Back to previous view

[PAR-255] [item Parking] Cubework Seller Parking - RFID Created: 10/28/24 Updated: 10/28/24 Resolved: 10/28/24

Status:	New
Project:	Cubework Parking
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	Story	**Priority:**	Major
Reporter:	Bassel Matta	**Assignee:**	zhenyan.guo
Resolution:	New	**Votes:**	0
Labels:	Parking		
Remaining Estimate:	Not Specified		
Time Spent:	Not Specified		
Original Estimate:	Not Specified		

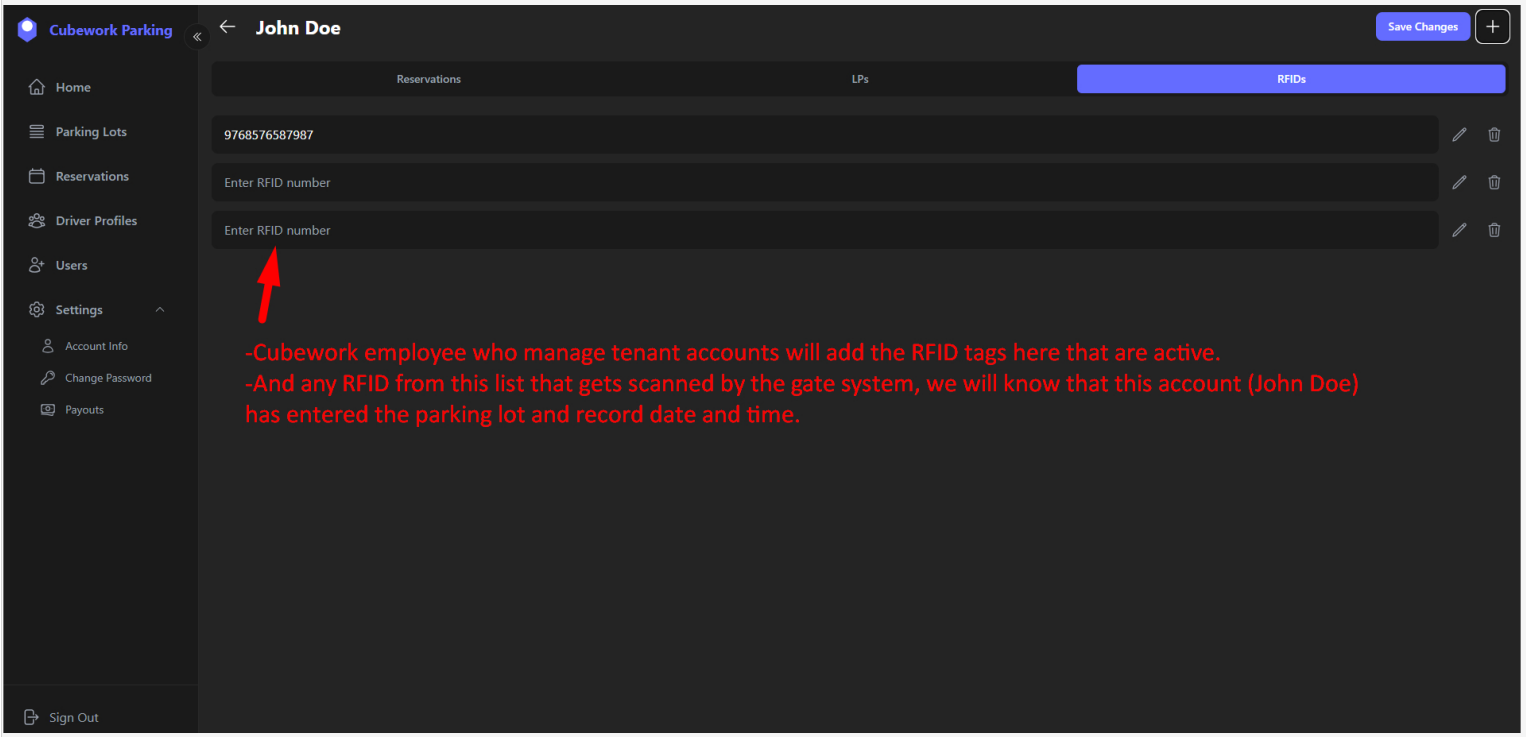
Attachments:	image-2024-10-28-10-17-27-806.png image-2024-10-28-10-28-02-428.png image-2024-10-28-10-30-47-220.png image-2024-10-28-10-53-50-055.png image-2024-10-28-10-53-56-217.png
Dev Hours:	0
Developer:	jie.cheng@unisco.com, mark.guamos@item.com
Story Points:	0
Test Hours:	0
Test Points:	0

Description

In this ticket i will explain the story of the RFID system and how it will be used by the cubework seller.

