

Technical Specification for Rental Process Application

Database Structure: laurel_rental_application

The database structure outlined below serves as the foundational framework for the Rental Process Application. This structure is designed to efficiently manage all aspects of the rental process, including inventory management, customer data, order processing, and user roles. While the following tables are critical to the application's functionality, additional tables can be added as per development needs.

1. Table: *warehouse_product_list*

Description:

This table stores information about the equipment available in the warehouse, including their identification, material grouping, and subtypes for accounting or classification purposes.

Columns:

1. **equipment_id** - INT - PRIMARY KEY, AUTO_INCREMENT
 - A unique identifier for each piece of equipment.
2. **equipment_material_group** - VARCHAR(255)
 - A text-based category representing the material group to which the equipment belongs
3. **equipment_name** - VARCHAR(255)
 - The name or description of the equipment (e.g., "TV," "Laser Cutter").
4. **subtype** - VARCHAR(255)
 - The subtype of the equipment, which categorizes it further for purposes such as accounting or inventory management.
5. **last_updated_by** - VARCHAR(255) - NOT NULL
 - The user_id of the user who last updated this record.

Table Structure Example:

```
CREATE TABLE warehouse_product_list (  
    equipment_id INT PRIMARY KEY AUTO_INCREMENT,  
    equipment_material_group VARCHAR(255) NOT NULL,  
    equipment_name VARCHAR(255) NOT NULL,  
    subtype VARCHAR(255) NOT NULL,  
    last_updated_by VARCHAR(255) NOT NULL,  
    FOREIGN KEY (last_updated_by) REFERENCES user_list(user_id)  
);
```

Notes:

The equipment_material_group column now stores descriptive text labels that categorize the equipment.

The subtype column remains for additional classification purposes, useful for accounting or organizational needs.

2. Table: warehouse_list

Description:

This table stores information about the various warehouse locations, including their names and addresses.

Columns:

1. **location_id** - INT - PRIMARY KEY, AUTO_INCREMENT
 - A unique identifier for each warehouse location.
6. **location_name** - VARCHAR(255)
 - The name or designation of the warehouse location (e.g., "Bakersfield," "Brawley").
7. **address** - VARCHAR(255)
 - The full address of the warehouse, including street address, city, state, and ZIP code.
8. **latitude**: Latitude of the warehouse location.
9. **longitude**: Longitude of the warehouse location.
10. **last_updated_by** - VARCHAR(255) - NOT NULL
 - The user_id of the user who last updated this record.

Table Structure Example:

```
CREATE TABLE warehouse_list (  
    location_id INT PRIMARY KEY AUTO_INCREMENT,  
    location_name VARCHAR(255) NOT NULL,  
    address VARCHAR(255) NOT NULL  
    last_updated_by VARCHAR(255) NOT NULL,  
    FOREIGN KEY (last_updated_by) REFERENCES user_list(user_id)  
);
```

Notes:

- The location_id column is a unique, auto-incremented identifier for each warehouse, ensuring that each location can be uniquely referenced.
- The location_name column stores the name or designation of the warehouse, which may reflect its geographic location or other identifying features.
- The address column is a text field that stores the complete address of each warehouse, making it easy to locate or reference specific warehouses.

3. Table: *account_managers*

Description:

This table stores information about account managers, including their names and email addresses.

Columns:

1. **manager_id** - INT - PRIMARY KEY, AUTO_INCREMENT
 - A unique identifier for each account manager.
2. **manager_name** - VARCHAR(255)
 - The full name of the account manager (e.g., "Brawley Smith," "Brooke Marchy").
3. **email_address** - VARCHAR(255) - UNIQUE
 - The email address of the account manager.
4. **last_updated_by** - VARCHAR(255) - NOT NULL
 - The user_id of the user who last updated this record.

Table Structure Example:

```
CREATE TABLE account_managers (  
    manager_id INT PRIMARY KEY AUTO_INCREMENT,  
    manager_name VARCHAR(255) NOT NULL,  
    email_address VARCHAR(255) UNIQUE NOT NULL,  
    last_updated_by VARCHAR(255) NOT NULL,  
    FOREIGN KEY (last_updated_by) REFERENCES user_list(user_id)  
);
```

Notes:

- The manager_id column is a unique, auto-incremented identifier for each account manager.
- The manager_name column stores the full name of the account manager.
- The email_address column is designed to be unique to prevent duplicate email entries and ensure that each manager's email is distinct.

4. Table: *customer_list*

Description:

This table stores information about customers, including their internal ID, customer ID, name.

Columns:

1. **internal_id** - INT - PRIMARY KEY, AUTO_INCREMENT
 - A unique internal identifier for each customer.
2. **customer_id** - VARCHAR(50) - UNIQUE
 - A unique customer identifier used within the system (e.g., "C006383").
3. **last_updated_by** - VARCHAR(255) - NOT NULL

- The `user_id` of the user who last updated this record.

Table Structure Example:

```
CREATE TABLE customer_list (
  internal_id INT PRIMARY KEY AUTO_INCREMENT,
  customer_id VARCHAR(50) UNIQUE NOT NULL,
  name VARCHAR(255) NOT NULL,
  FOREIGN KEY (last_updated_by) REFERENCES user_list(user_id)
);
```

Notes:

- The `internal_id` column is a unique identifier for each customer and serves as the primary key.
- The `customer_id` column is a unique identifier specific to the customer within the system.

5. Table: *stock_adjustment*

Description:

This table records adjustments to equipment stock in the warehouse, ensuring that all entries correspond to equipment listed in the `warehouse_product_list` table. The table includes details such as the location, date, quantity changes, and reasons for the adjustments.

Columns:

1. **adjustment_id** - INT - PRIMARY KEY, AUTO_INCREMENT
 - A unique identifier for each stock adjustment entry.
2. **equipment_id** - INT
 - The unique identifier for the equipment being adjusted, which must match an `equipment_id` in the `warehouse_product_list` table.
3. **equipment_material_group** - VARCHAR(255)
 - The material group of the equipment, corresponding to the `equipment_material_group` in the `warehouse_product_list` table.
4. **equipment_name** - VARCHAR(255)
 - The name of the equipment, corresponding to the `equipment_name` in the `warehouse_product_list` table.
5. **location** - VARCHAR(255)
 - The name of the location where the stock adjustment is taking place (e.g., "Colusa," "Old River").
6. **date** - DATE
 - The date on which the stock adjustment is recorded.
7. **original_quantity** - INT
 - The quantity of the stock before the adjustment.
8. **new_quantity** - INT
 - The quantity of the stock after the adjustment.

9. **plus** - INT - DEFAULT 0
 - The amount by which the stock quantity increased. If no increase, this field will be 0.
10. **minus** - INT - DEFAULT 0
 - The amount by which the stock quantity decreased. If no decrease, this field will be 0.
11. **reason** - VARCHAR(255)
 - The reason for the stock adjustment (e.g., "Damaged," "Restocked").
12. **comment** - VARCHAR(255)
 - Any additional comments related to the stock adjustment.
13. **changed_done_by** – VARCHAR(255) - NOT NULL
 - The user_id of the user who performed this stock adjustment.

Table Structure Example:

```
CREATE TABLE stock_adjustment (
  adjustment_id INT PRIMARY KEY AUTO_INCREMENT,
  equipment_id INT NOT NULL,
  equipment_material_group VARCHAR(255) NOT NULL,
  equipment_name VARCHAR(255) NOT NULL,
  location VARCHAR(255) NOT NULL,
  date NOT NULL,
  original_quantity INT NOT NULL,
  new_quantity INT NOT NULL,
  plus INT DEFAULT 0,
  minus INT DEFAULT 0,
  reason VARCHAR(255) NOT NULL,
  comment VARCHAR(255),
  changed_done_by VARCHAR(255),
  FOREIGN KEY (equipment_id) REFERENCES warehouse_product_list(equipment_id),
  FOREIGN KEY (changed_done_by) REFERENCES user_list(user_id)
);
```

Notes:

- The equipment_id, equipment_material_group, and equipment_name columns ensure that each stock adjustment corresponds directly to an entry in the warehouse_product_list table.
- The FOREIGN KEY constraint on equipment_id enforces referential integrity, ensuring that an adjustment can only be made for equipment that exists in the warehouse_product_list.
- This design prevents inconsistencies, ensuring that all stock adjustments are valid and correspond to actual equipment in the warehouse.

6. Table: user_list

Description:

This table stores information about the users who interact with the Laurel Rental Application, including their contact details, permissions, and account status. The table now includes fields to track whether a user is active and, if not, the date they were deactivated.

Columns:

1. **user_id** - INT - PRIMARY KEY, AUTO_INCREMENT
 - A unique identifier for each user.
2. **user_name** - VARCHAR(255)
 - The full name of the user.
3. **email** - VARCHAR(255) - UNIQUE
 - The email address of the user.
4. **contact_number** - VARCHAR(20)
 - The contact number of the user.
5. **permissions** - TEXT
 - The permissions assigned to the user. This will store a list of permissions as a comma-separated string, such as 'Check In, Maintain Master Data'.
6. **active** - BOOLEAN - DEFAULT TRUE
 - Indicates whether the user is currently active (TRUE) or inactive (FALSE).
7. **date_of_inactivation** - DATE - NULLABLE
 - The date on which the user was deactivated. This field remains NULL if the user is active.
8. **last_updated_by** - VARCHAR(255) - NOT NULL
 - The user_id of the user who last updated this record.

Table Structure Example:

```
CREATE TABLE user_list (  
  user_id INT PRIMARY KEY AUTO_INCREMENT,  
  user_name VARCHAR(255) NOT NULL,  
  email VARCHAR(255) UNIQUE NOT NULL,  
  contact_number VARCHAR(20),  
  permissions TEXT,  
  active BOOLEAN DEFAULT TRUE,  
  date_of_inactivation DATE,  
  last_updated_by VARCHAR(255) NOT NULL,  
  FOREIGN KEY (last_updated_by) REFERENCES user_list(user_id)  
);
```

Notes:

- The active column is a BOOLEAN type, used to quickly determine if a user is currently active. It defaults to TRUE when a new user is added.
- The date_of_inactivation column stores the date a user was deactivated. This field should only be populated when active is set to FALSE.
- If a user is active (TRUE), the date_of_inactivation should remain NULL.

7. Table: *order_line_items*

Description:

The *order_line_items* table stores detailed information about each individual item within an order or recurring order in the Laurel Rental Application. This table captures all relevant details for each product or service associated with an order, including customer information, pricing, rental periods, and recurrence details. Each entry in this table represents a specific line item within an order.

Columns:

1. **order_line_item_id** - INT - PRIMARY KEY, AUTO_INCREMENT
 - A unique identifier for each line item within an order.
2. **order_id** - VARCHAR(20) - NOT NULL
 - The identifier for the order to which this line item belongs.
 - Example: O20240200008 (O for order, 2024 for year of contract start date, 02 month of contract start date, 00008 sequence number) or OR20240200008 (O for order recurring, 2024 for year of contract start date, 02 month of contract start date, 00008 sequence number)
 - This field references the primary key of the orders table.
3. **customer_id** - VARCHAR(20) - NOT NULL
 - The unique identifier for the customer associated with the order.
 - This field references the primary key of the customer_list table.
4. **customer_name** - VARCHAR(255) - NOT NULL
 - The full name of the customer associated with the order.
 - This field references the customer's name of the customer_list table.
5. **account_manager** - VARCHAR(255) - NOT NULL
 - The name of the account manager responsible for the customer and the order.
 - This field references the accounts manager's name of the account_managers table.
6. **customer_email** - VARCHAR(255)
 - The email address of the customer associated with the order.
 - Example: customer@example.com
7. **region** - VARCHAR(100)
 - The region where the order is applicable.
 - Example: West
8. **delivery_location** - VARCHAR(255)
 - The address or location where the item(s) are to be delivered.
 - Example: 123 Ranch St, Bakersfield, CA
9. **pick_up_location** - VARCHAR(255)
 - The address or location where the item(s) are to be picked up after the rental period.
 - Example: 456 Pickup Rd, Bakersfield, CA
10. **water_source** - VARCHAR(255)
 - Details of the water source related to the order, if applicable.
 - Example: Well 12
11. **crop** - VARCHAR(100)

- The type of crop associated with the order, useful for agricultural rentals or sales.
 - Example: Wheat
12. **item_number** - VARCHAR(50) - NOT NULL
 - A unique identifier for the item being rented or sold.
 - Example: 113
 - This field references the primary key of the warehouse_product_list table.
 13. **item_name** - VARCHAR(255) - NOT NULL
 - The name or description of the item being ordered.
 - Example: Air Vent Installation
 - This field references the equipment name of the warehouse_product_list table.
 14. **quantity** - INT - NOT NULL
 - The quantity of the item ordered.
 - Example: 10
 15. **unit_price** - DECIMAL(10,2) - NOT NULL
 - The price per unit of the item.
 - Example: 1,111.00
 16. **line_item_amount** - DECIMAL(10,2)
 - The total amount for this line item, calculated as quantity * unit_price.
 - Example: 11,110.00
 17. **rent_amount** - DECIMAL(10,2)
 - The total rent amount for order, as SUM of line_item_amount
 - Example: 11,110.00
 18. **rent_per_month** - TEXT
 - A JSON string containing details about the rent per month, broken down by each month for this line item.
 - Example: {"August 2024": 55.00, "September 2024": 55.00}
 19. **rental_period_start** - DATE
 - The start date of the rental period for this line item.
 - Example: 2024-08-01
 20. **rental_period_end** - DATE
 - The end date of the rental period for this line item.
 - Example: 2024-09-26
 21. **recurrence_type** - VARCHAR(50)
 - The type of recurrence for this line item, such as "Recurring," "One Time," etc.
 - Example: Recurring
 22. **recurrence_details** - TEXT
 - A JSON string containing detailed recurrence settings, including frequency, duration, and specific dates.
 - Example: {"frequency": "Monthly", "duration": "10 months", "start_date": "2024-08-28", "end_date": "2024-10-24"}
 23. **created_by** - VARCHAR(255) - NOT NULL
 - The user_id of the user who created this record.
 24. **last_updated_by** - VARCHAR(255) - NOT NULL
 - The user_id of the user who last updated this record.

25. **last_updated_date** - DATE

- The date when the line item details were last updated.
- Example: 2024-10-25

Table Structure Example:

```
CREATE TABLE order_line_items (  
  order_line_item_id INT PRIMARY KEY AUTO_INCREMENT,  
  order_id VARCHAR(20) NOT NULL,  
  customer_id VARCHAR(20) NOT NULL,  
  customer_name VARCHAR(255) NOT NULL,  
  account_manager VARCHAR(255) NOT NULL,  
  customer_email VARCHAR(255),  
  region VARCHAR(100),  
  delivery_location VARCHAR(255),  
  pick_up_location VARCHAR(255),  
  water_source VARCHAR(255),  
  crop VARCHAR(100),  
  item_number VARCHAR(50) NOT NULL,  
  item_name VARCHAR(255) NOT NULL,  
  quantity INT NOT NULL,  
  unit_price DECIMAL(10,2) NOT NULL,  
  line_item_amount DECIMAL(10,2),  
  rent_amount DECIMAL(10,2),  
  rent_per_month TEXT,  
  rental_period_start DATE,  
  rental_period_end DATE,  
  recurrence_type VARCHAR(50),  
  recurrence_details TEXT,  
  created_by VARCHAR(255) NOT NULL,  
  last_updated_by VARCHAR(255) NOT NULL,  
  last_updated_date DATE,  
  FOREIGN KEY (order_id) REFERENCES orders(order_id),  
  FOREIGN KEY (customer_id) REFERENCES customer_list(customer_id),  
  FOREIGN KEY (last_updated_by) REFERENCES user_list(user_id)  
);
```

Notes:

- The order_line_item_id is an auto-incremented primary key that uniquely identifies each line item within an order.
- The order_id links each line item to its corresponding order/recurring order in the orders table.
- The last_updated_by field ensures traceability of changes, indicating which user last modified the record.
- The line_item_amount is calculated based on the quantity and unit_price of the item.
- Recurrence details and rent information are stored in JSON format to allow flexibility in handling complex data structures.

8. Table: recurring_orders

Description:

The recurring_orders table stores information about orders that recur on a regular basis within the Laurel Rental Application. This table tracks the relationship between the original order and its recurring instances, including customer details, project management, order amounts, and the recurrence schedule.

