Use Case User/Admin Operations - Hệ Thống Quản Lý Bãi Đỗ Xe Thông Minh

Mô Tả Tổng Quát

File này mô tả các thao tác của Admin và User (staff) trên giao diện Frontend (web/desktop) để tương tác với hệ thống parking. Bao gồm login, dashboard, quản lý thẻ, xem logs, thống kê. **Lưu ý:** Config Settings (UC-8) sẽ được implement ở version 2.0.

Use Case 1: User Login qua Frontend Web/Desktop

Trường	Nội Dung
Tên Use Case	User Login - Đăng Nhập vào Web/Desktop App
Mô Tả	User nhập username & password ở login page. Frontend validate, gửi Backend. Nếu thành công, nhận JWT token, lưu localStorage, redirect dashboard. Desktop app cũng dùng localStorage (tương tự web).
Tác Nhân	User (Admin hoặc Staff), Frontend App (web hoặc desktop), Backend API
Mô Tả Chi Tiết	LoginPage component render form. User submit. Frontend gọi parkingApi.login(username, password). Backend verify, return token. Frontend store token, redirect Dashboard. Desktop app lưu token secure. Mỗi request sau, token gắn vào Authorization header.
Điều Kiện Tiên Quyết	Frontend app khả dụng (http://localhost:3000 hoặc desktop app); User account tồn tại ở Backend; Backend running

Trường	Nội Dung
Luộng Chính	**Web Frontend:** 1. User vào http://localhost:3000 2. Router detect no token, redirect /login 3. LoginPage component render: - Input username - Input password - Button "Sign In" 4. User nhập: username="admin", password="admin123" 5. Nhấn "Sign In" 6. Frontend validate: - Check username not empty - Check password not empty 7. Frontend gọi parkingApi.login("admin", "admin123") 8. Axios POST /api/auth/login (auto detect localhost:5000) 9. Backend xử lý (ref: Backend_System_Usecase UC-1) 10. Nhận response 200: `{message, token, user}` 11. Frontend lưu token: localStorage.setItem("token", token) 12. Frontend lưu user info: useState(user) 13. Frontend redirect: navigate("/dashboard") 14. Dashboard component mount, render Tabs (Dashboard, Cards, Parking, Logs, Admin) **Desktop Electron App:** 15. Desktop app window load React component 16. LoginPage component render (tương tự web) 17. User submit 18. Frontend gọi parkingApi.login (axios query localhost:5000) 19. Backend response với token 20. Electron app: Desktop app sử dụng localStorage (tương tự web) để lưu token 21. Desktop app redirect dashboard 22. Main.js auto-start backend subprocess nếu cần
Luộng Thay Thế	A1: Remember me - Lưu token lâu hơn 24h (hoặc refresh token) A2: Biometric login - Face/Touch ID thay password (mobile/desktop) A3: OAuth - Login bằng Google/Facebook account A4: RFID card login - Admin scan RFID card thay username/password
Luộng Ngoài Lề	E1: Backend offline - Axios timeout 10s, hiển thị "Backend connection failed" E2: Wrong password - Backend 401, frontend hiển thị "Invalid credentials" (không reveal user exist) E3: User deactivated - Backend 401, message "Account disabled" E4: Token expired - Nếu token hết 24h, auto logout, redirect login E5: Brute force - Không implement rate limiting (VẤNĐỀ), frontend có thể add client-side throttle E6: HTTPS - Web production nên dùng HTTPS, token ở localStorage có risk (XSS), desktop app safer

Trường	Nội Dung
Điều Kiện Sau	User logged in, token stored, Dashboard display; Frontend ready cho requests; JWT gắn vào header