Steps:

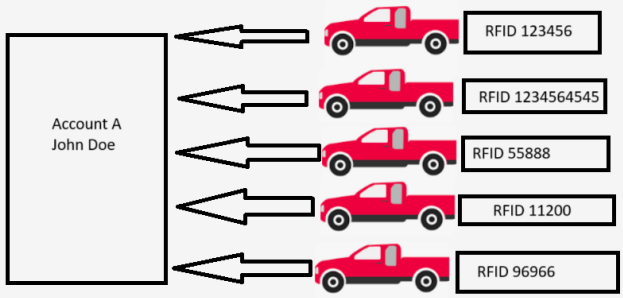
The cubework user will be able to enter the RFID number of the active tags in the seller dashboard under the specific account. (No 2 same RFID can be on different accounts, each account should have unique RFIDs)



For example, lets say we have 10 RFID numbers to give for account A for a monthly use.

so Account A will pay us $800 x 10 RFID tags to be able to use these RFID numbers for the whole month.

Cubework will give these 10 tags to the account A so they can place on any of their trucks.



Once driver place the RFID tag on their trucks and the truck comes to the gate then we will be able to know that this account has a truck in the yard.

Also once the truck comes to the gate, the gate will open if the RFID is active and we will record the entry time in the system for this account and we should be able to see how long this truck has stayed in the lot.

We should record the time and date and take image if camera is working of the car with the RFID tag once it is detected.

-Gate opens

-Record Time/Date

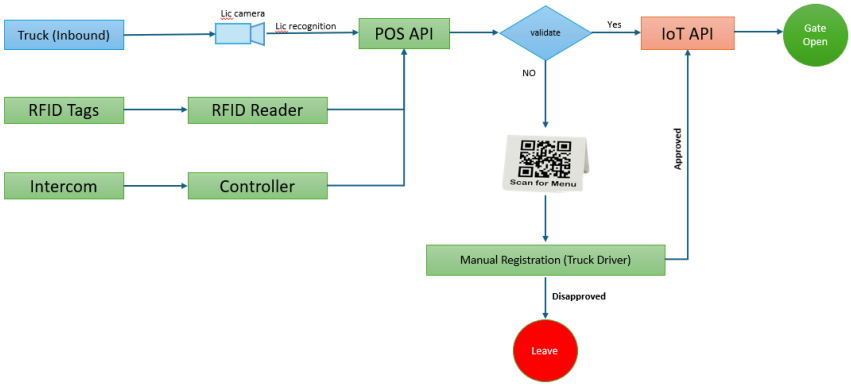
-Take Picture & save to database

side note - Cubework have the system to be able to turn ON/OFF a specific RFID number manually and we do not have to turn it ON/OFF by our system for now.

Workflow:

Inbound Process Flow:

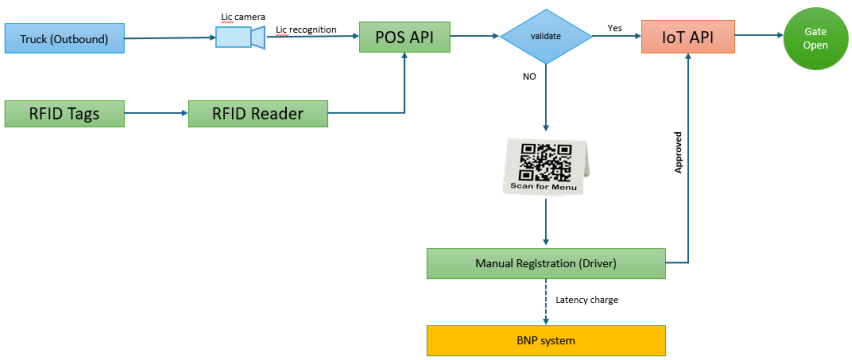
1. **Truck Arrival and License Plate Recognition**:
   - When a truck arrives at the parking lot, a **license plate camera** captures the truck's license plate.
   - The camera sends the license plate information to the **POS API (Cubework seller platform API)** for initial verification.
2. **RFID Detection**:
   - If the truck is equipped with an **RFID tag**, the **RFID reader** scans the tag.
   - The scanned RFID is sent to the **POS API** to check if the RFID is registered and associated with an active account.
3. **Validation Check in POS API**:
   - The POS API validates both the license plate and RFID against the Cubework database.
   - If the validation passes (i.e., the RFID is active and the truck is authorized):
     - The **POS API** sends a confirmation to the **IoT API**.
     - The IoT API communicates with the **gate barrier**, instructing it to open.
   - If the validation fails (i.e., the RFID is inactive or unregistered):
     - A **QR code** is generated on-site for manual registration.
4. **Manual Registration**:
   - If the RFID tag or license plate is not recognized, the driver is prompted to scan a QR code.
   - The QR code directs the driver to a **manual registration interface**, where they can enter their details.
   - Upon manual registration, if the driver is approved, they are granted entry, and the **IoT API** opens the gate.
   - If not approved, they are directed to leave.
5. **Data Recording**:
   - The system logs entry time, date, and the captured license plate or RFID data.
   - If a camera is enabled, it captures an image of the truck, which is saved in the database for record-keeping.