Columns:

1. **original_order_id** - VARCHAR(20) - NOT NULL
 - The unique identifier for the original order that serves as the basis for recurring orders.
 - Example: O20240300055
2. **recurring_order_id** - VARCHAR(20) - PRIMARY KEY
 - The unique identifier for each recurring instance of the original order.
 - Example: OR20260300055
3. **customer_id** - VARCHAR(20) - NOT NULL
 - The unique identifier for the customer associated with the order.
 - This field references the primary key of the customer_list table.
4. **customer_name** - VARCHAR(255) - NOT NULL
 - The full name of the customer associated with the order.
 - This field references the customer's name of the customer_list table.
5. **order_amount** - DECIMAL(10,2) - NOT NULL
 - The total amount for the recurring order.
 - Example: \$3300.00
6. **project_manager** - VARCHAR(255) - NOT NULL
 - The name of the project manager responsible for managing the recurring order.
 - Example: LINDA GARCIA
7. **recurrence_type** - VARCHAR(50) - NOT NULL
 - The type of recurrence for the order, such as "Weekly," "Monthly," or "Yearly."
 - Example: Weekly
8. **recurring_start** - DATE - NOT NULL
 - The start date of the recurrence period.
 - Example: 2024-01-05
9. **recurring_end** - DATE - NOT NULL
 - The end date of the recurrence period.
 - Example: 2024-12-31
10. **number_of_orders** - INT - NOT NULL
 - The total number of recurring orders that have been generated or are scheduled.
 - Example: 3
11. **last_updated_by** - VARCHAR(255) - NOT NULL
 - The user_id of the user who last updated this record.
 - Example: admin

12. **last_updated_date** - DATE

- The date when the recurring order details were last updated.
- Example: 2024-10-25

Table Structure Example:

```
CREATE TABLE recurring_orders (  
  original_order_id VARCHAR(20) NOT NULL,  
  recurring_order_id VARCHAR(20) PRIMARY KEY,  
  customer_name VARCHAR(255) NOT NULL,  
  order_amount DECIMAL(10,2) NOT NULL,  
  project_manager VARCHAR(255) NOT NULL,  
  recurrence_type VARCHAR(50) NOT NULL,  
  recurring_start DATE NOT NULL,  
  recurring_end DATE NOT NULL,  
  number_of_orders INT NOT NULL,  
  last_updated_by VARCHAR(255) NOT NULL,  
  last_updated_date DATE,  
  FOREIGN KEY (original_order_id) REFERENCES orders(order_id),  
  FOREIGN KEY (last_updated_by) REFERENCES user_list(user_id)  
);
```

Notes:

- The original_order_id references the original order from which recurring instances are generated.
- The recurring_order_id is the primary key and uniquely identifies each instance of a recurring order.
- The recurrence_type specifies how often the order recurs (e.g., Weekly, Monthly, Yearly).
- The number_of_orders field tracks how many recurring orders have been or will be generated within the specified period.
- The last_updated_by and last_updated_date fields track who made the last change and when.

9. *Table: delivery_details*

Description:

This table stores detailed information regarding deliveries and return deliveries within the rental process. Each entry is associated with an order_id, and a single order can have multiple deliveries. The table distinguishes between normal deliveries and return deliveries. For return deliveries, if the deliver_to_customer column is set to TRUE, the reference_order column is used to link back to the original delivery order. Specific columns in this table reference data from other related tables to maintain consistency and data integrity. Additionally, this table includes information about the users responsible for performing each action within the delivery process.

Columns:

1. **delivery_id** - VARCHAR(255) - PRIMARY KEY

- A unique identifier for each delivery or return delivery operation.
 - Example: 'O20240200018-1' for delivery or 'O20240200018-R1' for return delivery
2. **delivery_type** - ENUM('Delivery', 'Return Delivery')
 - Specifies whether the record is for a delivery to a customer or a return delivery from a customer based on the stage.
 3. **order_id** - VARCHAR(255) - FOREIGN KEY
 - The identifier for the order related to this delivery, referencing the order_table.
 - Example: 'O20240200008'
 4. **total_quantity** - INT
 - The total number of items in this delivery or return delivery.
 - Example: 28
 5. **contract_start_date** - DATE
 - The start date of the contract related to this delivery; rental period start date.
 - Example: '2024-04-02'
 6. **contract_end_date** - DATE
 - The end date of the contract related to this delivery or return delivery; rental period end date.
 - Example: '2024-04-02'
 7. **delivery_date** - DATE
 - The date on which the delivery was made or is scheduled.
 - Example: '2024-04-03'
 8. **pickup_date** - DATE
 - The date on which the items were picked up for delivery.
 - Example: '2024-08-28'
 9. **returned_pickup_date** - DATE
 - The date on which the items were picked up for delivery.
 - Example: '2024-08-28'
 10. **check_out_date** - DATE
 - The date on which the items were checked out from the warehouse.
 - Example: '2024-08-28'
 11. **check_in_date** - DATE
 - The date on which the items were checked in after delivery.
 - Example: '2024-08-28'
 12. **inspection_date** - DATE
 - The date on which the items were inspected after check-in.
 - Example: '2024-08-28'
 13. **removal_date** - DATE
 - The date on which items were removed (if applicable).
 - Example: '2024-04-02'
 14. **delivery_status** - VARCHAR(50)
 - The current status of the delivery or return delivery. Status values are based on the stages within the rental process, such as 'Created', 'Loaded', 'Check Out', 'Delivered', 'Check In', 'Inspection Completed', etc.
 - Example: 'Delivered'

15. **material_id** - INT - FOREIGN KEY
 - The identifier for the material being delivered or returned, referencing the warehouse_product_list table.
 - Example: 20315
16. **material_name** - VARCHAR(255) - FOREIGN KEY
 - The name of the material being delivered or returned, referencing the warehouse_product_list table.
 - Example: 'Rental: 10" Plain Coupler'
17. **requested_qty** - INT
 - The quantity of the material that was requested.
 - Example: 2
18. **return_qty** - INT
 - The quantity of the material that was returned (for return deliveries).
 - Example: 10
19. **quantity** - INT
 - The actual quantity of the material delivered or returned.
 - Example: 14
20. **status** - VARCHAR(50)
 - The status of the material within the delivery process.
 - Example: 'Loaded'
21. **delivery_site** - VARCHAR(255)
 - The site where the delivery was made.
 - Example: '425 Webster Street'
22. **pickup_site** - VARCHAR(255)
 - The site from where the delivery was made.
 - Example: '425 Webster Street'
23. **shipping_carrier** - VARCHAR(50)
 - The carrier used for shipping the delivery.
 - Example: 'Hired'
24. **PO_number** - VARCHAR(50)
 - The Purchase Order number associated with this delivery.
 - Example: 'PO123456'
25. **deliver_to_customer** - BOOLEAN - DEFAULT FALSE
 - Indicates whether the return delivery is being sent directly to the customer (TRUE) or to the warehouse (FALSE).
 - Example: TRUE
26. **reference_order** - VARCHAR(255) - FOREIGN KEY
 - If deliver_to_customer is TRUE, this column holds the order ID that the return delivery is referencing. This allows tracking of returns that go directly back to the customer instead of the warehouse, referencing the order_table.
 - Example: 'O20230200015'
27. **reference_delivery_site** - VARCHAR(255)
 - If deliver_to_customer is TRUE, this column holds the pickup_site from original delivery that the return delivery is referencing. This allows tracking of returns that go directly back

to the customer instead of the warehouse, referencing the order_table – always hold warehouse information.

- Example: '425 Webster Street'
- 28. **created_by** - VARCHAR(255) - FOREIGN KEY
 - The identifier of the user who created the delivery entry, referencing the user_list table.
 - Example: 'admin_user'
- 29. **last_updated_by** - VARCHAR(255) - FOREIGN KEY
 - The identifier of the user who last updated this record, referencing the user_list table.
 - Example: 'admin_user'
- 30. **picked_up_by** - VARCHAR(255) - FOREIGN KEY
 - The identifier of the user who performed the pickup of the delivery items, referencing the user_list table.
 - Example: 'warehouse_staff'
- 31. **returned_picked_up_by** - VARCHAR(255) - FOREIGN KEY
 - The identifier of the user who performed the pickup of the delivery items, referencing the user_list table.
 - Example: 'warehouse_staff'
- 32. **checked_out_by** - VARCHAR(255) - FOREIGN KEY
 - The identifier of the user who checked out the items from the warehouse, referencing the user_list table.
 - Example: 'warehouse_staff'
- 33. **checked_in_by** - VARCHAR(255) - FOREIGN KEY
 - The identifier of the user who checked in the items after delivery, referencing the user_list table.
 - Example: 'warehouse_staff'
- 34. **inspected_by** - VARCHAR(255) - FOREIGN KEY
 - The identifier of the user who inspected the items after check-in, referencing the user_list table.
 - Example: 'inspection_staff'
- 35. **comment** - TEXT
 - A field to hold additional information or notes at each stage of the delivery or return delivery process. This can be updated as needed during the process.
 - Example: 'Items were checked out and loaded on the truck without issues.'

Table Structure Example:

```
CREATE TABLE delivery_details (  
  delivery_id VARCHAR(255) PRIMARY KEY,  
  order_id VARCHAR(255) NOT NULL,  
  total_quantity INT NOT NULL,  
  contract_start_date DATE NOT NULL,  
  contract_end_date DATE,  
  delivery_date DATE NOT NULL,  
  pickup_date DATE,  
  returned_pickup_date DATE,  
  check_out_date DATE,
```

```

check_in_date DATE,
inspection_date DATE,
removal_date DATE,
delivery_status VARCHAR(50) NOT NULL,
material_id INT NOT NULL,
material_name VARCHAR(255) NOT NULL,
requested_qty INT,
return_qty INT,
quantity INT NOT NULL,
status VARCHAR(50),
delivery_site VARCHAR(255),
shipping_carrier VARCHAR(50),
PO_number VARCHAR(50),
deliver_to_customer BOOLEAN DEFAULT FALSE,
reference_order VARCHAR(255),
created_by VARCHAR(255) NOT NULL,
last_updated_by VARCHAR(255) NOT NULL,
picked_up_by VARCHAR(255),
returned_picked_up_by VARCHAR(255),
checked_out_by VARCHAR(255),
checked_in_by VARCHAR(255),
inspected_by VARCHAR(255),
comment TEXT,
FOREIGN KEY (order_id) REFERENCES order_table(order_id),
FOREIGN KEY (material_id) REFERENCES warehouse_product_list(equipment_id),
FOREIGN KEY (material_name) REFERENCES warehouse_product_list(equipment_name),
FOREIGN KEY (delivery_site) REFERENCES warehouse_list(address),
FOREIGN KEY (reference_order) REFERENCES order_table(order_id),
FOREIGN KEY (created_by) REFERENCES user_list(user_id),
FOREIGN KEY (last_updated_by) REFERENCES user_list(user_id),
FOREIGN KEY (picked_up_by) REFERENCES user_list(user_id),
FOREIGN KEY (checked_out_by) REFERENCES user_list(user_id),
FOREIGN KEY (checked_in_by) REFERENCES user_list(user_id),
FOREIGN KEY (inspected_by) REFERENCES user_list(user_id)

```

References:

- order_id references order_table(order_id).
- material_id and material_name reference warehouse_product_list(equipment_id) and warehouse_product_list(equipment_name), respectively.
- reference_order references order_table(order_id).
- XXXXX_by reference `user_list`

Landing Page (log-in)

This part outlines the general specifications applicable across all pages of the Laurel Rental Application. These specifications ensure consistency and standardization throughout the application.

1. Navigation and Layout

1.1 Header Section

- **Logo:**
 - **Position:** Top-left corner.
 - **Functionality:** Clicking on the logo navigates the user back to the Home or Dashboard page.
 - **Styling:** The logo is prominently displayed with the Laurel AG & Water branding.
- **User Profile Information:**
 - **Position:** Top-right corner.
 - **Details Displayed:**
 - **Email Address:** Displays the email address of the logged-in user.
 - **User Profile Picture:** A thumbnail image of the user.
 - **Functionality:** Clicking on the user profile opens a dropdown menu with options such as "Switch to Proposal Application," and "Logout."

1.2 Minimize & Global Search

- **Position:** Left top side of the screen.
- **Functionality:** Enable full screen mode/or revert full screen mode. Global search allows searching across the application.

1.3 Sidebar Menu

- **Position:** Left side of the screen.
- **Menu Sections:**
 - **Master Data:**
 - Subsections include Products, Warehouse, Customers, and Account Managers.
 - **Rent Process:**
 - Related to the management of rental transactions.
 - **Stock Adjustment:**
 - Access to the Stock Adjustment interface.
 - **Reports:**
 - For generating and viewing various reports.
 - **Authorizations:**
 - Access to user roles and permissions management.
 - **Workflow:**
 - Management of workflow processes and notifications.
 - **Inbox:**
 - Access to messages and notifications within the application.

- **Styling and Functionality:**
 - **Expandable/Collapsible Sections:** Each section in the sidebar can be expanded or collapsed by clicking on it.
 - **Active Highlighting:** The active section is highlighted to indicate the current page or section the user is viewing.
 - **Icons:** Each section and subsection are accompanied by a relevant icon to enhance usability.

1.3 Footer Section

- **Position:** Bottom of the page.
- **Content:**
 - **Copyright Notice:**
 - "© 2024 Laurel AG & Water."
 - **Design Credit:**
 - "Designed by Markman Group, All rights reserved."
 - **Functionality:** The footer is static and remains consistent across all pages, providing company and design credits.

1.4 Breadcrumb Navigation

- **Position:** Below the header, above the main content.
- **Functionality:** Displays the current page and its hierarchy within the application, allowing users to navigate back to previous sections easily.
- **Styling:** Simple text with arrows separating each level in the navigation hierarchy.

2. General UI/UX Standards

2.1 Consistency

- **Color Scheme:**
 - The application follows a consistent color scheme aligned with the Laurel AG & Water branding.
- **Typography:**
 - Standard fonts and sizes are used throughout the application to maintain readability and a cohesive look.
- **Button Styles:**
 - Buttons across the application have consistent styles, with primary actions highlighted in a prominent color.
- **Input Fields:**
 - All input fields follow a standard design with appropriate labels, placeholders, and validation messages.

2.2 Responsiveness

- **Mobile and Tablet Compatibility:**

- The application layout is responsive, adjusting to different screen sizes, including mobile phones and tablets.
- **Collapsible Menus:**
 - On smaller screens, the sidebar menu collapses into a hamburger menu for easy navigation.

2.4 Performance

- **Loading Speed:**
 - Pages are optimized to load quickly, with lazy loading implemented for images and heavy data elements.
- **Caching:**
 - Frequently accessed data and pages are cached to improve performance and reduce server load.

3. Security and Authentication

3.1 User Authentication with Microsoft Login

- **Single Sign-On (SSO):**
 - Integrated with Microsoft Login, allowing users to sign in using their Microsoft credentials.

3.1 User Authentication

- **Login and Logout:**
 - Secure login and logout mechanisms are in place, using industry-standard encryption protocols.
- **Session Management:**
 - User sessions are managed securely with automatic timeouts for inactivity.

3.2 Data Protection

- **Access Control:**
 - Role-based access control ensures that users can only access information and functions relevant to their role.

