API $\rightarrow$ UI), documentation, poster presentation

Extra Feature Suggestions

- Occupancy heatmap over time.
- Revenue projection for metered lots.
- Camera calibration tool in UI for perspective adjustment.
- Edge deployment support (run YOLO inference on Rasberry Pi).
- API key management for external integration/deployment.