Use Case 2: Admin Dashboard - Tổng Quan Hệ Thống

Trường	Nội Dung
Tên Use Case	Admin Dashboard - Xem Tổng Quan Hệ Thống
Mô Tả	Admin vào dashboard, thấy KPI cards (total cards, inside, outside, unknown cards), charts (occupancy trend, busy hours), real-time parking slots grid (6 slots), system status, alerts.
Tác Nhân	Admin User, Frontend Dashboard Component, Backend API
Mô Tả Chi Tiết	Dashboard component mount, gọi 3 API song song: getSlots(), getStatistics(), getSystemHealth(). Render KPI cards, charts (Chart.js), 6-slot grid, status indicator. Auto-refresh: slots mỗi 10s, stats mỗi 30s, health mỗi 60s.
Điều Kiện Tiên Quyết	Admin JWT token valid; Dashboard component mounted; Backend APIs running
Luộng Chính	1. Admin login thành công, redirect /dashboard 2. Dashboard.tsx component mount 3. useEffect(() => { fetchData() }, []) gọi 3 APIs song song: - getSlots(): GET /api/parking-slots/ - getStatistics(): GET /api/cards/statistics - getSystemHealth(): GET /api/system/health 4. Parallel axios calls, timeout 10s 5. Responses nhận vào useState: - slotData: {occupied, available, occupancy_rate, slots: [...]} - stats: {total_cards, inside, outside, unknown, avg_duration} - health: {healthy, services: {...}} 6. Render JSX: **Section 1: KPI Cards (4 cards)** 7. Total Cards: Display "100 cards" 8. Inside Now: Display "45 inside (45%)" with green color 9. Outside Now: Display "55 outside (55%)" with gray color 10. Unknown Cards: Display "5 unknown" with red/warning color **Section 2: Occupancy Trend Chart (24h)** 11. Chart.js line chart

Trường	Nội Dung

	- X-axis: time 0-24h
	- Y-axis: occupancy % 0-100%
	- Fetch từ backend (UC-3 Backend: UC-7) để get historical data
	- Render line trend
	
	Section 3: Busy Hours Chart
	12. Bar chart: jam 0-23
	13. Chiều cao bar = số lần entry/exit
	14. Identify busy hours (e.g., 8-9am, 17-18pm)
	
	Section 4: Parking Slots Grid (6 slots)
	15. Render 2x3 grid (6 ô)
	16. Mỗi ô: slot_id 1-6
	17. Color: green=empty (status=0), red=occupied (status=1)
	18. Display distance nếu có (e.g., "15cm", "5cm")
	19. Click on slot: open detail popup
	
	Section 5: System Status
	20. Backend: green circle = healthy, red = error
	21. ESP32: green/red indicator
	22. UNO R4: green/red indicator
	23. WiFi: connected / disconnected
	
	Section 6: Alerts (nếu có)
	24. Alert cards:
	- "Unknown cards detected: 5 scans in last 1h"
	- "Sensor IN unreliable: check connection"
	- "Database size: 250MB (healthy)"
	
	Auto-refresh timers:
	25. setInterval(() => { refetch slots }, 10000) // 10s
	26. setInterval(() => { refetch stats }, 30000) // 30s
	27. setInterval(() => { refetch health }, 60000) // 60s
	
	Error handling:
	28. Nếu API fail, show "Loading..." hoặc last cached data
	29. Nếu backend offline (health=unhealthy), show red banner "System Degraded"
	30. Nếu multiple fail, show error modal

Luộng Thay Thế	A1: Minimalist dashboard - Chỉ hiển thị occupancy_rate, hide charts A2: Customizable dashboard - User drag-drop widgets, save layout A3: Analog gauge - Thay progress bar, use analog meter gauge A4: Real-time WebSocket - Replace polling với WebSocket push

Trường	Nội Dung
Luộng Ngoài Lề	E1: Slow network - API slow, chart load delay, show skeleton loader E2: ESP32 offline - ESP32 health red, don't show slots data, display "ESP32 Offline" E3: Old data - Cache old, display "Last update: 2 min ago" E4: Memory leak - setInterval not cleanup, component unmount, call clearInterval E5: Infinite loop - Chart rerender, optimize useMemo/useCallback
Điều Kiện Sau	Dashboard fully loaded; Admin có overview hệ thống; Auto-refresh hoạt động

Use Case 3: Admin Quản Lý Thẻ Xe (Card Management)