4. Integration Points

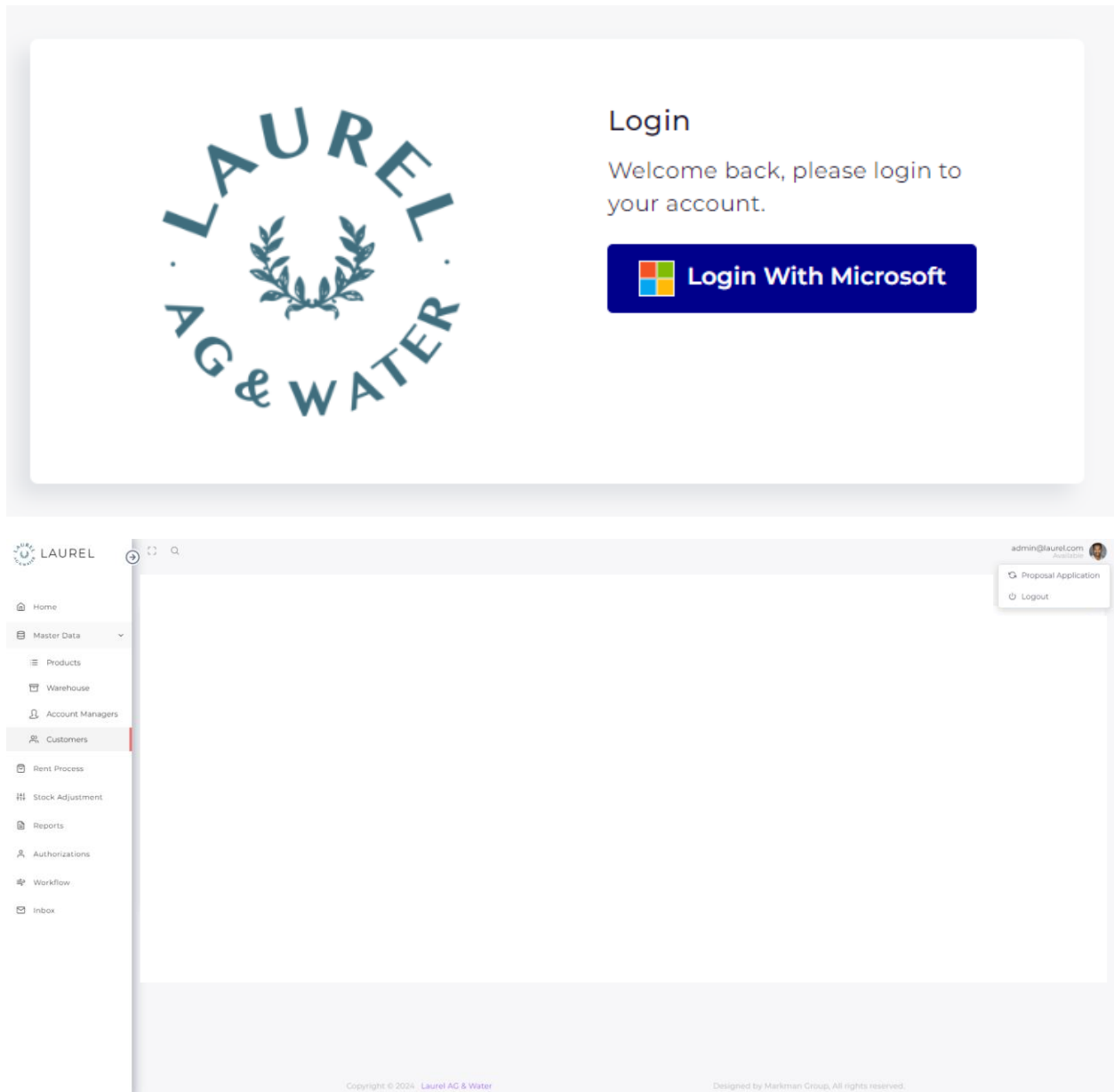
4.1 API Integration

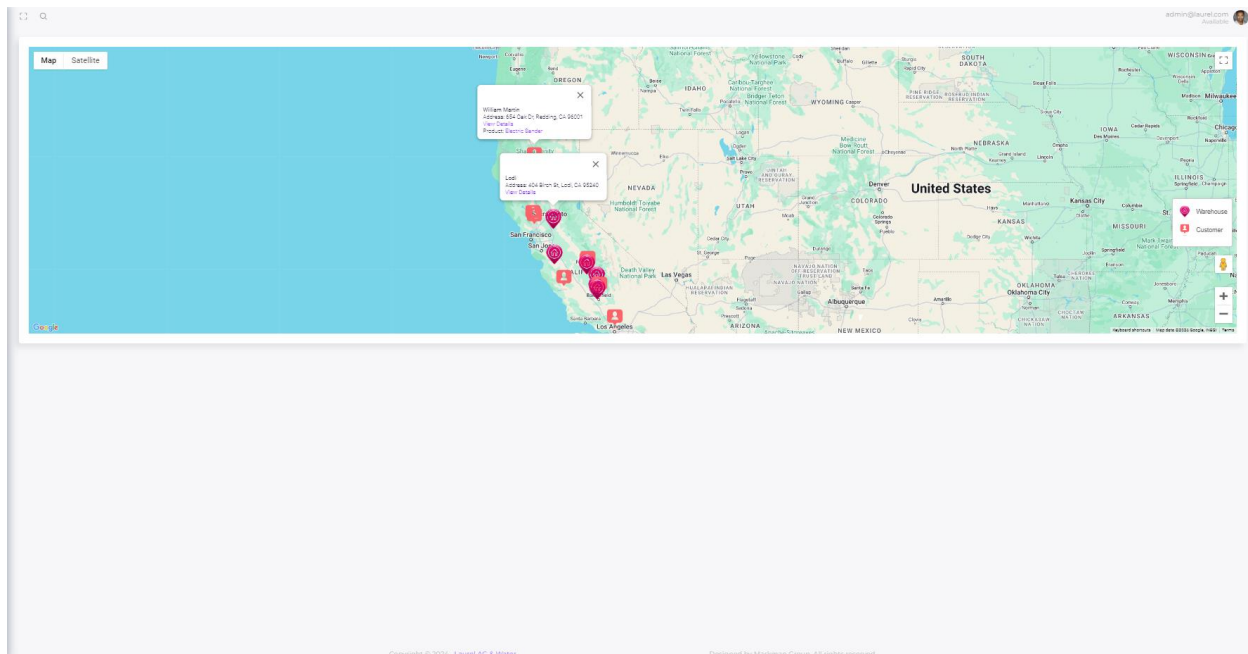
- **Internal APIs:**
 - The application interacts with several internal APIs to fetch and display data across different sections.
- **External Services:**
 - Integration with external services, such as Google Maps for geolocation, is managed securely with API keys stored and managed in a secure environment.

4.2 Data Import/Export

- **File Formats:**
 - The application supports importing and exporting data in Excel (.xlsx) and CSV (.csv) formats across relevant sections like Customers, Products, and Warehouses.
- **Validation:**

- Comprehensive validation processes are in place for all data imports, ensuring data integrity and consistency across the application.





1. Purpose

The Home page is the main interface that displays a map showing the locations of warehouses and customers. The map allows users to interact with location pins, view detailed information, and navigate to specific product listing pages.

2. User Interface Components

2.1. Map Interface

Component: Interactive Map

Design Specifications:

Size & Placement:

- The map should cover 80% of the viewport width and 70% of the viewport height in the default view.
- It should be centered horizontally and vertically within the page.

Initial Zoom Level:

- The map should initialize with a zoom level that fits all visible pins within the viewport.
- Default zoom level should be set to 10 but should dynamically adjust based on the distribution of pins.

Functional Specifications:

Zoom Controls:

Mouse/Trackpad:

- Users can zoom in/out by scrolling the mouse wheel or using the pinch gesture on touch-enabled devices.

UI Controls:

- Zoom in (+) and zoom out (-) buttons should be displayed at the bottom-right corner of the map.

Keyboard Shortcuts:

- + key for zoom in.
- - key for zoom out.

Full-Screen Mode:

Button Placement:

- A full-screen toggle button should be placed in the top-right corner of the map.

Functionality:

- When clicked, the map should expand to cover the entire screen, hiding all other UI elements (header, footer, sidebars, etc.).
- A minimize button should appear in the top-left corner when in full-screen mode to exit back to the default view.
- Ensure the transition between normal view and full-screen is smooth.

Street View Integration:

Icon (Little Figure):

- Place a draggable Street View icon in the bottom-right corner of the map.

Functionality:

- Users can drag the icon to any street on the map, triggering the Street View mode.
- Once activated, a full screen Street View should be displayed.
- Provide a button to return to the default map view.
- Implement a smooth transition between map view and Street View mode.

2.2. Location Pins

Pin Behavior:

Tooltip on Hover:

- On mouse hover, a tooltip will appear above the pin displaying:
 - **Warehouse:** "Warehouse Name" (bold) and "Full Address".
 - **Customer:** "Customer Name" (bold) and "Full Address".
- Tooltip Background: White (#ffffff) with a slight shadow for visibility.

Info Window on Click:

Display: When a pin is clicked, an info window should pop up next to the pin showing:

- **Warehouse:** "Warehouse Name" (bold), "Full Address", and a button labeled "View Products".
- **Customer:** "Customer Name" (bold), "Full Address", and a button labeled "View Products".

Interaction:

- The "View Products" button should be a primary button with a blue white and blue (#007bff) text underline to clearly showcase it is hyperlink.
- Clicking the button should open a new browser tab or window displaying the product listing page for the respective warehouse or customer.
-

John Deo

Show 10 entries Search:

ITEM NO	ITEMS	QUANTITY	ORDER DATE	ORDER ADDRESS
121	SN BL10.6CPH STK30" LD LRSS	6	June 25,2024	456 Oak Avenue, Smallville, USA
122	SN PC HEAD 18.5CPH BLU W/DFLTR	5	June 25,2024	123 Main Street, Anytown, USA

Showing 1 to 2 of 2 entries Previous 1 Next

2.3. Map Interactions

Panning:

- Users should be able to click and drag the map to pan across different areas.
- The map should have smooth panning with a minimum latency of 50ms.

Dynamic Pin Placement:

- Pins should be placed dynamically based on real-time data fetched from the backend.
- Ensure no overlap between pins when they are too close; implement clustering where necessary.
- When clicked on the clustering, zoom-in on cluster break-down

Map Layers:

- Provide a control in the top-left corner to toggle between different map layers:
 - Standard Map (default).
 - Satellite View.
 - Terrain View.

3. Backend Integration

3.2. Database Structure

2.2 Geocoding Service

- **Purpose:** Converts the addresses stored in the warehouse_list and customer_list tables into geographical coordinates (latitude and longitude) to accurately place markers on the map.
- **API Integration:** Utilizes a third-party geocoding API (e.g., Google Geocoding API) to perform address-to-coordinate conversion.

3. Database Interactions

3.1 Warehouse List Table

- **Table Name:** warehouse_list
- **Columns:**
 - location_name: Name of the warehouse.

- address: Full address of the warehouse.
- latitude: Latitude of the warehouse location.
- longitude: Longitude of the warehouse location.
- **Functionality:** Stores the necessary data to place warehouse markers on the map.

3.2 Customer List Table

- **Table Name:** customer_list, order_items
- **Functionality:** Stores the necessary data to place customer markers on the map.

4. Business Logic

4.1 Real-Time Data Display

- **Logic:** The map is dynamically updated to display the most current data from the warehouse_list and customer_list tables. Any changes made to these tables (such as adding a new warehouse or updating a customer's address) will be reflected on the map after the data is fetched and geocoded.

4.2 Geocoding Process

- **Logic:** When new addresses are added or existing addresses are updated in the warehouse_list or customer_list tables, the application automatically sends these addresses to the geocoding service. The returned coordinates are then stored in the corresponding tables to be used for map marker placement.

5. Frontend Development

4.1. UI/UX Considerations

Responsiveness:

- Ensure the map and UI components are fully responsive across devices (desktop, tablet, mobile).
- Implement media queries for different screen sizes:
 - **Mobile:** Map should cover 100% of the viewport width and height. Full-screen mode should hide the browser's address bar if possible.
 - **Tablet/Desktop:** Map size should be as per the default view mentioned above.

4.2. Error Handling

Map Load Errors:

- If the map fails to load, display an error message in place of the map:
 - "Unable to load the map. Please check your internet connection and try again."
 - Provide a "Retry" button to attempt reloading the map.
 - Log errors for further analysis.

API Call Failures:

- Implement retry logic for API calls with exponential backoff.
- Display an appropriate message if the location or product data fails to load

6. Testing Requirements

5.1 Unit Testing

- Test zoom functionality across different devices.
- Verify full-screen mode activation and deactivation.
- Ensure all pins are correctly placed and clickable.

5.2 Integration Testing

- Validate data accuracy from the API.
- Test interaction between map components and data rendering.
- Ensure proper navigation between the map and product list screens.

5.3 User Acceptance Testing (UAT)

- Collect feedback on usability from a sample user base.
- Ensure the flow between map interactions and product listings meets user expectations.

7. Deployment Considerations

6.1 Server Requirements

- Ensure the server can handle multiple requests for map data and product listings efficiently.
- Optimize API responses for speed.

6.2 Performance Optimization

- Implement lazy loading for the map to improve initial load time.
- Use caching for frequently accessed location data.

8. Security Considerations

7.1 Data Security

- Ensure all API endpoints are secured with authentication.
- Protect sensitive customer and warehouse data.

7.2 Map Interaction Security

- Implement rate limiting to prevent abuse of the map functionalities.
- Sanitize user inputs to prevent XSS attacks in map interactions.

Master Data

1. Products

The **Warehouse Product List** screen is an essential part of the Laurel Rental Application, designed to provide a comprehensive view of all equipment stored in various warehouse locations. This screen interfaces with the `warehouse_product_list` database table, displaying key details about each piece of equipment, such as its ID, material group, name, and subtype. Users can filter, search, and import new products into the warehouse inventory through this interface.

1. User Interface (UI) Components

1.1.1. Filters and Search Panel

- *Equipment Material Group Filter:*
 - **Type:** Text Input
 - **Functionality:** Filters the displayed records by the `equipment_material_group`, allowing users to view only specific categories of equipment (e.g., "Electronics," "Machinery").
 - **Validation:** Must match valid material group categories present in the system.
- *Equipment Name Filter:*
 - **Type:** Text Input
 - **Functionality:** Filters the records by the name or description of the equipment.
 - **Validation:** Must match the names or descriptions as listed in the `warehouse_product_list`.
- *Search Box:*
 - **Type:** Global Search Input
 - **Functionality:** Allows users to quickly search across all displayed fields, providing fast access to specific records.

1.1.2. Data Table Display

Columns:

- *Equipment ID:*
 - **Type:** Numeric Display
 - **Data Source:** `equipment_id` from the `warehouse_product_list` table.
 - **Functionality:** Displays the unique identifier for each piece of equipment.
- *Equipment Material Group:*
 - **Type:** Text Display
 - **Data Source:** `equipment_material_group` from the `warehouse_product_list` table.
 - **Functionality:** Displays the material group classification for the equipment, helping users to categorize items.
- *Equipment Name:*
 - **Type:** Text Display
 - **Data Source:** `equipment_name` from the `warehouse_product_list` table.
 - **Functionality:** Displays the name or description of the equipment.

- *Subtype:*
 - **Type:** Text Display
 - **Data Source:** subtype from the warehouse_product_list table.
 - **Functionality:** Displays additional classification details for accounting or inventory management purposes (e.g., "Income Account," "Tax Schedule").

1.1.3. *Action Buttons*

- *Import Warehouse Product Button:*
 - **Type:** Action Button
 - **Functionality:** Opens an interface to import new products into the warehouse inventory. This allows for bulk entry or single entries of new equipment into the warehouse_product_list.
 - **Database Interaction:** Upon import, the new data is added/updated to the warehouse_product_list table and displayed immediately on the screen.

3. *Database Interactions*

3.1 *Warehouse Product List Table*

- **Table Name:** warehouse_product_list
- **Primary Keys:**
 - equipment_id - Unique integer identifier for each product.
- **Foreign Keys:**
 - None, though other tables such as stock_adjustment may reference equipment_id.
- **Triggers:**
 - **On Insert:** Validates that the equipment_id does not duplicate an existing entry. Updates relevant indices for faster search and retrieval.

4. *Business Logic*

4.1 *Real-Time Data Display*

- **Logic:** The screen retrieves data in real-time, with filters applied on the client side based on user input. Any imports are processed immediately, with new products appearing in the list without requiring a page refresh.
- **Concurrency Control:** The system ensures that concurrent imports or updates to the same equipment do not cause conflicts or data corruption.

4.2. *Validation and Error Handling*

Validation Process for Imports

- *File Type Validation:*
 - **Validation Rule:** Only Excel (.xlsx) or CSV (.csv) file formats are accepted.
 - **Error Handling:** If an invalid file format, empty file or file with multiple sheets is uploaded, the system will reject the file and prompt the user to upload a valid file.