Inbound diagram

Outbound Process Flow:

1. **Truck Departure and License Plate Recognition**:
   - As the truck exits, the license plate camera captures the truck's license plate.
   - The POS API verifies the license plate information against its records.
2. **RFID Detection**:
   - The RFID reader scans the truck's RFID tag and sends this information to the POS API for validation.
   - The POS API checks if the tag is still valid (e.g., has not expired and is linked to an active account).
3. **Validation Check in POS API**:
   - If the RFID and license plate match the POS API's records, the IoT API sends a signal to open the gate.
   - If the validation fails (due to expired RFID, unrecognized license plate, etc.), a QR code is presented for manual registration.
4. **Manual Registration and Latency Charge**:
   - If the validation fails, the driver must scan a QR code and go through **manual registration**.
   - After successful manual registration, the driver may incur a **latency charge** if applicable, which is processed through the **BNP system**.
   - Upon successful registration, the IoT API opens the gate to allow exit.
5. **Data Logging**:
   - The system records the exit time and captures a final image of the truck, if possible.
   - This data is stored for reporting and audit purposes, allowing Cubework to monitor usage duration and calculate any applicable fees.



Outbound diagram

Comments

Comment by Bassel Matta [10/28/24]

AI Notes: (more detailed)

Overview of the RFID System for Cubework Parking Lot Management

The RFID system in this setup is designed to streamline and automate the entry and monitoring of trucks entering the Cubework parking lot. Here's a detailed breakdown for the development team on the workflow, API integration requirements, and functionalities that need to be implemented.

System Workflow and User Story

1. **Account and RFID Association in Dashboard**:
   - Cubework employees (using the seller dashboard) can assign RFID tags to tenant accounts. Each account corresponds to a company or tenant, and each tenant can have multiple RFID tags assigned to it.
   - In the Cubework seller dashboard, employees will manually enter each RFID number under the tenant's account (like "Account A" for "John Doe").
   - **Business Rule**: Each RFID must be unique to a single account. No RFID can be shared across multiple accounts. This prevents access conflicts and ensures that each truck is linked to the correct tenant.
2. **RFID Tag Distribution and Payment**:
   - RFID tags are provided to the tenant on a subscription basis (e.g., Account A rents 10 RFIDs for a month for $800 each).
   - These tags are placed on the tenant's trucks to allow entry into the parking lot.
   - The tenant can then place the RFID on any truck they want to give access, but the system will track and record the entry and exit times for each tag.
3. **Entry Detection and Gate Interaction**:
   - When a truck with a registered RFID approaches the gate, an **IoT RFID Detector** at the gate will detect the RFID tag.
   - **Validation Process**:
     - The RFID detector sends the detected RFID number to the backend system via an API.
     - The backend verifies if the RFID is active and associated with an account.
   - **Gate Action**:
     - If the RFID is valid and active, the backend system sends a signal to the **Gate Barrier** API, instructing it to open the gate for entry.
4. **Recording Entry Data**:
   - Upon validation, the system records the following:
     - **Date and Time**: Logs the exact time when the truck enters the yard.
     - **Image Capture**: If a camera is installed, it captures an image of the truck with the RFID tag, which is stored in the database along with the entry details.
   - This data allows Cubework to monitor how long each truck stays in the parking lot.
5. **Manual Control of RFID Status**:
   - Cubework has the capability to manually activate or deactivate specific RFID tags through the seller dashboard.
   - This control is handled manually and is not tied to an automated process at the moment. For example, if a tag needs to be deactivated due to non-payment, Cubework employees can do this directly in the system.