Trường	Nội Dung
Tên Use Case	Card Management - Admin Add/Edit/Delete/Search Card
Mô Tả	Admin vào tab "Cards", xem danh sách phân trang, tìm kiếm, click để edit, hoặc nhấn nút Add/Delete. Form validation client-side + server-side.
Tác Nhân	Admin User, Frontend CardList Component, Backend API (/api/cards)
Mô Tả Chi Tiết	CardList component render paginated table. Each row: uid, name, card_type, status, entry_time, exit_time, actions (edit, delete, view logs). Top form: search box, add button. Edit/delete trigger modal atau navigate form page.
Điều Kiện Tiên Quyết	Admin JWT token valid; Card data available ở backend; CardList component mounted
Luộng Chính	**Initial Load - List Cards:** 1. Admin click "Cards" tab 2. CardList component mount 3. useEffect(() => { fetchCards() }, [page]) 4. Frontend gọi getCards(page=1, limit=10) 5. Backend GET /api/cards?page=1&limit=10 (ref: Backend UC-5) 6. Response: `{success: true, cards: [{uid, name, status, ...}], count: 150, message: "..."` 7. Render table: - Header: UID
Luộng Thay Thế	A1: Bulk import - CSV upload, parse, create many cards A2: Card expiry - Set expiry date, auto-disable A3: Card suspension - Suspend card (blacklist) thay delete A4: Card QR code - Generate QR từ UID, print untuk tài xế

Trường	Nội Dung
Luộng Ngoài Lề	E1: Duplicate UID - Add card với UID duplicate, backend reject 409 E2: Network timeout - Search slow, debounce prevent spam requests E3: Delete active card - Card inside (status=1), confirm modal warn E4: Validation fail - Client & server both validate, server authoritative E5: Concurrent edit - 2 admin edit cùng card, last-write-win
Điều Kiện Sau	Card list updated; Cards có thể add/edit/delete; Search hoạt động

Use Case 4: Admin Xem Unknown Cards (Thẻ Lạ)

Trường	Nội Dung
Tên Use Case	Unknown Cards - Admin Xem & Whitelist Thẻ Lạ
Mô Tả	Khi có thẻ lạ quét (không trong database), tự động log vào unknown_cards. Admin xem danh sách, có thể whitelist (thêm vào card list), hoặc ignore.
Tác Nhân	Admin User, Frontend UnknownCards Component, Backend API (/api/cards/unknown)
Mô Tả Chi Tiết	GET /api/cards/unknown fetch unknown_cards.json từ backend. Render table: uid, scan_count, first_seen, last_seen, actions (whitelist, delete). Modal form để whitelist: nhập name, type, save.
Điều Kiện Tiên Quyết	Admin JWT token valid; Unknown cards log available (unknown_cards.json hoặc database)
Luộng Chính	1. Admin click "Unknown Cards" từ sidebar 2. UnknownCards component mount 3. Frontend gọi getUnknownCards() 4. Backend GET /api/cards/unknown (ref: Backend UC-5) 5. Response: `{success: true, cards: [{uid, scan_count, first_seen, last_seen}, ...], message: "..."}` 6. Render table: - Header: UID
Luộng Thay Thế	A1: Auto-whitelist - System auto-whitelist sau N scans (configurable) A2: Notification - SMS/Email alert admin khi unknown card detect A3: Blacklist - Thay whitelist, admin add vào blacklist (deny entry)
Luộng Ngoài Lề	E1: Empty list - Không có unknown cards, display "No unknown cards" E2: Whitelist duplicate - Card UID duplicate với existing card, error E3: Network timeout - Fetch unknown list slow

Trường	Nội Dung
Điều Kiện Sau	Admin thấy unknown cards; Có thể whitelist hoặc delete

Use Case 5: Admin Quản Lý Người Dùng (User Management)

Trường	Nội Dung
Tên Use Case	User Management - Admin Create/Edit/Delete User Account
Mô Tả	Admin vào "Admin Panel" → "User Management", xem list users, add/edit/delete user, assign role (admin/user).
Tác Nhân	Admin User, Frontend AdminPanel Component, Backend API (/api/users)
Mô Tả Chi Tiết	AdminPanel component render tabs. UserManagement tab render user table + form. CREATE: form inputs, validate, POST /api/users. READ: GET /api/users list. UPDATE: PUT /api/users/. DELETE: DELETE /api/users/.
Điều Kiện Tiên Quyết	Current user role='admin'; JWT token valid; User API endpoints available
Luộng Chính	**List Users:** 1. Admin click "User Management" tab 2. Frontend gọi getAllUsers() 3. Backend GET /api/users 4. Response: `{users: [{id, username, email, role, is_active}, ...]}` 5. Render table: - Header: ID
Luộng Thay Thế	A1: LDAP sync - Sync user từ LDAP (enterprise) A2: Batch import - CSV upload users A3: Password reset - Admin force reset user password, send email link A4: 2FA - Require 2-factor authentication (OTP, authenticator app)
Luộng Ngoài Lề	E1: Last admin - Cannot delete last admin user E2: Self delete - Admin cannot delete self E3: Duplicate username - Check duplicate, error E4: Weak password - Warn if password weak
Điều Kiện Sau	User accounts managed; New user account available; User can login