- *Column Validation:*
 - **Required Columns:** The import file must contain the following columns:
 - equipment_id (Equipment ID)
 - equipment_material_group (Equipment Material Group)
 - equipment_name (Equipment Name)
 - subtype (Subtype)
 - **Validation Rule:** The file must include all required columns with exact header names. The system is case-sensitive and expects an exact match.
 - **Error Handling:** If any required column is missing or incorrectly named, the system will reject the file and notify the user of the specific issue.
- *Data Validation:*
 - equipment_id:
 - **Type:** Integer
 - **Validation Rule:** Must be a unique, non-empty integer.
 - **Error Handling:** If equipment_id is missing, non-integer, or duplicates an existing entry, the record will be flagged, and the user will be notified.
 - equipment_material_group:
 - **Type:** String
 - **Validation Rule:** Must be a non-empty string, containing only alphabetic characters and spaces.
 - **Error Handling:** If the equipment_material_group is invalid, the record will be flagged.
 - equipment_name:
 - **Type:** String
 - **Validation Rule:** Must be a non-empty string.
 - **Error Handling:** If the equipment_name is empty or contains invalid characters, the record will be flagged.
 - subtype:
 - **Type:** String
 - **Validation Rule:** Must be a non-empty string, containing only alphabetic characters and spaces.
 - **Error Handling:** If the subtype is invalid, the record will be flagged.
- *Update vs. Add Logic:*
 - **Update Logic:** If an equipment_id in the import file matches an existing record, the system updates the record with new information.
 - **Add Logic:** If the equipment_id does not exist, a new record is created.
- *Error Handling:*
 - The system processes valid records and provides a detailed error report for any failed records. Users can correct the errors and re-import the data.

5. Security Considerations

5.1 Access Control

- **Role-Based Access:** Only users with the appropriate permissions (typically warehouse managers or administrators) can access the Warehouse Product List screen and perform imports.
- **Authorization:** Each API request checks the user's role and permissions before allowing data retrieval or import actions.

6. Performance Considerations

6.1 Optimization Techniques

- **Indexing:** Indexes are maintained on equipment_id, equipment_material_group, and equipment_name to ensure fast data retrieval.
- **Pagination:** The data is displayed in paginated views (e.g., 10 entries per page) to reduce load times and improve user experience, especially with large datasets.
- **Caching:** Frequently accessed data may be cached to improve performance, particularly when users repeatedly filter similar records.

Warehouse Product List

Import Warehouse Product

Equipment Material Group Equipment Name
Filter based on Equipme Filter based on Equipme

Show 10 entries Search:

EQUIPMENT ID	EQUIPMENT MATERIAL GROUP	EQUIPMENT NAME	SUBTYPE
17727	001	High-Speed Drill	Income Account
17728	002	Laser Cutter	Expense/COGS Account
17729	003	CNC Milling Machine	Tax Schedule
17730	004	Electric Sander	Class
17731	005	Hydraulic Press	Department
17732	006	3D Printer	Date Created
17733	007	Welding Machine	Last Modified
17734	008	Drill Press	Income Account
17735	009	Band Saw	Expense/COGS Account
17736	010	Surface Grinder	Tax Schedule

Showing 1 to 10 of 14 entries

Previous 1 2 Next

2. Warehouse

The **Warehouse List** screen is a vital part of the Laurel Rental Application, designed to manage and display information about the various warehouse locations used by the company. This screen interacts directly with the warehouse_list database table, providing users with the ability to view, search, and import new warehouse locations into the system.

1. User Interface (UI) Components

1.1 Filters and Search Panel

- *Search Box:*
 - **Type:** Global Search Input
 - **Functionality:** Allows users to search across all displayed fields, such as location and address, to quickly find specific warehouses.
 - **Validation:** Ensures that the input matches records in the warehouse_list table.

1.2 Data Table Display

- *Columns:*
 - *Location:*
 - **Type:** Text Display
 - **Data Source:** location_name from the warehouse_list table.
 - **Functionality:** Displays the name or designation of each warehouse location (e.g., "Bakersfield," "Yuma").
 - *Address:*
 - **Type:** Text Display
 - **Data Source:** address from the warehouse_list table.
 - **Functionality:** Displays the full address of each warehouse, including the street address, city, state, and ZIP code.

1.3 Action Buttons

- *Import Warehouse Button:*
 - **Type:** Action Button
 - **Functionality:** Opens an interface to import new warehouse locations into the system. This feature allows for the bulk addition or single entry of new warehouse details into the warehouse_list.
 - **Database Interaction:** Upon successful import, the new locations are added to the warehouse_list table and immediately displayed on the screen.

3. Database Interactions

3.1 Warehouse List Table

- **Table Name:** warehouse_list
- **Primary Keys:**
 - location_id - Unique integer identifier for each warehouse.
- **Foreign Keys:**
 - None, though other tables, such as stock_adjustment, may reference location_name.
- **Triggers:**

- **On Insert:** Validates that the location_name does not duplicate an existing entry. Updates relevant indices for faster search and retrieval.

4. Business Logic

4.1 Real-Time Data Display

- **Logic:** The screen retrieves data in real-time, with search terms applied on the client side based on user input. Any new imports are processed immediately, with the new warehouse locations appearing in the list without requiring a page refresh.
- **Concurrency Control:** The system ensures that concurrent imports or updates to the same location do not cause conflicts or data corruption.

4.2 Validation and Error Handling

- *Import Warehouse Button*
 - **Functionality:** Allows users to import warehouse location data from Excel or CSV files.
 - **Database Interaction:** Upon successful import, the data is added to or updated in the warehouse_list table.
- *File Type Validation:*
 - **Validation Rule:** Only Excel (.xlsx) or CSV (.csv) file formats are accepted.
 - **Error Handling:** If an invalid file format is uploaded, empty file or multiple sheets, the system will reject the file and prompt the user to upload a valid file.
- *Column Validation:*
 - **Required Columns:** The import file must contain the following columns:
 - location_name (Location Name)
 - address (Full Address)
 - **Validation Rule:** The file must include both columns with exact header names. The system is case-sensitive and expects an exact match.
 - **Error Handling:** If any required column is missing or incorrectly named, the system will reject the file and notify the user of the specific issue.
- *Data Validation:*
 - **location_name:**
 - **Type:** String
 - **Validation Rule:** Must be a non-empty, unique string.
 - **Error Handling:** If location_name is missing or duplicates an existing location, the record will be flagged, and the user will be notified.
 - **address:**
 - **Type:** String
 - **Validation Rule:** Must be a valid, non-empty address format.
 - **Error Handling:** If address is empty or invalid, the record will be flagged.

- *Update vs. Add Logic:*
 - **Update Logic:** If a location_name in the import file matches an existing record, the system updates the record with new information.
 - **Add Logic:** If the location_name does not exist, a new record is created.
- *Error Handling:*
 - The system processes valid records and provides a detailed error report for any failed records. Users can correct the errors and re-import the data.

5. Performance Considerations

5.1 Optimization Techniques

- **Indexing:** Indexes are maintained on location_name and address to ensure fast data retrieval.
- **Pagination:** The data is displayed in paginated views (e.g., 10 entries per page) to reduce load times and improve user experience, especially with large datasets.
- **Caching:** Frequently accessed data may be cached to improve performance, particularly when users repeatedly search similar records.

Warehouse List

Import Warehouse

Show 10 entries

Search:

LOCATION	ADDRESS
Bakersfield	3876 Allen Rd, Bakersfield, CA 93314
Brawley	803 CA-78, Brawley, CA 92227
Colusa	2959 Davison Ct, Colusa, CA 95932
Corning	2920 Hwy 99 W, Corning, CA 96021
Delano	325 Rd 192, Delano, CA 93215
Hanford	10757 Energy St, Hanford, CA 93230
Hollister	2241 OLD AIRLINE HWY, PAICINES, CA 95043
Lodi	1301 Armstrong Rd, Lodi, CA 95242
Old River	9213 Old River Rd, Bakersfield, CA 93311
Yuma	8853 S Avenue 7 E, Yuma, AZ 85365

Showing 1 to 10 of 10 entries

Previous 1 Next

3. Account Managers

The **Account Managers** screen in the Laurel Rental Application is designed to manage and display information about the account managers within the system. This screen interfaces directly with the account_managers database table, allowing users to view, search, and import account manager data from Excel and CSV files. The import function supports both updating existing records and adding new records, with comprehensive validation processes in place.

1. User Interface (UI) Components

1.1 Filters and Search Panel

- *Search Box:*
 - **Type:** Global Search Input
 - **Functionality:** Allows users to search across all displayed fields, such as account manager name and email address, to quickly find specific records.
 - **Validation:** Ensures that the input matches records in the account_managers table.

1.2 Data Table Display

- *Columns:*
 - **Account Managers:**
 - **Type:** Text Display
 - **Data Source:** manager_name from the account_managers table.
 - **Functionality:** Displays the full name of each account manager (e.g., "Brawley Smith," "Brooke Marchy").
 - **Email Address:**
 - **Type:** Text Display
 - **Data Source:** email_address from the account_managers table.
 - **Functionality:** Displays the email address associated with each account manager.

1.3 Action Buttons

- *Import Account Managers Button:*
 - **Type:** Action Button
 - **Functionality:** Opens an interface to import account manager data from Excel or CSV files. This feature allows for the bulk addition or update of account manager details into the account_managers table.
 - **Database Interaction:** Upon successful import, the data is added to or updated in the account_managers table and immediately displayed on the screen.

2. Database Interactions

2.1 Account Managers Table

- **Table Name:** account_managers
- **Primary Keys:**
 - manager_id - Unique integer identifier for each account manager.
- **Foreign Keys:**
 - None, though other tables may reference manager_name or email_address.
- **Triggers:**

- **On Insert or Update:** Ensures that the `manager_name` and `email_address` fields are unique and that any updates do not conflict with existing records.

3. Business Logic

3.1 Real-Time Data Display

- **Logic:** The screen retrieves data in real-time, with search terms applied on the client side based on user input. Any new imports or updates are processed immediately, with the account manager records appearing in the list without requiring a page refresh.
- **Concurrency Control:** The system ensures that concurrent imports or updates to the same account manager do not cause conflicts or data corruption.

3.2 Validation and Error Handling

3.2.1 Validation Process for Imports

- *File Type Validation:*
 - **Validation Rule:** The import function only accepts Excel (.xlsx) or CSV (.csv) file formats.
 - **Error Handling:** If the uploaded file is not in one of these formats, is empty or have multiple sheets, the system will reject the file and prompt the user to upload a valid file.
- *Column Validation:*
 - **Required Columns:** The import file must contain the following columns:
 - `manager_name` (Account Manager's Full Name)
 - `email_address` (Account Manager's Email Address)
 - **Validation Rule:** The file must include both columns with valid headers exactly matching the expected names. Case sensitivity and exact match are required.
 - **Error Handling:** If any required column is missing or incorrectly named, the system will reject the file and notify the user of the specific issue.
- *Data Validation:*
 - **manager_name:**
 - **Type:** String
 - **Validation Rule:** Must be a non-empty string, containing only alphabetic characters and spaces.
 - **Error Handling:** If the `manager_name` contains invalid characters or is left empty, the record will be flagged, and the user will be informed of the invalid entry.
 - **email_address:**
 - **Type:** String
 - **Validation Rule:** Must be a valid email format, unique in the database.
 - **Error Handling:** If the `email_address` is not in a valid format or if it duplicates an existing email in the `account_managers` table, the record will be flagged, and the user will be notified.

3.2.2 Update vs. Add Logic

- *Update Logic:*
 - If an email_address in the import file matches an existing record in the account_managers table, the corresponding record will be updated with the new information from the import file.
- *Add Logic:*
 - If the email_address does not exist in the account_managers table, a new record will be created with the data provided in the import file.

3.2.3 Error Handling:

- *Individual Record Handling:*
 - **Rule:** If individual records within the import file fail validation, only those records will be rejected, while the rest of the file will be processed.
 - **Error Reporting:** A detailed report is provided after the import, listing any errors encountered and the specific rows affected. The user can then correct these errors and re-import the file.

4. Performance Considerations

4.1 Optimization Techniques

- **Indexing:** Indexes are maintained on manager_name and email_address to ensure fast data retrieval.
- **Pagination:** The data is displayed in paginated views (e.g., 10 entries per page) to reduce load times and improve user experience, especially with large datasets.
- **Caching:** Frequently accessed data may be cached to improve performance, particularly when users repeatedly search similar records.

Account Managers

Import Account Managers

Show 10 entries

Search

ACCOUNT MANAGERS	EMAIL ADDRESS
Brawley Smith	brawleysmith@laurel.com
Brooke Marchy	brookemarchy@laurel.com
Colusa John	colusa@laurel.com
Delano	delanomosby@laurel.com
Hanford Page	hanford.p@laurel.com
Overland Stockyard	overland.stockyard@laurel.com
Paramdeep Chohan	pchohan@laurel.com
Steven Roland	Steven@laurel.com
Wackman Ranch	wackmanranch@laurel.com
Zachary Scott	Zachary@laurel.com

Showing 1 to 10 of 10 entries

Previous 1 Next

4. Customers

The **Customer List** screen in the Laurel Rental Application is designed to manage and display information about customers within the system. This screen interfaces directly with the customer_list

database table, allowing users to view, search, and import customer data from Excel or CSV files. The import function supports both updating existing records and adding new records, with comprehensive validation processes in place.

1. User Interface (UI) Components

1.1 Filters and Search Panel

- *Search Box:*
 - **Type:** Global Search Input
 - **Functionality:** Allows users to search across all displayed fields, such as internal_id, customer_id and name to quickly find specific customer records.
 - **Validation:** Ensures that the input matches records in the customer_list table.

1.2 Data Table Display

- *Columns:*
 - **Internal ID:**
 - **Type:** Numeric Display
 - **Data Source:** internal_id from the customer_list table.
 - **Functionality:** Displays the unique internal identifier for each customer.
 - **ID:**
 - **Type:** Text Display
 - **Data Source:** customer_id from the customer_list table.
 - **Functionality:** Displays the system-wide unique identifier for each customer.
 - **Name:**
 - **Type:** Text Display
 - **Data Source:** name from the customer_list table.
 - **Functionality:** Displays the name of the customer.

1.3 Action Buttons

- *Import Customer Button:*
 - **Type:** Action Button
 - **Functionality:** Opens an interface to import customer data from Excel or CSV files. This feature allows for the bulk addition or update of customer details into the customer_list.
 - **Database Interaction:** Upon successful import, the data is added to or updated in the customer_list table and immediately displayed on the screen.

2. Database Interactions

2.1 Customer List Table

- **Table Name:** customer_list
- **Primary Keys:**
 - internal_id - Unique integer identifier for each customer.
- **Foreign Keys:**
 - None, though other tables may reference customer_id.

- **Triggers:**
 - **On Insert or Update:** Ensures that the `customer_id` and `email_address` fields are unique and that any updates do not conflict with existing records.

4. Business Logic

4.1 Real-Time Data Display

- **Logic:** The screen retrieves data in real-time, with search terms applied on the client side based on user input. Any new imports or updates are processed immediately, with the customer records appearing in the list without requiring a page refresh.
- **Concurrency Control:** The system ensures that concurrent imports or updates to the same customer do not cause conflicts or data corruption.

4.2 Validation and Error Handling

4.2.1 Validation Process for Imports

- *File Type Validation:*
 - **Validation Rule:** The import function only accepts Excel (.xlsx) or CSV (.csv) file formats.
 - **Error Handling:** If the uploaded file is not in one of these formats or is empty/with multiple sheets, the system will reject the file and prompt the user to upload a valid file.
- *Column Validation:*
 - **Required Columns:** The import file must contain the following columns:
 - `internal_id` (Internal ID)
 - `customer_id` (Customer ID)
 - `name` (Customer Name)
 - **Validation Rule:** The file must include all required columns with exact header names. The system is case-sensitive and expects an exact match.
 - **Error Handling:** If any required column is missing or incorrectly named, the system will reject the file and notify the user of the specific issue.
- *Data Validation:*
 - `internal_id`:
 - **Type:** Integer
 - **Validation Rule:** Must be a unique, non-empty integer.
 - **Error Handling:** If `internal_id` is missing, non-integer, or duplicates an existing entry, the record will be flagged, and the user will be notified.
 - `customer_id`:
 - **Type:** String
 - **Validation Rule:** Must be a non-empty, unique string.
 - **Error Handling:** If `customer_id` is missing or duplicates an existing ID, the record will be flagged.
 - `name`:
 - **Type:** String

- **Validation Rule:** Must be a non-empty string containing the customer's full name.
- **Error Handling:** If name is empty, the record will be flagged.