Key Functionalities and API Integrations

1. **RFID Registration and Management in Seller Dashboard**:
   - **Frontend Requirements**:
     - Form fields for entering RFID numbers associated with a specific account.
     - Validation to ensure no duplicate RFIDs across different accounts.
     - Option to add or remove RFID tags from an account.
   - **Backend Requirements**:
     - Store RFID numbers with a relational mapping to account IDs.
     - API endpoints to add, update, and remove RFID tags linked to specific accounts.
2. **RFID Detection and Verification API**:
   - **API from RFID Detector to Backend**:
     - Endpoint that receives the detected RFID data from the gate's RFID detector.
     - Backend verifies the received RFID by checking the database for an active status and correct account association.
   - **Response**:
     - If the RFID is valid and active, send a response to open the gate.
     - If not, deny access and log the unauthorized access attempt.
3. **Gate Control API**:
   - **API from Backend to Gate Barrier**:
     - Once an RFID is validated, the backend sends an open gate command to the gate's control system.
     - The gate barrier system should respond back to confirm the action.
   - **Potential Fail-Safe**: Include error handling to ensure that if the gate does not open or if there's a communication issue, the system logs the error and alerts the maintenance team.
4. **Entry Logging and Image Capture**:
   - **Backend Requirements**:
     - Record entry date and time for each RFID detection associated with an account.
     - If an image capture device is in place, link the captured image to the entry record.
   - **Database Requirements**:
     - Store images and timestamps in a database, linked to each account and RFID tag for historical tracking and reporting.
5. **Manual RFID Activation/Deactivation**:
   - **Dashboard Controls**:
     - In the Cubework seller dashboard, provide a toggle to activate/deactivate RFIDs associated with accounts.
   - **Backend Process**:
     - Update the RFID status in the database as active or inactive based on user action.
     - Ensure that any deactivated RFID does not allow entry through the gate.

Technical Notes for Developers

- **Database Design**: Ensure the schema is designed to support unique constraints on RFID numbers across accounts, with efficient querying for real-time validation at the gate.
- **Scalability**: The system should handle high concurrency for real-time checks at the gate to avoid entry delays.
- **Error Handling**: Implement logging for failed RFID detections, API communication errors, and image capture issues to streamline troubleshooting.
- **API Security**: Use secure, authenticated APIs between the gate system, RFID detector, and backend to prevent unauthorized access.
- **Testing**: Simulate RFID detection and gate control actions in a staging environment to ensure seamless integration before deployment.

Comment by Bassel Matta [10/28/24]

API Integration and Communication Requirements:

1. POS API (Cubework Seller Platform):

- **Functions**:
  - Accepts incoming data from the license plate camera and RFID reader.
  - Validates license plate numbers and RFID tags against active records in the Cubework database.
  - Forwards validation results to the IoT API for gate control.
- **Endpoints**:
  - `POST /validate_entry`: Accepts RFID and license plate data for inbound validation.
  - `POST /validate_exit`: Accepts RFID and license plate data for outbound validation.
  - `POST /manual_registration`: Receives data from QR code scans for manual entry.
- **Responses**:
  - `200 OK` if validation is successful.
  - `403 Forbidden` if validation fails (triggers manual registration).

2. IoT API:

- **Functions**:
  - Receives gate control commands from the POS API.
  - Sends open/close commands to the gate barrier system.
- **Endpoints**:

- `POST /gate_control`: Accepts requests to open or close the gate based on validation results.
- **Responses**:
  - `200 OK` if the gate opens successfully.
  - `500 Internal Server Error` if there's an issue, which should log and alert for troubleshooting.

3. BNP System:

- **Functionality**: Manages latency charges for trucks that exceed their allowed parking duration or require manual registration for extended periods.
- **Integration**:
  - POS API communicates with the BNP system to calculate any applicable charges and records them under the tenant's account.

Edge Cases and Error Handling

- **Failed RFID or License Plate Detection**:
  - Fallback to manual registration with QR code and log the incident for review.
- **IoT API Communication Failure**:
  - Implement a retry mechanism and an alert system for quick troubleshooting.
- **Manual Registration Failures**:
  - Log all manual entries for verification and dispute resolution in case of validation errors or mismatches.