Use Case 6: View Activity Logs (Log Entry/Exit)

Trường	Nội Dung
Tên Use Case	Activity Logs - User/Admin Xem Lịch Sử Vào/Ra
Mô Tả	User vào tab "Logs", xem lịch sử RFID scan (entry/exit), filter theo card/ngày/action, export CSV.
Tác Nhân	Admin/User, Frontend LogViewer Component, Backend API (/api/cards/logs)
Mô Tả Chi Tiết	LogViewer component render table + filter form. Filter: card_id, action, date_from, date_to. GET /api/cards/logs?filters. Render paginated table: timestamp, uid, action, duration. Export button: CSV download.
Điều Kiện Tiên Quyết	JWT token valid; Activity logs available ở backend
Luộng Chính	1. User/Admin click "Logs" tab 2. LogViewer component mount 3. Initial fetch: GET /api/cards/logs?limit=50 (latest 50) Response: `{success: true, logs: [...], total: X, message: "..."}` 4. Render table: - Header: Timestamp
Luộng Thay Thế	A1: Real-time WebSocket - Push logs instead poll A2: Monthly report - Aggregate logs per month A3: Analytics - Chart entry/exit trends over time A4: Alert rules - Alert if unusual pattern detected
Luộng Ngoài Lề	E1: No logs - Filter return empty, display "No logs found" E2: Large dataset - If 10k+ rows, paginate, slow query cache E3: User permission - User see only own logs, admin see all E4: Timezone - Display server time, may differ client timezone
Điều Kiện Sau	User/Admin xem activity logs; Có thể filter, export, audit trail

Use Case 7: System Monitoring & Health Check (Admin)

Trường	Nội Dung
Tên Use Case	System Monitoring - Admin Xem Status & Health Check
Mô Tả	Admin vào "System" tab, xem health check: Backend status, ESP32 status, UNO R4 status, database, WiFi, disk space. Alert if problem.

Trường	Nội Dung
Tác Nhân	Admin User, Frontend System Component, Backend API (/api/system)
Mô Tả Chi Tiết	System component GET /api/system/health, /api/system/status. Render status dashboard: components health (green/red), metrics (uptime, database size, log size), alerts. Auto-refresh 60s.
Điều Kiện Tiên Quyết	Admin JWT token valid; Backend health endpoints available

Luộng Chính	1. Admin click "System" tab
	2. Frontend gọi getSystemHealth() + getSystemStatus()
	3. Backend GET /api/system/health (ref: Backend UC-10)
	4. Response:
	`{`
	`healthy: true,`
	`health: {`
	`system: {status: "healthy", platform: "Linux", python: "3.11"},`
	`services: {`
	`card_service: {healthy: true, message: "OK"},`
	`esp32_service: {healthy: false, message: "Connection timeout"}`
	`},`
	`files: {`
	`cards_file: {accessible: true, size: 5120},`
	`unknown_cards_file: {accessible: true, size: 1024}`
	`}`
	`}`
	`}`
	
	5. Render dashboard:
	
	Component Status Cards (health):
	6. Card: Backend
	- Icon: green circle (healthy)
	- Status: "Healthy"
	- Uptime: "5d 12h 30m"
	- Version: "1.0.0"
	
	7. Card: ESP32
	- Icon: green circle (healthy)
	- Status: "Connected"
	- WiFi: "Connected to UNO-R4-AP"
	- IP: "192.168.4.5"
	- Last heartbeat: "10s ago"