4.2.2 Update vs. Add Logic

- **Update Logic:**
 - If an internal_id or customer_id in the import file matches an existing record, the corresponding record will be updated with the new information from the import file.
- **Add Logic:**
 - If the internal_id or customer_id does not exist in the customer_list table, a new record will be created with the data provided in the import file.

4.2.3 Error Handling:

- **Individual Record Handling:**
 - **Rule:** If individual records within the import file fail validation, only those records will be rejected, while the rest of the file will be processed.
 - **Error Reporting:** A detailed report is provided after the import, listing any errors encountered and the specific rows affected. The user can then correct these errors and re-import the file.

4. Performance Considerations

4.1 Optimization Techniques

- **Indexing:** Indexes are maintained on internal_id, customer_id, and name to ensure fast data retrieval.
- **Pagination:** The data is displayed in paginated views (e.g., 10 entries per page) to reduce load times and improve user experience, especially with large datasets.
- **Caching:** Frequently accessed data may be cached to improve performance, particularly when users repeatedly search similar records.

Customer List

Import Customer

Show 10 entries

Search:

INTERNAL ID	ID	NAME	SALES REP	BILLING ADDRESS 1	BILLING ADDRESS 2	BILLING CITY	BILLING STATE/PROVINCE	BILLING ZIP	BILLING COUNTRY
298809	C006383	Anndria Barcelona	-	-	-	-	-	-	-
298811	C006384	Unit B Irrigation & Drainage	Paul Leimgruber	-	-	-	-	-	-
298854	C006385	MAS Property Holdings LLC	Brooke Marchy	506 Couch St.	-	Vallejo	CA	94560	United States
298871	C006386	City of Colusa	Patrick Stern	425 Webster Street	-	Colusa	CA	95932	United States
298902	C006387	FFR Animas, LLC.	-	-	-	-	-	-	-
298903	C006388	Fantastic Farms, LLC	Zachary Scott	1621 E 27th St	-	Los Angeles	CA	90011	United States
298904	C006389	Colusa Golf & Country Club	Patrick Stern	PO Box 827	-	Colusa	CA	95932	United States
298911	C006390	Waybright Orchards	William Martin	21420 Gaines Ln	-	Anderson	CA	96007	United States
298916	C006391	Venter Family Estate	Brooke Marchy	2144 Las Amigas Rd.	-	Napa	CA	94559	United States
298921	C006392	Coastal Vineyard Services	-	-	-	-	-	-	-

Showing 1 to 10 of 100 entries

Previous 1 2 3 4 5 ... 10 Next

Rent Process

1. Main Screen Description

Page Name: Order Management – Orders

Order

Open Recurring OrderOpen Return Order

Order

Import OrderCreate Order

Show 10 entries

Search:

ORDER ID	DELIVERY	STATUS	CUSTOMER	ORDER AMOUNT	PROJECT MANAGER	START DATE	END DATE	RECCURENCE TYPE	LAST UPDATED DATE	ACTIONS
C00240200001	C00240200001-1 C00240200001-2	Pick Up Check out	C005888 Rocha Junior Berry Farms LLC	\$ 1500.00	Matthew Waterman	-	-	Recurring	25/10/2024	
C00240200004	C00240200004-1 C00240200004-2	Delivered Check In	C005897 Rio Farms	\$ 1800.00	Matthew Waterman	01/04/2024	30/09/2024	Recurring	25/10/2024	
C00240200005	C00240200005-1 C00240200005-2	New New	C005897 Rio Farms	\$ 1500.00	Matthew Waterman	01/04/2024	30/09/2024	One Time	25/10/2024	
C00240200007	C00240200007-1 C00240200007-2	Create Return Delivery Create Return Delivery	C006219 Braga Ranch	\$ 1500.00	Matthew Waterman	01/04/2024	30/09/2024	Recurring	25/10/2024	
C00240200008	C00240200008-1 C00240200008-2	Pick Up Check out	C005414 3-Brand Cattle	\$ 1500.00	Anthony Freitas	01/04/2024	31/05/2024	Recurring	25/10/2024	
C00240200009	C00240200009-1 C00240200009-2	Pick Up Pick Up	C005477 H & H Farms Inc.	\$ 1800.00	Anthony Freitas	01/04/2024	30/09/2024	One Time	25/10/2024	
C00240200010	C00240200010-1 C00240200010-2	Create Delivery Create Delivery	C005477 H & H Farms Inc.	\$ 1800.00	Anthony Freitas	01/04/2024	30/09/2024	Recurring	25/10/2024	
C00240200011	C00240200011-1 C00240200011-2	Pick Up Check In	C005941 Top Flavor Farms San Juan	\$ 1500.00	Matthew Waterman	01/04/2024	30/09/2024	One Time	25/10/2024	
C00240200012	C00240200012-1 C00240200012-2	Check In Check In	C005941 Top Flavor Farms San Juan	\$ 1800.00	Matthew Waterman	01/04/2024	30/09/2024	One Time	25/10/2024	
C00240200013	C00240200013-1 C00240200013-2	Check In Check In	C003623 Gledstone Land	\$ 1800.00	Anthony Freitas	-	-	One Time	25/10/2024	

Showing 1 to 10 of 20 entries

PreviousNext

1. Purpose

The Order Management screen is designed for administrators and project managers to manage rental orders, including viewing, creating, and updating orders. This interface provides an overview of all orders, allows for the creation of new orders, and offers tools to manage existing orders, including recurring and return orders.

2. UI Components

1.1 Order List Panel

1. Component: Order Table

- *Position:* Central area of the screen.
- *Structure:*
 - *Columns:*
 - **Order ID:** Displays the unique identifier for each order. Clicking the Order ID navigates to a detailed view of the order. Hyperlinked to order overview page.
 - **Delivery:** Shows the associated delivery IDs linked to the order. Each delivery ID is clickable, leading to its specific details. Hyperlinked to specific delivery and its active stage.

- **Status:** Indicates the current status of the order, such as "Pick Up", "Check Out", "Delivered", etc. Each status is color-coded for quick identification.
 - **Customer:** Displays the customer associated with the order, identified by a customer ID or name.
 - **Order Amount:** Shows the total monetary value of the order.
 - **Project Manager:** Indicates the project manager responsible for the order.
 - **Start Date:** Displays the start date of the order.
 - **End Date:** Displays the end date of the order.
 - **Recurrence Type:** Indicates whether the order is recurring or a one-time order.
 - **Last Updated Date:** Shows the last date when the order was modified.
 - **Actions:** Provides icons for editing, deleting, and managing the order.
 - *Pagination:*
 - **Position:** Bottom of the table.
 - **Function:** Allows navigation through multiple pages of orders. The number of entries per page can be adjusted using a dropdown.
2. *Component:* Actions Column
- *Position:* Rightmost column of the order table.
 - *Structure:*
 - **Edit Button:** A pencil icon that opens the "Edit Order" modal, allowing the admin to modify order details.
 - **Delete Button:** A trash icon that allows the admin to delete the order (only in status new)
 - **Delivery Management Button:** An icon representing a truck, which opens a modal for managing deliveries related to the order.
 - **Order Analytics Button:** A bar chart icon that provides access to analytics related to the order's performance and history.
 - *Interaction:*
 - **Edit Button:** Clicking this button opens the "Edit Order" modal where the admin can change the order's details, including customer information, items, and delivery schedules.
 - **Delete Button:** Clicking this button prompts the admin to confirm the deletion of the order. If confirmed, the order is removed from the system.
 - **Delivery Management Button:** Opens a modal that provides detailed delivery schedules and options to update delivery status.
 - **Order Analytics Button:** Opens a dashboard that presents analytical data related to the order.
3. *Component:* Search Box
- *Position:* Top right corner of the order table.
 - *Function:* Allows admins to search for orders by Order ID, customer name, project manager, or status.

- *Interaction:* Typing into the search box filters the table to show only orders matching the search criteria.

1.2 Create Order Modal (data source order_line_items and delivery_details grouped by order_id, delivery_id)

1. *Component:* Create Order Modal

- *Trigger:* Activated by clicking the "Create Order" button located in the top right corner of the Order Management screen.
- *Structure:*
 - *Fields:*
 - **Customer:** Dropdown to select the customer associated with the order.
 - **Link NetSuite:** Button to link the order to NetSuite for financial tracking and reporting.
 - **Account Manager:** Dropdown to select the account manager responsible for the order.
 - **Region:** Input field to specify the region related to the order.
 - **Customer Email:** Input field for the customer's email address.
 - **Pick-Up Location:** Input field for the location where items will be picked up.
 - **Delivery Location:** Input field for the delivery location of the items.
 - **Water Source:** Input field specifying the water source if applicable to the order.
 - **Crop:** Input field for specifying the crop involved in the order, if applicable.
 - **Items:**
 - **Items No.:** Dropdown to select the item number.
 - **Items:** Dropdown to select the material name.
 - **Quantity:** Input field for the quantity of items.
 - **Per Unit Price:** Input field for the price per unit.
 - **Rent Amount:** Input field to specify the total rent amount.
 - **Rent Per Month:** Input field to specify the monthly rent amount.
 - **Recurring Order:** Checkbox to indicate if the order is recurring.
 - **Recurrence Settings:**
 - **Repeats:** Dropdown to select the frequency of recurrence (e.g., daily, weekly, monthly).
 - **For:** Input field to specify the duration of recurrence.
 - **From:** Date picker to specify the start date of the recurring order.
 - **To:** Date picker to specify the end date of the recurring order.
 - *Buttons:*
 - **Close:** Closes the modal without saving changes.

- **Add Order:** Saves the new order to the system, including all specified details and scheduling.

2. *Interaction:*

- **Add Order Button:** When clicked, the system validates all input fields and then adds the new order to the database. The new order is displayed in the order table upon successful addition.

3. Functional Requirements

2.1 Order Management

1. *Creating a New Order:*

- **Trigger:** Clicking the "Create Order" button.
- **Input Fields:** The admin must fill in all necessary fields related to the customer, items, and scheduling.
- **Validation:** The system checks that all required fields are filled and that values are valid (e.g., dates, amounts).
- **Database Update:** Upon clicking "Add Order," a new order record is created in the order management system's database.

2. *Editing an Existing Order:*

- **Trigger:** Clicking the Edit button for an order.
- **Fields:** The admin can modify details such as customer information, item details, and delivery schedules.
- **Database Update:** Upon clicking "Save," the order's record in the database is updated with the new details.

3. *Deleting an Order:*

- **Trigger:** Clicking the Delete button for an order.
- **Confirmation:** The system prompts the admin to confirm the deletion.
- **Database Update:** Upon confirmation, the order is removed from the system's database.

2.2 Delivery Management

1. *Managing Deliveries:*

- **Trigger:** Clicking the Delivery Management Button.
- **Modal:** Opens a delivery management modal where the admin can update delivery schedules, statuses, and add notes related to each delivery.
- **Database Update:** Changes are saved to the associated delivery records in the database.

2.3 Order Analytics

1. Viewing Analytics:

- **Trigger:** Clicking the Order Analytics Button.
- **Dashboard:** Opens a dashboard that displays key metrics and historical data related to the order's performance, such as delivery success rates, delays, and customer satisfaction scores.
- **Data Source:** The dashboard pulls data from the order management system's analytics database.

2.4 Search and Filtering

1. Search Functionality:

- **Trigger:** Typing in the Search box filters the displayed orders in the table.
- **Database Query:** The search function queries the order database, filtering results based on order ID, customer name, project manager, or status.

4. Error Handling





















4.1 Validation Errors

- **Missing Fields:** If required fields (e.g., customer, items, delivery locations) are not filled in the Create or Edit Order modals, the system should highlight the missing fields and provide a message indicating that all fields are required.
- **Invalid Data:** If invalid data is entered (e.g., incorrect date formats, negative quantities), the system should prevent the order from being saved and provide a descriptive error message.

4.2 Backend Errors

- **Database Update Failure:**
 - **Scenario:** If there is an issue updating the orders or deliveries tables (e.g., network issues, database errors), the system should display an error message to the admin and suggest retrying or contacting support.

Page Name: Order Management – Open Recurring Orders

Order Open Recurring Order Open Return Order									
Recurring Order									
Show 10 of 32 entries									
ORIGINAL ORDER ID	RECURRING ORDER ID	CUSTOMER	ORDER AMOUNT	PROJECT MANAGER	RECURRING TYPE	RECURRING START	RECURRING END	NUMBER OF ORDERS	ACTIONS
020240300055	OR020262300055	Vaimont Industries	\$3300.00	LINDA GARCIA	Weekly	01/05	31/12	3	 
020240300055	OR020270300055	Vaimont Industries	\$3300.00	WILLIAM MARTINEZ	Weekly	01/05	31/12	3	 
020240300055	OR020280300055	China Leasing Group	\$3750.00	ELIZABETH HERNANDEZ	Monthly	15/05	30/09	5	 
020240300055	OR020290300055	McWane	\$4500.00	DAVID MILLER	Yearly	01/05	31/12	2	 
020240300057	OR020340300057	Mueller Industries	\$2700.00	SUSAN DAVIS	Monthly	15/05	30/11	5	 
020240300058	OR020360300058	Mueller Industries	\$2700.00	JAMES SMITH	Monthly	15/05	30/11	5	 
020240300058	OR020370300058	Nord Stream AG	\$900.00	MARY JOHNSON	Monthly	15/05	30/11	5	 
020240300058	OR020380300058	Nord Stream AG	\$900.00	ROBERT WILLIAMS	Yearly	15/05	30/11	5	 
020240700045	OR020460700045	Saint-Gobain	\$900.00	JAMES SMITH	Yearly	01/05	31/12	5	 
020240700045	OR020707000045	Wavin	\$2000.00	MARY JOHNSON	Monthly	01/05	30/09	2	 
Showing 1 to 10 of 32 entries									
Previous 1 2 3 4 Next									

1. Purpose

The Open Recurring Order screen is designed to manage and track recurring orders within the rental process application. This screen lists all recurring orders that are still active and those that remain open up to 30 days after the contract end date. The interface allows administrators and project managers to view, edit, and manage recurring orders efficiently.

1. UI Components

1.1 Recurring Order List Panel

1. *Component:* Recurring Order Table

- *Position:* Central area of the screen.
- *Structure:*
 - **Columns and Database Mapping:**
 - **Original Order ID:** Displays the unique identifier for the original order that initiated the recurring order series.
 - **Source Table:** orders
 - **Column:** order_id
 - **Recurring Order ID:** Displays the unique identifier for the specific instance of the recurring order.
 - **Source Table:** recurring_orders
 - **Column:** recurring_order_id
 - **Customer:** Displays the customer associated with the recurring order.
 - **Source Table:** customers
 - **Column:** customer_name
 - **Order Amount:** Shows the total monetary value of the recurring order.
 - **Source Table:** recurring_orders
 - **Column:** order_amount
 - **Project Manager:** Indicates the project manager responsible for the recurring order.
 - **Source Table:** project_managers
 - **Column:** project_manager_name
 - **Recurring Type:** Indicates the frequency of recurrence (e.g., Weekly, Monthly, Yearly).
 - **Source Table:** recurring_orders
 - **Column:** recurrence_type

- **Recurring Start:** Displays the start date of the recurring order series.
 - **Source Table:** recurring_orders
 - **Column:** recurrence_start_date
 - **Recurring End:** Displays the end date of the recurring order series.
 - **Source Table:** recurring_orders
 - **Column:** recurrence_end_date
 - **Number of Orders:** Displays the number of orders that have been generated from this recurring order template.
 - **Source Table:** recurring_orders
 - **Column:** number_of_orders
 - **Actions:** Provides icons for editing and deleting the recurring order.
 - **Source Table:** Not applicable (frontend action triggers)
 - **Pagination:**
 - **Position:** Bottom of the table.
 - **Function:** Allows navigation through multiple pages of recurring orders. The number of entries per page can be adjusted using a dropdown.
2. *Component:* Actions Column
- *Position:* Rightmost column of the recurring order table.
 - *Structure:*
 - **Edit Button:** A pencil icon that opens the "Edit Recurring Order" modal, allowing the admin to modify recurring order details.
 - **Trigger Action:** Opens the Edit Recurring Order modal (no database mapping).
 - **Delete Button:** A trash icon that allows the admin to delete the recurring order.
 - **Trigger Action:** Deletes the recurring order (calls API endpoint to remove from recurring_orders table).
 - *Interaction:*
 - **Edit Button:** Clicking this button opens the "Edit Recurring Order" modal where the admin can change the details of the recurring order, including frequency, start and end dates, and associated customer and project manager.
 - **Delete Button:** Clicking this button prompts the admin to confirm the deletion of the recurring order. If confirmed, the recurring order is removed from the system.
3. *Component:* Search Box
- *Position:* Top right corner of the recurring order table.
 - *Function:* Allows admins to search for recurring orders by Original Order ID, Customer Name, Project Manager, or Recurrence Type.
 - *Interaction:* Typing into the search box filters the table to show only recurring orders matching the search criteria.
 - *Database Mapping:* Queries the recurring_orders, orders, customers, project_managers tables to filter results.

