Use Cases

Driver Use Cases

Name:	**Entry — Not Listed (Payment Deferred)**
Actor:	Driver
Description:	Driver enters when plate is neither whitelisted nor blacklisted; session opens; payment deferred to exit.
Pre-Condition:	camera operational.
Scenario:	1) Camera captures plate at **entry** (timestamp, read quality). 2) System confirms plate is not on lists. 3) System **opens a session** for the plate. 4) System commands barrier to **Open**; barrier raises. 5) Optional display: "Welcome — pay at exit."
Result:	Session **open**; vehicle inside; no payment yet.
Extensions:	1) Low read quality → 1.1 allow entry -¿ 1.2 session flagged for confirmation at exit.
Exceptions:	1) Camera offline → 1.1 local policy decides (fail-safe open/queue).

Name:	**Entry — Whitelisted (Subscription)**
Actor:	Driver
Description:	Recognized subscriber enters automatically; fee waived/discounted per policy.
Pre-Condition:	The vehicle should be whitelisted.
Scenario:	1) Camera captures plate at **entry** (timestamp, **read quality**). 2) System matches whitelist and checks expiry. 3) System **opens** (or logs) a zero-fee/discounted session per policy. 4) System commands barrier to **Open**; display "Subscription recognized."
Result:	Entry granted; session tracked (fee waived/discounted).
Extensions:	2) Subscription nearing expiry 2.1 System displays heads-up in portal.
Exceptions:	2) Subscription expired 2.1 fall back to "Not Listed" entry flow.

Name:	**Entry — Blacklisted (Denied)**
Actor:	Driver
Description:	Access is denied for blacklisted plates.
Pre-Condition:	Plate exists on blacklist.
Scenario:	1) Camera captures plate at **entry** (timestamp, **read quality**). 2) System matches blacklist. 3) System does not open; event is logged. 4) Lane display: "Access denied."
Result:	No session opened; car remains outside.
Extensions:	1) Very low read quality → 1.1 prompt re-approach 1.2 still deny on match.
Exceptions:	1) Misread reported by driver via help URL 1.1 flagged for audit review. 1.2 Returns to step 1

Name:	**Exit — Scan QR, Pay in Dashboard, Barrier Opens**
Actor:	Driver
Description:	Driver pays at exit via self-service dashboard and the barrier opens automatically afterward.
Pre-Condition:	Open **unpaid** session exists for plate; exit camera online.
Scenario:	1) Camera reads plate at **exit** (timestamp, **read quality**); system finds open unpaid session; barrier remains closed. 2) Lane display shows **QR** to self-service dashboard. 3) Driver scans QR; dashboard shows session with **duration** and **fee**. 4) Driver initiates **card payment**; payment service authorizes/captures. 5) System marks session **paid** and shows **receipt** (amount, plate, duration). 6) System commands barrier to **Open**; barrier raises; session **closes**.
Result:	Payment completed; receipt displayed; barrier opens; session closed.
Extensions:	4) Network lag after payment 4.1 dashboard shows **exit token/QR** 4.2 barrier validates and opens once confirmed.
Exceptions:	4) Payment failed/declined 4.1 error shown retry allowed 4.2 barrier stays closed until paid.

Name:	**Subscribe Vehicle / Register for Whitelist**
Actor:	Driver
Description:	Driver self-registers a license plate for a subscription so the plate is added to the whitelist (auto-open, fee waived/discounted per policy) **after admin authentication**.
Pre-Condition:	Self-service dashboard reachable; subscription plans available; payment service available. Plate is **not** currently on an active whitelist subscription. **Admin must authenticate the license plate before activation.**
Scenario:	1) Driver opens **Subscription** in the dashboard (via QR/URL). 2) Driver enters **license plate** and contact details (email/phone), accepts terms. 3) Driver selects a **plan** (e.g., monthly), chooses **start date** (now or future), and **auto-renew** option. 4) System checks for **existing/overlapping** subscriptions and shows summary. 5) Driver proceeds to **pay**; payment service authorizes/captures. 6) System creates a **subscription = Pending Admin Authentication** (no whitelist privileges yet). 7) **Admin authenticates** the plate. 8) System **activates whitelist entry** with policy (auto-open; fee waived/discounted) and **expiry** per plan; dashboard shows **confirmation/receipt** and status: "Subscription active."
Result:	Plate is **whitelisted** with the chosen policy and validity window after admin authentication; receipt delivered; future entries will auto-open.
Extensions:	3) **Future start date** 3.1 subscription marked **Scheduled + Pending**; 3.2 admin authenticates and the start date arrives, entries follow "Not Listed." 7) **ownership verification** (email/SMS code or next-lane read with good read quality) to assist admin review.
Exceptions:	4) Plate **already whitelisted** 4.1 system blocks duplicate and shows existing subscription with renewal options. 5) Payment failed/declined 5.1 subscription not created; driver returns to step 3 . 2) Invalid plate format 2.1 inline validation prevents continuation. 2.2 drivers reenters the plate step 2 7) **Admin rejects authentication** 7.1 subscription marked **Rejected**; notify driver and follow refund/cancellation policy.