Trường	Nội Dung
	8. Card: UNO R4 - Icon: green circle (healthy) - Status: "Connected" - WiFi AP: "UNO-R4-AP running" - IP: "192.168.4.3" - Last heartbeat: "5s ago" 9. Card: Database - Icon: green circle (healthy) - Status: "OK" - Size: "250 MB" - Records: "5000 cards, 50000 logs" - Last backup: "30 min ago" **Metrics Section:** 10. Table metrics: - Metric
Luộng Thay Thế	A1: Detailed metrics - Add more system metrics (CPU, memory, network) A2: Alert configuration - Admin customize alert thresholds A3: Logs viewer - Live streaming logs từ backend A4: Crash reporting - Send crash dumps to admin
Luộng Ngoài Lề	E1: Backend offline - Health check fail, show red status E2: Slow response - Health check slow, show skeleton loader E3: Permission denied - If user not admin, hide System tab
Điều Kiện Sau	Admin có complete picture của system health; Có thể detect issues early

---

> **UC-8 Config Settings sẽ được implement ở version 2.0** - Các endpoints (configure, backup-config, notification-config, maintenance) chưa implement trong backend. Tính năng này sẽ được phát triển trong phiên bản tiếp theo.

Use Case 9: User Logout

Trường	Nội Dung
Tên Use Case	User Logout - Đăng Xuất Hệ Thống
Mô Tả	User click "Logout" button. Frontend delete token, redirect login page. Backend không cần làm gì (JWT stateless).
Tác Nhân	User, Frontend, Browser/App storage
Mô Tả Chi Tiết	Logout button call handleLogout(). Remove token từ localStorage. Clear user state. Redirect /login.

Trường	Nội Dung
Điều Kiện Tiên Quyết	User logged in; JWT token in storage
Luộng Chính	**Web Frontend:** 1. User click "Logout" button (top right) 2. Confirm modal: "Sure you want to logout?" 3. User click "Logout" 4. handleLogout(): - localStorage.removeItem("authToken") - setState(user, null) - navigate("/login") 5. Token deleted, cannot use API anymore 6. Frontend redirect login page **Desktop Electron App:** 7. User click "Logout" button 8. Confirm modal 9. handleLogout(): - localStorage.removeItem("authToken") → clear token from localStorage - setState(user, null) - navigate("/login") 10. Desktop app logout
Luộng Thay Thế	A1: Auto-logout - Auto logout after 30 min inactivity A2: Logout all devices - Admin logout user from all sessions (requires session tracking) A3: Session management - Display active sessions, logout from other devices
Luộng Ngoài Lề	E1: Token not in storage - Logout on non-existent token (edge case) E2: Network error - Logout works client-side (stateless), no server call needed
Điều Kiện Sau	User logged out; Token deleted; Cannot access protected endpoints; Redirect login page

Bảng Tóm Tắt User Key (Frontend Operations)

User Key	Use Case	Component	Hành Động
Authentication	UC-1, UC-9	LoginPage.tsx	Login, logout, token management
Dashboard	UC-2	Dashboard.tsx	View system overview, KPIs, charts, alerts
Card Management	UC-3	CardList.tsx	Add/edit/delete cards, search, bulk operations
Unknown Cards	UC-4	AdminPanel.tsx ⚠	Review unknown cards, whitelist, delete

User Key	Use Case	Component	Hành Động
User Management	UC-5	AdminPanel.tsx	Create/edit/delete users (admin only)
Activity Logs	UC-6	LogViewer.tsx	View entry/exit logs, filter, export CSV
System Monitoring	UC-7	Dashboard.tsx ⚠	Check health, view metrics, alerts (admin only)
Auto-refresh	UC-2, UC-6, UC-7	Multiple	Real-time updates via polling/WebSocket

Ghi Chú:

- ⚠ **Missing Components**: System Monitoring, Unknown Cards UI - cần implement hoặc merge vào AdminPanel

---

Kết Luận

Frontend (web/desktop) cung cấp user interface cho admin và staff để tương tác với parking system. Admin có full control (dashboard, card management, user management, system config). User (staff) chỉ xem dashboard, logs (read-only). Tất cả request yêu cầu JWT token, backend verify authorization trước execute API. Frontend auto-refresh các component để có dữ liệu real-time.