2. Functional Requirements

2.1 Recurring Order Management

- **Listing Open Recurring Orders:**
 - **Trigger:** The screen automatically lists all recurring orders where the recurrence_end_date is within the last 30 days or is still in the future.
 - **Database Query:**
 - The system performs a query on the recurring_orders table to retrieve orders where:
 - recurrence_end_date >= CURRENT_DATE - INTERVAL '30 days' OR recurrence_end_date IS NULL (indicating the order is still active).
 - The resulting list is displayed in the Recurring Order Table.
- **Editing a Recurring Order:**
 - **Trigger:** Clicking the Edit button for a recurring order.
 - **Fields:** The admin can modify details such as customer information, project manager, recurrence type, and dates.
 - **Database Update:** Upon clicking "Save," the recurring order's record in the recurring_orders table is updated with the new details.
- **Deleting a Recurring Order:**
 - **Trigger:** Clicking the Delete button for a recurring order.
 - **Confirmation:** The system prompts the admin to confirm the deletion.
 - **Database Update:** Upon confirmation, the recurring order is removed from the recurring_orders table.

2.2 Search and Filtering

- **Search Functionality:**
 - **Trigger:** Typing in the Search box filters the displayed recurring orders in the table.
 - **Database Query:** The search function queries the recurring_orders, orders, customers, and project_managers tables, filtering results based on the Original Order ID, Recurring Order ID, Customer Name, Project Manager, or Recurrence Type.

4. Error Handling

4.1 Validation Errors

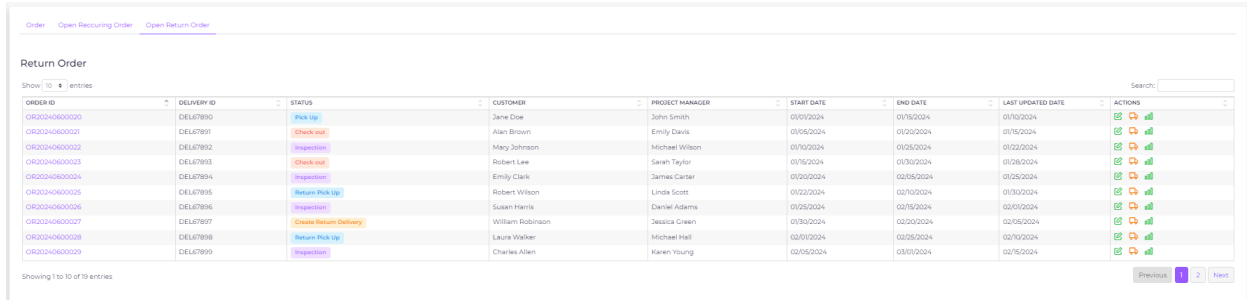
- **Missing Fields:** If required fields (e.g., recurrence type, start and end dates, customer) are not filled in the Create or Edit Recurring Order modals, the system should highlight the missing fields and provide a message indicating that all fields are required.
- **Invalid Data:** If invalid data is entered (e.g., incorrect date formats, negative order amounts), the system should prevent the recurring order from being saved and provide a descriptive error message.

4.2 Backend Errors

1. Database Update Failure:

- **Scenario:** If there is an issue updating the recurring_orders or orders tables (e.g., network issues, database errors), the system should display an error message to the admin and suggest retrying or contacting support.

Page Name: Order Management – Open Return Deliveries



The screenshot shows a web application interface for managing return orders. At the top, there are navigation links: "Order", "Open Recurring Order", and "Open Return Order". Below these is a section titled "Return Order" with a "Show 10 entries" dropdown and a search bar. The main part of the screen is a table with the following columns: ORDER ID, DELIVERY ID, STATUS, CUSTOMER, PROJECT MANAGER, START DATE, END DATE, LAST UPDATED DATE, and ACTIONS. The table contains 10 rows of data, each representing a return order. The STATUS column has color-coded buttons: "Pick Up" (blue), "Check Out" (orange), "Inspection" (yellow), "Return Pick Up" (blue), "Create Return Delivery" (orange), and "Inspection" (yellow). The ACTIONS column contains icons for edit, delete, and a green checkmark. At the bottom, there is a pagination bar showing "Showing 1 to 10 of 10 entries" and buttons for "Previous", "1", "2", and "Next".

ORDER ID	DELIVERY ID	STATUS	CUSTOMER	PROJECT MANAGER	START DATE	END DATE	LAST UPDATED DATE	ACTIONS
OR20240600020	DEL67890	Pick Up	Jane Doe	John Smith	01/01/2024	01/10/2024	01/10/2024	
OR20240600021	DEL67891	Check Out	Alan Brown	Emily Davis	01/05/2024	01/20/2024	01/15/2024	
OR20240600022	DEL67892	Inspection	Mary Johnson	Michael Wilson	01/10/2024	01/25/2024	01/22/2024	
OR20240600023	DEL67893	Check Out	Robert Lee	Sarah Taylor	01/15/2024	01/30/2024	01/28/2024	
OR20240600024	DEL67894	Inspection	Emily Clark	James Carter	01/20/2024	02/05/2024	01/25/2024	
OR20240600025	DEL67895	Return Pick Up	Robert Wilson	Linda Scott	01/22/2024	02/10/2024	01/30/2024	
OR20240600026	DEL67896	Inspection	Susan Harris	Daniel Adams	01/25/2024	02/15/2024	02/01/2024	
OR20240600027	DEL67897	Create Return Delivery	William Robinson	Jessica Green	01/30/2024	02/20/2024	02/05/2024	
OR20240600028	DEL67898	Return Pick Up	Laura Walker	Michael Hall	02/01/2024	02/25/2024	02/10/2024	
OR20240600029	DEL67899	Inspection	Charles Allen	Karen Young	02/05/2024	03/01/2024	02/16/2024	

Purpose:

The Open Return Order screen is designed to manage and track return orders within the rental process application. This screen lists all return orders associated with deliveries that have not been completed. The data for this screen is sourced from the delivery_details table, specifically where the delivery_type is "Return Delivery" and deliver_to_customer is False. The interface allows administrators and project managers to view, edit, and manage return orders efficiently.

1. UI Components

1.1 Return Order List Panel

1. Component: Return Order Table

- **Position:** Central area of the screen.
- **Structure:**
 - **Columns and Database Mapping:**
 - **Order ID:** Displays the unique identifier for each return order.
 - **Source Table:** delivery_details
 - **Column:** order_id
 - **Delivery ID:** Displays the unique identifier for the delivery associated with the return order.
 - **Source Table:** delivery_details
 - **Column:** delivery_id
 - **Status:** Indicates the current status of the return order, such as "Pick Up", "Check Out", "Inspection", etc. Each status is color-coded for quick identification.
 - **Source Table:** delivery_details
 - **Column:** status

- **Customer:** Displays the customer associated with the return order.
 - **Source Table:** delivery_details
 - **Column:** customer_name
 - **Project Manager:** Indicates the project manager responsible for the return order.
 - **Source Table:** delivery_details
 - **Column:** project_manager_name
 - **Start Date:** Displays the start date of the return order.
 - **Source Table:** delivery_details
 - **Column:** start_date
 - **End Date:** Displays the end date of the return order.
 - **Source Table:** delivery_details
 - **Column:** end_date
 - **Last Updated Date:** Shows the last date when the return order was modified.
 - **Source Table:** delivery_details
 - **Column:** updated_at
 - **Actions:** Provides icons for editing, managing, and viewing analytics related to the return order.
 - **Source Table:** Not applicable (frontend action triggers)
 - **Pagination:**
 - **Position:** Bottom of the table.
 - **Function:** Allows navigation through multiple pages of return orders. The number of entries per page can be adjusted using a dropdown.
2. **Component: Actions Column**
- **Position:** Rightmost column of the return order table.
 - **Structure:**
 - **Edit Button:** A pencil icon that opens the "Edit Return Order" modal, allowing the admin to modify return order details.
 - **Trigger Action:** Opens the Edit Return Order modal (no database mapping).
 - **Delivery Management Button:** An icon representing a truck, which opens a modal for managing the delivery associated with the return order.
 - **Trigger Action:** Opens the Delivery Management modal (retrieves data from delivery_details table).
 - **Order Analytics Button:** A bar chart icon that provides access to analytics related to the return order's performance and history.
 - **Trigger Action:** Opens the Analytics Dashboard (pulls data from analytics database or data warehouse).
 - **Interaction:**
 - **Edit Button:** Clicking this button opens the "Edit Return Order" modal where the admin can change the details of the return order, including customer information, items, and delivery status.

- **Delivery Management Button:** Opens a modal that provides detailed delivery schedules and options to update delivery status.
 - **Order Analytics Button:** Opens a dashboard that presents analytical data related to the return order.
3. **Component:** Search Box
- **Position:** Top right corner of the return order table.
 - **Function:** Allows admins to search for return orders by Order ID, Delivery ID, Customer Name, Project Manager, or Status.
 - **Interaction:** Typing into the search box filters the table to show only return orders matching the search criteria.
 - **Database Mapping:** Queries the delivery_details table to filter results based on the specified criteria.

2. Functional Requirements

2.1 Return Order Management

1. **Listing Open Return Orders:**
 - **Trigger:** The screen automatically lists all return orders associated with deliveries that have not been completed.
 - **Database Query:**
 - The system performs a query on the delivery_details table to retrieve orders where:
 - delivery_type = 'Return Delivery' AND deliver_to_customer = False AND status != 'Completed'
 - The resulting list is displayed in the Return Order Table.
2. **Editing a Return Order:**
 - **Trigger:** Clicking the Edit button for a return order.
 - **Fields:** The admin can modify details such as customer information, project manager, status, and delivery information.
 - **Database Update:** Upon clicking "Save," the return order's record in the delivery_details table is updated with the new details.
3. **Managing the Delivery Associated with a Return Order:**
 - **Trigger:** Clicking the Delivery Management Button.
 - **Modal:** Opens a delivery management modal where the admin can update delivery schedules, statuses, and add notes related to the delivery.
 - **Database Update:** Changes are saved to the associated delivery record in the delivery_details table.

2.2 Search and Filtering

- **Search Functionality:**
 - **Trigger:** Typing in the Search box filters the displayed return orders in the table.
 - **Database Query:** The search function queries the delivery_details table, filtering results based on the Order ID, Delivery ID, Customer Name, Project Manager, or Status.

4. Error Handling

4.1 Validation Errors

- **Missing Fields:** If required fields (e.g., delivery ID, status, customer) are not filled in the Create or Edit Return Order modals, the system should highlight the missing fields and provide a message indicating that all fields are required.
- **Invalid Data:** If invalid data is entered (e.g., incorrect date formats, negative order amounts), the system should prevent the return order from being saved and provide a descriptive error message.

4.2 Backend Errors

- **Database Update Failure:**
 - **Scenario:** If there is an issue updating the delivery_details table (e.g., network issues, database errors), the system should display an error message to the admin and suggest retrying or contacting support.

Rental Process

General Notes

Navigation & Steps:

- Once the user clicks on "Next" and confirms a pending step, that step becomes unchangeable.
- Users can revisit previously completed steps for reference, but they cannot make any changes.
- Users cannot proceed to the next screen without completing the current screen.
- The status will change to "Completed" at the delivery level once the inspection is completed.
- The status displays as "New" if no delivery is created for the order.
- **Delete Option:** Users can delete an order if it is in the "New" status, meaning no delivery has been created for the order. Delivery and return delivery can be deleted if the subsequent steps have not been completed. A confirmation popup window will appear before the deletion is finalized.



Rental Process
Laurel.pdf

Order Overview Page

Purpose:

The "Order Overview" screen provides a comprehensive summary of the order details, allowing users to quickly review all relevant information. This includes customer details, internal order management

details, order duration, and line item details. The screen is intended to ensure that all order information is accurately captured and readily accessible for reference or auditing purposes.

UI/UX Description:

General Layout:

- **Header Section:**
 - **Order Overview Title:** Positioned at the top-left corner of the screen, with the order number prominently displayed next to it.

Sections:

1. **Customer Details:**
 - **Positioning:** Displayed in a horizontal card format, aligned at the top of the screen.
 - **Fields:**
 - **Customer ID:** Displayed with a label.
 - **Customer Name:** Displayed next to the customer ID.
 - **Customer Email:** Positioned next to the customer's name.
 - **Delivery Location:** Positioned on the far right within the same card.
 - **Design:** Use a card layout with subtle borders and shading to separate the "Customer Details" from other sections.
2. **Internal Details:**
 - **Positioning:** Positioned directly below the "Customer Details" section.
 - **Fields:**
 - **Account Manager:** Displayed at the left.
 - **Link Netsuite:** Positioned next to the account manager.
 - **Region:** Positioned next to the Link Netsuite field.
 - **Pick-Up Location:** Positioned next to the Region field.
 - **Water Source:** Positioned next to the pick-up location.
 - **Crop:** Positioned next to the water source.
 - **Design:** Similar card layout as "Customer Details" but with slightly darker shading to create visual hierarchy.
3. **Duration Details:**
 - **Positioning:** Positioned beneath the "Internal Details" section.
 - **Fields:**
 - **Recurring Order ID:** Positioned at the top-left.
 - **Rental Period:** Positioned directly below the Recurring Order ID.
 - **Recurring Order Number List:** Positioned to the right of the Rental Period.
 - **Design:** Use a visually distinct card with a light background to make this section stand out. Consider using icons to represent recurring orders and rental periods to make the information more intuitive.
4. **Line Details:**
 - **Positioning:** Positioned at the bottom of the screen.
 - **Fields:**

- **Item No.:** Displayed at the left.
- **Item Name:** Positioned next to the Item No.
- **Quantity:** Positioned next to the Item Name.
- **Per Unit Price:** Positioned next to the Quantity field.
- **Rent Amount:** Positioned next to the Per Unit Price.
- **Rent Per Month:** Positioned next to the Rent Amount field.
- **Design:** Use a table format with alternating row colors to improve readability. Headers should be bold, and use borders to separate the rows and columns clearly.

Interactions:

- **Hover Effects:** Add subtle hover effects on cards and buttons to improve user interaction, such as changing the background color or slightly raising the card.
- **Responsive Design:** Ensure the layout is responsive, adjusting to different screen sizes by stacking sections vertically on smaller screens.