Name:	**View Subscription & History**
Actor:	Driver (Whitelisted)
Description:	Driver reviews current subscription status (plan, active/expiry) and sees past parking activity/receipts in the self-service dashboard.
Pre-Condition:	Plate is on whitelist; dashboard reachable.
Scenario:	1) Driver opens **Subscriber Portal** (QR/URL). 2) Verifies plate (simple lookup or one-time code). 3) System shows **subscription status** (plan, expiry, auto-renew). 4) System shows **history** with receipts/download.
Result:	Driver has visibility of subscription and past sessions; receipts can be viewed/downloaded.
Extensions:	3) Offer renewal/upgrade links (future); export history (CSV/PDF).
Exceptions:	3) No active subscription 3.1 Current status is shown and guidance 3.2 history still visible.

Admin Use Cases

Name:	**Create Report Templates**
Actor:	Admin
Description:	Define reusable report blueprints (e.g., Revenue by Period, Occupancy by Hour).
Pre-Condition:	Admin authenticated.
Scenario:	1) Admin opens **Reports → Templates**. 2) Chooses report type and sets parameters (defaults, filters, columns). 3) Optionally adds a chart visualization. 4) Saves template.
Result:	Template saved and available for future runs; change is audit-logged.
Extensions:	2) Copy existing template 2.1 edit 2.2 save as new.
Exceptions:	2) Missing/invalid parameters 2.1 inline validation prevents save.

Name:	**Access & Run Reports**
Actor:	Admin
Description:	Generate reports from saved templates for a chosen period and export results.
Pre-Condition:	At least one template exists; data present for selected period.
Scenario:	1) Admin opens **Reports → Run**. 2) Selects a template and period/filters. 3) Clicks **Run**; system renders table (and chart if defined). 4) Admin exports to CSV/PDF. 5) System audit-logs the run.
Result:	Report displayed and downloadable.
Extensions:	-
Exceptions:	3) Sparse/no data informational notice 3.1 export still allowed.

Name:	**Authorize/Block Vehicles (Whitelist/Blacklist)**
Actor:	Admin
Description:	Manage plates that auto-open (whitelist) or are denied (blacklist); optional expiry/labels.
Pre-Condition:	Admin authenticated.
Scenario:	1) Admin opens **Lists**. 2) Adds/edits/removes entries (plate, label, expiry, policy). 3) System updates matching cache immediately. 4) System audit-logs the change.
Result:	Access policies take effect for future camera recognitions.
Extensions:	-
Exceptions:	2) Duplicate plate detected 2.1 save blocked by the system 2.2 System links to existing record.

Name:	**Configure Access Roles & Authentication**
Actor:	Admin (owner)
Description:	Manages admin users and roles (Owner/Admin/Viewer) and defines authentication policies (login methods, MFA—optional), credential lifecycle, and session controls.
Pre-Condition:	Owner-level admin is authenticated; at least one Owner must always exist.
Scenario:	1) Admin opens **Users, Roles & Auth**. 2) Sets **auth policy**: allowed login method(s) (email/password), session timeout, password rules. 3) **Invites** a user (email), assigns **role**, selects auth method. 4) User **verifies email** and completes **MFA** enrollment (if required). 5) System **saves** and begins enforcing the updated policies. 6) Admin may **change role**, **suspend** a user, **revoke sessions**, or **reset credentials**.
Result:	Access list updated; auth policies enforced; invited users can sign in via chosen method; existing users inherit new requirements (e.g., MFA); revoked sessions are terminated.
Extensions:	3) Enforce email-domain allowlist for invites 3.1 configurable **MFA period**.
Exceptions:	3) Invite expired/rejected 3.1 re-send invite; invalid email blocked. • 6)Attempt to remove the last Owner 6.1 blocked with clear message. • 6)Role change that would lock out all Owners 6) The system displays explicit confirmation.

Name:	**Adjust Prices (Change Pricing via Dashboard)**
Actor:	Admin
Description:	Admin updates pricing rules (rates, blocks, grace, rounding) that determine session fees.
Pre-Condition:	Admin is authenticated; existing pricing set available.
Scenario:	1) Open **Pricing** in dashboard. 2) Create/edit a pricing rule (values + effective start/end). 3) System validates conflicts and previews sample fees. 4) Save changes; rule activates per effective date and is audit-logged.
Result:	New/updated rule stored; future fee calculations use it.
Extensions:	2) Save as **Inactive** to activate later 2.1 User duplicates an existing rule and edit.
Exceptions:	4) Overlapping active periods 4.1 System blocks save and highlights the conflicts.