Technical Specification:

1. Customer Details:

- **Customer ID:**
 - **Database Reference:** customer_list.customer_id
 - **Changeable:** No
- **Customer Name:**
 - **Database Reference:** customer_list.customer_name
 - **Changeable:** No
- **Customer Email:**
 - **Database Reference:** customer_list.customer_email
 - **Changeable:** No
- **Delivery Location:**
 - **Database Reference:** order_line_items.delivery_location
 - **Changeable:** No

2. Internal Details:

- **Account Manager:**
 - **Database Reference:** account_managers.manager_name
 - **Changeable:** No
- **Link Netsuite:**
 - **Database Reference:** order_line_items.netsuite_link
 - **Changeable:** No
- **Region:**
 - **Database Reference:** order_line_items.region
 - **Changeable:** No
- **Pick-Up Location:**
 - **Database Reference:** order_line_items.pickup_location
 - **Changeable:** No

- Water Source:
 - Database Reference: order_line_items.water_source
 - Changeable: No
- Crop:
 - Database Reference: order_line_items.crop
 - Changeable: No
- 3. Duration Details:
 - Recurring Order ID:
 - Database Reference: order_line_items.recurring_order_id
 - Changeable: No
 - Rental Period:
 - Database Reference: order_line_items.rental_period
 - Changeable: No
 - Recurring Order Number List:
 - Database Reference: order_line_items.recurring_order_numbers
 - Changeable: No
- 4. Delivery Details:
 - Delivery ID:
 - Database Reference: delivery_details.delivery_id
 - Changeable: No
 - Delivery Status:
 - Database Reference: delivery_details.delivery_status
 - Changeable: No
- 5. Line Details:
 - Item No.:
 - Database Reference: order_line_items.item_no
 - Changeable: No
 - Item Name:
 - Database Reference: order_line_items.item_name
 - Changeable: No
 - Quantity:
 - Database Reference: order_line_items.quantity
 - Changeable: No
 - Per Unit Price:
 - Database Reference: order_line_items.unit_price
 - Changeable: No
 - Rent Amount:
 - Database Reference: order_line_items.rent_amount
 - Changeable: No
 - Rent Per Month:
 - Database Reference: order_line_items.rent_per_month
 - Changeable: No

Order Overview: O20240200018

Customer Details:						
Customer ID	Customer Name	Customer Email	Delivery Location			
1021	John Smith	johnsmith@acme.com	3876 Allen Rd, Bakersfield, CA 93314			
Internal Details:						
Account Manager	Link Netsuite	Region	Pick-up Location	Water Source	Crop	
Brawley Smith	brawley.smith	WEST	325 Rd 192, Delano, CA 93215	325 Rd 192, Delano, CA 93215	325 Rd 192, Delano, CA 93215	
Duration Details:						
Recurring Order <input type="checkbox"/>						
Rental Period			Recurring Order Number List			
<input type="text" value="July 24, 2024 - August 22, 2024"/>			O20240200008-1 O20240200008-2			
Line Details:						
Item No.	Item Name	Quantity	Per Unit Price	Rent Amount	Rent per month	
17727	10" Clamp	10	\$ 100.00	\$ 1000.00	July - \$ 500.00 August - \$ 500.00	

Create Order

Purpose:

The "Create Order" screen is designed to facilitate the creation of new orders by allowing users to input both header data (related to the customer and general order information) and line data (detailing the specific items and quantities involved in the order). This screen ensures that all necessary information is collected in a structured and efficient manner.

UI/UX Description:

General Layout:

- **Header Section:**
 - **Positioning:** Located at the top of the screen, divided into a grid layout to neatly display the customer-related information.
 - **Design:** Use clear and labeled fields with consistent spacing. Dropdowns and text fields should have consistent sizing for a clean look.
- **Line Data Section:**
 - **Positioning:** Below the header section, featuring a dynamic grid layout for entering multiple line items.
 - **Design:** Use a table format with the ability to add or remove rows. Each row should include a "Remove" button with a clear icon.
- **Recurring Order Section:**
 - **Positioning:** At the bottom of the screen, below the line items, with options appearing dynamically when "Recurring Order" is selected.
 - **Design:** Use conditional visibility to hide or show recurring options based on user selection.

Interactions:

- **Hover Effects:** Slight color change or shadow effect on input fields and buttons to indicate interactivity.
- **Validation:** Immediate visual feedback on errors (e.g., red border around required fields if left empty).

- **Responsive Design:** The layout adjusts for different screen sizes, with fields stacking vertically on smaller screens.

Technical Specification:

Header Data:

1. **Customer:**
 - **Type:** Single selection dropdown list.
 - **Data Source:** customer_list.name
 - **Changeable:** Yes
2. **Link Netsuite:**
 - **Type:** Free text
 - **Data Source:** Manually entered
 - **Changeable:** Yes
3. **Account Manager:**
 - **Type:** Single selection dropdown list
 - **Data Source:** account_managers.manager_name
 - **Changeable:** Yes
4. **Region:**
 - **Type:** Free text
 - **Data Source:** Manually entered
 - **Changeable:** Yes
5. **Customer Email:**
 - **Type:** Preloaded (based on selected customer)
 - **Data Source:** customer_list.email
 - **Changeable:** Yes
6. **Pick-up Location:**
 - **Type:** Single selection dropdown list
 - **Data Source:** warehouse_list.location_name | warehouse_list.address
 - **Changeable:** Yes
7. **Delivery Location:**
 - **Type:** Preloaded (based on selected customer)
 - **Data Source:** customer_list.delivery_location
 - **Changeable:** Yes
8. **Water Source:**
 - **Type:** Free text
 - **Data Source:** Manually entered
 - **Changeable:** Yes
9. **Crop:**
 - **Type:** Free text
 - **Data Source:** Manually entered
 - **Changeable:** Yes

Line Data:

1. **Item No.:**
 - **Type:** Single selection dropdown list
 - **Data Source:** warehouse_product_list.equipment_id
 - **Behavior:** Automatically populates the "Items" field upon selection.
 - **Changeable:** Yes
2. **Items:**
 - **Type:** Single selection dropdown list
 - **Data Source:** warehouse_product_list.equipment_name
 - **Behavior:** Automatically populates the "Item No." field upon selection.
 - **Changeable:** Yes
3. **Quantity:**
 - **Type:** Free text
 - **Data Source:** Manually entered
 - **Validation:** Must be a positive number
 - **Changeable:** Yes
4. **Price:**
 - **Type:** Free text
 - **Data Source:** Manually entered
 - **Validation:** Must be a valid currency format
 - **Changeable:** Yes

Recurring Order:

1. **Recurring Order Toggle:**
 - **Type:** Selection button
 - **Behavior:** If selected, the screen expands to show recurring options.
 - **Changeable:** Yes
2. **Rent Amount:**
 - **Type:** Calculation (Price * Quantity) – sum for all lines
 - **Data Source:** Automatically calculated based on "Price" and "Quantity"
 - **Changeable:** No
3. **Rent per Month:**
 - **Type:** Calculation
 - **Data Source:** Rent Amount / Rental Period (months)
 - **Display:** Listed per month, e.g., "August: \$5,000.00"
 - **Changeable:** No
4. **Rental Period:**
 - **Type:** Calendar selection
 - **Data Source:** User input
 - **Behavior:** Available only if "Recurring Order" is not selected
 - **Changeable:** Yes
5. **Repeats:**

- **Type:** Dropdown list (Weekly, Monthly, Yearly, etc.)
 - **Data Source:** User selection
 - **Behavior:** Available only if "Recurring Order" is selected
 - **Changeable:** Yes
6. **For:**
- **Type:** Free text
 - **Data Source:** User input (number of recurrences)
 - **Validation:** Must be a positive integer
 - **Behavior:** Available only if "Recurring Order" is selected
 - **Changeable:** Yes
7. **From/To:**
- **Type:** Calendar selection
 - **Data Source:** User input
 - **Behavior:** Available only if "Recurring Order" is selected
 - **Changeable:** Yes

Notes:

- The system will display a confirmation popup when the user attempts to create an order without filling in all mandatory fields
- All data are stored in the order_line_items table.

Pick Up Ticket

Purpose:

The "Pickup Ticket" screen is designed to manage and finalize the details of a delivery. It allows the user to set the pickup date, update the status of the delivery, and confirm the completion of the

delivery process. This screen ensures that each delivery is properly documented and the status is accurately reflected in the system.

UI/UX Description:

General Layout:

- **Header Section:**
 - **Positioning:** Located at the top of the screen, capturing overall delivery information such as pickup date and global status.
 - **Design:** The layout is intuitive, with dropdown menus and calendar options for easy data entry. The status dropdowns are prominently placed to ensure quick updates.
- **Line Data Section:**
 - **Positioning:** Below the header, displaying each item in the delivery with its current status.
 - **Design:** A simple table format where users can update the status of individual items. Editable fields are clearly distinguishable from non-editable ones.
- **Action Buttons:**
 - **Positioning:** At the bottom of the screen, including buttons for saving and proceeding to the next step.
 - **Design:** Clear and labeled action buttons. Upon clicking "Next," a confirmation dialog ensures that the user intends to finalize the delivery.

Interactions:

- **Hover Effects:** Fields and buttons will have slight color changes or shadows to indicate interactivity.
- **Validation:** Immediate visual feedback is provided on actions (e.g., status updates), ensuring that all necessary fields are completed before moving forward.
- **Responsive Design:** The screen layout adjusts for different device sizes, ensuring usability on both desktop and mobile devices.

Technical Specification:

Header Data:

1. **Pick Up Date:**
 - **Type:** Calendar selection
 - **Default:** Today's date
 - **Changeable:** Yes, through calendar input
2. **Global Status:**
 - **Type:** Single selection dropdown
 - **Options:** None, Loaded
 - **Behavior:** Changing this status updates the status of all line items displayed for the delivery.

Line Data:

1. Status:

- **Type:** Single selection dropdown
- **Options:** None, Loaded
- **Behavior:** Changing the status here updates only the specific line item selected.

Business Logic:

- **Delivery Confirmation:**
 - Upon clicking "Next" for the first time, the delivery_details table is updated to reflect the current state of the delivery.
 - Once the step is completed, no further updates are allowed on the table, and all fields that were previously editable become unchangeable.
 - **Confirmation Dialog:** A popup confirmation will appear when "Next" is clicked, confirming that the user is ready to complete the step.
- **Delivery Management:**
 - Each delivery must have a separate pick-up ticket. Although the current example only demonstrates one delivery, this logic applies to all deliveries.
 - Only one delivery can be managed per page to ensure clarity and prevent confusion.

Check Out

Purpose:

The "Check Out" screen is designed to finalize the details of a delivery as it is prepared to be sent out. This screen allows the user to confirm the checkout date, update the status of the delivery, and ensure that all quantities match the expected amounts before finalizing the process.

UI/UX Description:

General Layout:

- **Header Section:**
 - **Positioning:** Positioned at the top of the screen, this section includes the overall delivery information such as checkout date and global status.
 - **Design:** A clean, straightforward layout that prioritizes key information. Dropdowns and calendar inputs are easy to access and modify, and important warnings (like quantity mismatches) are prominently displayed.
- **Line Data Section:**
 - **Positioning:** Below the header, detailing each item within the delivery along with its quantity and status.
 - **Design:** Organized in a table format, allowing for clear visibility of each line item. Editable fields are highlighted, and non-editable fields are grayed out or disabled.
- **Action Buttons:**
 - **Positioning:** Located at the bottom of the screen, including options to save and proceed to the next step.
 - **Design:** Buttons are clearly labeled and easy to use. A confirmation dialog appears upon clicking "Next" to ensure the user is ready to complete the checkout.

Interactions:

- **Hover Effects:** Interactive elements (fields, buttons) have subtle color changes or shadows on hover.
- **Validation:** Immediate feedback is provided when quantities do not match, preventing the user from proceeding without addressing the issue.
- **Responsive Design:** The layout adapts to different screen sizes, maintaining usability on both desktop and mobile devices.

Technical Specification:

Header Data:

1. **Check Out Date:**
 - **Type:** Calendar selection
 - **Default:** Today's date
 - **Changeable:** Yes, through calendar input
2. **Global Status:**
 - **Type:** Single selection dropdown
 - **Options:** None, Check Out
 - **Behavior:** Changing this status updates the status of all line items displayed for the delivery.

Line Data:

1. **Status:**
 - **Type:** Single selection dropdown
 - **Options:** None, Check Out
 - **Behavior:** Changing the status here updates only the specific line item selected.
2. **Quantity:**
 - **Type:** Free text
 - **Behavior:** If the quantity entered does not match the expected quantity (from the delivery), a warning is displayed to the user. The user cannot proceed until the discrepancy is resolved.

Business Logic:

- **Delivery Confirmation:**
 - Upon clicking "Next" for the first time, the delivery_details table is updated to reflect the current state of the delivery.
 - Once this step is completed, no further updates are allowed on the table, and all fields that were previously editable become unchangeable.
 - **Confirmation Dialog:** A popup confirmation will appear when "Next" is clicked, confirming that the user is ready to finalize the checkout.
- **Delivery Management:**
 - Each delivery must have a separate check-out ticket. Although the current example only demonstrates one delivery, this logic applies to all deliveries.
 - Only one delivery can be managed per page to ensure clarity and prevent confusion.

The screenshot displays a delivery management interface with a progress bar at the top indicating eight steps: 1. Create Delivery, 2. Pickup Ticket, 3. Check Out, 4. Delivered to customer, 5. Create Return Deliveries, 6. Pickup Ticket, 7. Check In, and 8. Inspection completed. The 'Check Out' step is currently active.

Below the progress bar, there is a 'Done' button and a 'GLOBAL STATUS' dropdown menu set to 'None'.

The main section contains a table with delivery details:

Delivery ID	Total Quantity	CheckOut Date	Delivery Date
0202HQ00008-1	28	08/28/2024	3 Apr, 2024

Below this table is an 'Items information' table:

Sr No	Internal Id	Items	Quantity	Status
1	2035	Rental 10" Plain Coupler	14	None
2	2034	Rental 10" Tapped Coupler		None

A warning message is displayed below the 'Quantity' column for item 2: "Quantity does not match, Please Recount".

Below the items table, there is an 'ATTACH DOCUMENTS' section with a text input field for 'Add a comment'.

At the bottom, there is a 'SHOW 10 ENTRIES' section with a table of documents:

DOCUMENTS	CREATED AT	COMMENTS
File 1	20-05-2024	Farming pumps and pipes are indispensable tools, ensuring efficient irrigation and water management. They're the lifeline of agriculture, delivering sustenance to crops, fostering growth, and... Read More

The interface also includes a 'Showing 1 to 1 of 1 entries' message and 'Previous' and 'Next' buttons.

Delivered to Customer

Purpose:

The "Delivered to Customer" screen finalizes the delivery process by confirming the delivery date and updating the status of the delivery. This step ensures that the delivery details are locked in, marking the items as delivered to the customer.

UI/UX Description:

General Layout:

- **Header Section:**
 - **Positioning:** Located at the top, displaying crucial information like the delivery date and global status.
 - **Design:** A clean, user-friendly interface that makes it easy for users to input or change details. Calendar inputs and dropdowns are easy to navigate and interact with.
- **Line Data Section:**
 - **Positioning:** Below the header, this section lists each item included in the delivery along with its quantity and status.
 - **Design:** Presented in a table format for clarity. Editable fields are prominently displayed, while fields that become unchangeable after confirmation are visually distinct.
- **Action Buttons:**
 - **Positioning:** At the bottom, the "Next" button finalizes the delivery.
 - **Design:** Buttons are designed with clarity in mind. A confirmation dialog appears when the user attempts to finalize the delivery, preventing accidental completion.

Interactions:

- **Hover Effects:** Subtle color changes on interactive elements like fields and buttons when hovered over.
- **Validation:** Immediate feedback if there are any mismatches or issues with the data entered.
- **Responsive Design:** The layout adjusts smoothly to different screen sizes, ensuring usability on both desktop and mobile devices.

Technical Specification:

Header Data:

1. **Delivery Date:**
 - **Type:** Calendar selection
 - **Default:** Today's date
 - **Changeable:** Yes, through calendar input
2. **Global Status:**
 - **Type:** Single selection dropdown
 - **Options:** None, Delivered

- **Behavior:** Changing this status updates the status of all line items displayed for the delivery.

Line Data:

1. **Quantity:**
 - o **Type:** Free Text
 - o **Behavior:** Prefilled from previous stages. If the quantity entered does not match the expected quantity (from the delivery), a warning is displayed to the user. The user can proceed with the discrepancy in quantity.
2. **Status:**
 - o **Type:** Single selection dropdown
 - o **Options:** None, Delivered
 - o **Behavior:** Changing the status here updates only the specific line item selected.

Business Logic:

- **Delivery Confirmation:**
 - Upon clicking "Next" for the first time, the delivery_details table is updated to reflect the current state of the delivery.
 - Once this step is completed, no further updates are allowed on the table, and all previously editable fields become unchangeable.
 - **Confirmation Dialog:** A popup confirmation will appear when "Next" is clicked, ensuring the user is ready to finalize the delivery.
- **Delivery Management:**
 - Each delivery has a separate "Delivered to Customer" entry. While this example demonstrates the process for one delivery, the same logic applies to all deliveries.
 - Only one delivery is managed per page to maintain clarity and prevent errors.

The screenshot displays a delivery management interface with a progress bar at the top showing 8 steps: 1. Create Delivery, 2. Pickup Ticket, 3. Check Out, 4. Delivered to customer (current step), 5. Create Return Deliveries, 6. Pickup Ticket, 7. Check In, and 8. Inspection completed. Below the progress bar is a 'Done' button and a 'GLOBAL STATUS' dropdown set to 'Delivered'.

The main section contains a table with delivery details:

Delivery ID	Total Quantity	Contract Start Date	Delivery Date
Q202402000284	28	2 Apr, 2024	08/28/2024

Below this is a table for 'Items Information':

Sr No	Internal Id	Items	Quantity	Status
1	20335	Rental 10" Plain Coupler	14	Delivered
2	20334	Rental 10" Tapped Coupler	14	Delivered

Below the items table is an 'ATTACH DOCUMENTS' section with a text area for 'Add a comment:'.

At the bottom, there is a 'SHOW 10 ENTRIES' section with a table for documents and comments:

DOCUMENTS	CREATED AT	COMMENTS
File	20-05-2024	Farming pumps and pipes are indispensable tools, ensuring efficient irrigation and water management. They're the lifeline of agriculture, delivering sustenance to crops, fostering growth, and... Read More

Navigation buttons include 'Previous', 'Next', and 'Done'.

Return Pick Up Ticket

Purpose:

The "PickUp Ticket" screen facilitates the scheduling and confirmation of pickup dates for deliveries. This screen ensures that all relevant details regarding the pickup process are captured, and the status of each item in the delivery is appropriately managed.

UI/UX Description:

General Layout:

- **Header Section:**
 - **Positioning:** The top of the screen, displaying the pickup date, global status, and removal date.
 - **Design:** User-friendly interface with calendar inputs and dropdowns for easy interaction. The layout is clean, with fields clearly separated to reduce confusion.
- **Line Data Section:**
 - **Positioning:** Below the header, listing each item in the delivery along with its quantity and status.
 - **Design:** Presented in a structured table format, ensuring clarity and ease of use. Editable fields are highlighted, while fields that become unchangeable after confirmation are visually distinct.
- **Action Buttons:**
 - **Positioning:** The "Next" button is located at the bottom right, providing a clear call to action to finalize the pickup details.
 - **Design:** Designed for ease of access, the button triggers a confirmation popup before finalizing the process.

Interactions:

- **Hover Effects:** Subtle color changes on interactive elements like fields and buttons when hovered over.
- **Validation:** Immediate feedback if there are any mismatches or issues with the data entered.
- **Responsive Design:** The layout adjusts smoothly to different screen sizes, ensuring usability on both desktop and mobile devices.

Technical Specification:

Header Data:

1. **PickUp Date:**
 - **Type:** Calendar selection
 - **Default:** Today's date
 - **Changeable:** Yes, through calendar input
2. **Global Status:**
 - **Type:** Single selection dropdown

- **Options:** None, Loaded
 - **Behavior:** Changing this status updates the status of all line items displayed for the delivery.
3. **Removal Date:**
- **Type:** Calendar selection
 - **Changeable:** Yes, through calendar input

Line Data:

1. **Status:**
- **Type:** Single selection dropdown
 - **Options:** None, Loaded
 - **Behavior:** Changing the status here updates only the specific line item selected.

Business Logic:

- **Pickup Confirmation:**
 - Upon clicking "Next" for the first time, the delivery_details table is updated to reflect the current state of the delivery.
 - Once this step is completed, no further updates are allowed on the table, and all previously editable fields become unchangeable.
 - **Confirmation Dialog:** A popup confirmation will appear when "Next" is clicked, ensuring the user is ready to finalize the pickup details.
- **Delivery Management:**
 - Each delivery has a separate "PickUp Ticket." While this example demonstrates the process for one delivery, the same logic applies to all deliveries.
 - Only one delivery is managed per page to maintain clarity and prevent errors.

The screenshot displays a delivery management interface. At the top, a progress bar shows eight steps: 1. Create Delivery, 2. Pickup Ticket, 3. Check out, 4. Delivered to customer, 5. Create Return Deliveries, 6. Pickup Ticket, 7. Check in, and 8. Inspection completed. Step 2 is currently active.

Below the progress bar, there is a "Pin" button and a "GLOBAL STATUS" dropdown menu set to "None".

The main section contains a table with two delivery entries:

Delivery ID	Total Quantity	Pickup Date	Removal Date															
OR20240200098-1	28	08/28/2024	2 Apr. 2024															
<table border="1"> <thead> <tr> <th>Sr No</th> <th>Internal Id</th> <th>Items</th> <th>Quantity</th> <th>Status</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>20335</td> <td>Rental 10" Plain Coupler</td> <td>14</td> <td>None</td> </tr> <tr> <td>2</td> <td>20374</td> <td>Rental 10" Tapped Coupler</td> <td>14</td> <td>None</td> </tr> </tbody> </table>				Sr No	Internal Id	Items	Quantity	Status	1	20335	Rental 10" Plain Coupler	14	None	2	20374	Rental 10" Tapped Coupler	14	None
Sr No	Internal Id	Items	Quantity	Status														
1	20335	Rental 10" Plain Coupler	14	None														
2	20374	Rental 10" Tapped Coupler	14	None														
OR20240200098-2	28	08/28/2024	3 Apr. 2024															
<table border="1"> <thead> <tr> <th>Sr No</th> <th>Internal Id</th> <th>Items</th> <th>Quantity</th> <th>Status</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>20333</td> <td>Rental 10" Tapped Coupler</td> <td>28</td> <td>None</td> </tr> </tbody> </table>				Sr No	Internal Id	Items	Quantity	Status	1	20333	Rental 10" Tapped Coupler	28	None					
Sr No	Internal Id	Items	Quantity	Status														
1	20333	Rental 10" Tapped Coupler	28	None														

Below the table, there is an "ATTACH DOCUMENTS" section with a text input field for "Add a comment".

At the bottom, there is a "SHOW" dropdown set to "ENTRIES", a "DOCUMENTS" section with a "File" button, a "CREATED AT" field showing "20-05-2024", and a "COMMENTS" section with a search bar and a text area containing the text: "Farming pumps and pipes are indispensable tools, ensuring efficient irrigation and water management. They're the lifeline of agriculture, delivering sustenance to crops, fostering growth, and... [Read More](#)".

At the bottom right, there are "Previous" and "Next" buttons.

Check In

Purpose:

The "Check In" screen is designed to allow users to confirm the receipt of goods or items. This step ensures that all delivered items are accounted for and any discrepancies in quantities are addressed before finalizing the delivery process.

UI/UX Description:

General Layout:

- **Header Section:**
 - **Positioning:** Located at the top of the screen, displaying the inspection date and global status.
 - **Design:** Clean and intuitive layout with easily accessible calendar and dropdown options for quick data entry. The design is streamlined for efficiency, minimizing the number of clicks needed to complete the process.
- **Line Data Section:**
 - **Positioning:** Below the header, presenting each item in the delivery along with its quantity and status.
 - **Design:** The table format is used for clarity, with warnings and validation messages appearing inline to alert users of any issues. Editable fields are distinctly highlighted.
- **Action Buttons:**
 - **Positioning:** The "Next" button is at the bottom right, guiding the user to complete the check-in process.
 - **Design:** The button is prominently displayed and triggers a confirmation popup before finalizing the step, ensuring the user is ready to proceed.

Interactions:

- **Hover Effects:** Subtle highlights on interactive elements to indicate they are clickable.
- **Validation:** Instant feedback provided if the quantity does not match, as demonstrated in the screenshot.
- **Responsive Design:** The screen layout adjusts gracefully to different screen sizes, ensuring usability on both large and small devices.

Technical Specification:

Header Data:

1. **Inspection Date:**
 - **Type:** Calendar selection
 - **Default:** Today's date
 - **Changeable:** Yes, through calendar input
2. **Global Status:**
 - **Type:** Single selection dropdown

- **Options:** None, Check In
- **Behavior:** Changing this status updates the status of all line items displayed for the delivery.

Line Data:

1. **Status:**
 - **Type:** Single selection dropdown
 - **Options:** None, Check In
 - **Behavior:** Changing the status here updates only the specific line item selected.
2. **Quantity:**
 - **Type:** Free text
 - **Validation:** If the quantity does not match the delivery quantity, a warning is displayed to the user.
 - **Behavior:** Allows the user to manually enter the quantity checked in.

Business Logic:

- **Check-In Confirmation:**
 - When "Next" is clicked for the first time, the delivery_details table is updated to reflect the inspection's outcome.
 - After this step is confirmed, no further updates are allowed, and all previously editable fields become unchangeable.
 - **Confirmation Dialog:** A popup will confirm the user's intent to finalize the check-in, preventing accidental completion.
- **Delivery Management:**
 - Each delivery is associated with a separate "Check In" record. This documentation is focused on one delivery for demonstration purposes, but the logic applies universally across all deliveries.
 - Only one delivery is handled per page to maintain simplicity and prevent errors.

Inspection

Purpose:

The "Inspection Completed" screen finalizes the delivery process by allowing the user to verify and confirm the inspection of all delivered items. This step ensures that all items have been inspected, and their statuses are correctly recorded before closing the delivery process.

UI/UX Description:

General Layout:

- **Header Section:**
 - **Positioning:** Top of the screen, displaying the check-in date and global status.
 - **Design:** Clear and intuitive layout with a calendar option for date input and dropdowns for status selection. The design is streamlined to guide the user effortlessly through the inspection confirmation process.
- **Line Data Section:**
 - **Positioning:** Below the header, showing each item included in the delivery, along with its quantity and inspection status.
 - **Design:** The table format is utilized for clarity, with warnings and validation messages displayed inline to alert users of any issues. Editable fields are highlighted to distinguish them from non-editable content.
- **Action Buttons:**
 - **Positioning:** The "Submit" button is at the bottom right, prompting the user to complete the inspection process.
 - **Design:** The button is prominently displayed and triggers a confirmation popup before finalizing the inspection, ensuring the user is ready to proceed.

Interactions:

- **Hover Effects:** Subtle highlights on interactive elements to indicate they are clickable.

- **Validation:** Instant feedback provided if the quantity does not match the quantity listed in the delivery, as demonstrated in the screenshot.
- **Responsive Design:** The screen layout adjusts gracefully to different screen sizes, ensuring usability on both large and small devices.

Technical Specification:

Header Data:

1. **Inspection Date:**
 - **Type:** Calendar selection
 - **Default:** Today's date
 - **Changeable:** Yes, through calendar input
2. **Global Status:**
 - **Type:** Single selection dropdown
 - **Options:** None, Loaded
 - **Behavior:** Changing this status updates the status of all line items displayed for the delivery.

Line Data:

1. **Status:**
 - **Type:** Single selection dropdown
 - **Options:** None, Loaded
 - **Behavior:** Changing the status here updates only the specific line item selected.
2. **Quantity:**
 - **Type:** Free text
 - **Validation:** If the quantity does not match the delivery quantity, a warning is displayed to the user.
 - **Behavior:** Allows the user to manually enter the quantity inspected.

Business Logic:

- **Inspection Confirmation:**
 - When "Submit" is clicked for the first time, the delivery_details table is updated to reflect the inspection's outcome.
 - After this step is confirmed, no further updates are allowed, and all previously editable fields become unchangeable.
 - **Confirmation Dialog:** A popup will confirm the user's intent to finalize the inspection, preventing accidental completion.
- **Delivery Management:**
 - Each delivery is associated with a separate inspection record. This documentation is focused on one delivery for demonstration purposes, but the logic applies universally across all deliveries.
 - Only one delivery is handled per page to maintain simplicity and prevent errors.

Stock Adjustment (valid for Check In and Inspection)

Purpose:

The "Stock Adjustment" screen is used during the Check-in and Inspection stages when there is a discrepancy between the delivered quantity and the quantity recorded in the system. This screen allows the user to adjust the stock levels accordingly before finalizing the check-in or inspection process.

UI/UX Description:

General Layout:

- **Header Section:**
 - **Positioning:** Top of the screen, displaying the item number and the adjustment reference.
 - **Design:** Clean and concise layout, with the focus on key input fields required for the stock adjustment. The header clearly identifies the item and the adjustment being made.
- **Input Fields Section:**
 - **Positioning:** Centered below the header, containing fields for item number, location, date, current quantity, revised quantity, plus/minus quantity, reason, and comment.
 - **Design:** Each input field is clearly labeled and aligned for easy data entry. The date field includes a calendar option for selection, and dropdowns are used for selecting reasons and locations to minimize user error.
- **Action Buttons:**
 - **Positioning:** Bottom right, with "Cancel" and "Save" buttons.
 - **Design:** The "Save" button is highlighted in green to indicate the primary action, while the "Cancel" button is red to allow the user to discard changes easily.

Interactions:

- **Hover Effects:** Input fields and buttons highlight slightly when hovered over to indicate interactivity.
- **Validation:** Automatic calculation and validation for fields like Revised Quantity, Plus Quantity, and Minus Quantity based on user input and system data.
- **Responsive Design:** The layout adjusts for different screen sizes, ensuring usability on both desktop and mobile devices.

Technical Specification:

Fields and Data Handling:

1. **Item Number:**
 - **Type:** Preloaded, non-editable
 - **Source:** Loaded from `delivery_details.item_number`
 - **Changeable:** No
2. **Location:**
 - **Type:** Preloaded, non-editable
 - **Source:** Loaded from `delivery_details.pickup_location`
 - **Changeable:** No
3. **Date:**
 - **Type:** Calendar selection
 - **Default:** Today's date
 - **Changeable:** Yes, through calendar input
4. **Current Quantity:**
 - **Type:** Preloaded, non-editable
 - **Source:** Loaded from the stock adjustment screen as current quantity to show the current warehouse stock
5. **Revised Quantity:**
 - **Type:** Auto-calculated
 - **Formula:** Current Quantity $-/+$ (Plus Quantity or Minus Quantity)
 - **Behavior:** Reflects the updated stock level after adjustment
 - **Changeable:** No
6. **Plus Quantity:**
 - **Type:** Auto-calculated, pre-filled
 - **Source:** Based on user input during the Check-in or Inspection stage, reflects any addition to stock
7. **Minus Quantity:**
 - **Type:** Auto-calculated, pre-filled
 - **Source:** Based on user input during the Check-in or Inspection stage, reflects any deduction from stock
8. **Reason:**
 - **Type:** Single selection dropdown
 - **Options:** Set automatically based on the stage (e.g., "Check-in" or "Inspection")

9. Comment:

- **Type:** Auto-populated, non-editable
- **Source:** Set as the delivery number for reference

Business Logic:

- **Stock Adjustment Process:**
 - **Trigger:** This screen is accessed during the Check-in or Inspection process when a quantity discrepancy is detected.
 - **Data Update:** Upon completion and saving of the stock adjustment, the system updates the stock_levels table with the revised quantity.
 - **Finalization:** After the stock adjustment is confirmed, the user can finalize the Check-in or Inspection process with the updated quantity for the specific delivery.
- **Validation and Confirmation:**
 - **Reconciliation:** The system ensures the reconciliation on the quantity for specific rental stages reconciled with the stock adjustment.
 - **Validation:** The system ensures that all required fields are filled and calculations are correct before allowing the user to save the adjustment.
 - **Confirmation Dialog:** A popup window asks the user to confirm the stock adjustment before saving, preventing accidental changes.

PROJ016968 R1/2 : Stock Adjustment

ITEM NUMBER

20315

LOCATION

Colusa

DATE

08/29/2024

CURRENT QUANTITY

1593

REVISED QUANTITY

1589

PLUS QUANTITY

0

MINUS QUANTITY

4

REASON

Inspection

COMMENT

PROJ016968 R1/2

Cancel

Save

Stock Adjustment

Stock Adjustment

Equipment ID Equipment Material Group Equipment Name Location

[Download Stock Adjustment Report CSV](#) [Mass Stock Adjustment](#)

Show 10 entries

EQUIPMENT ID	EQUIPMENT MATERIAL GROUP	EQUIPMENT NAME	QUANTITY	LOCATION	ACTION
17727	001	Rental 12" X 40' Aluminum Mainline		1593 Colata	Edit
17728	002	20" X 30' Steel Reinforced Pipe		2,467 Los Angeles	Edit
17729	003	8" X 20' PVC Pipe		875 San Diego	Edit
17730	004	16" X 50' Galvanized Steel Pipe		1,234 New York	Edit
17731	005	6" X 10' Copper Pipe		345 Chicago	Edit
17732	006	4" X 25' Fiber Optic Cable		987 Houston	Edit
17733	007	10" X 15' Reinforced Concrete Pipe		762 Philadelphia	Edit
17734	008	18" X 40' PVC Drainage Pipe		1122 San Jose	Edit
17735	009	24" X 10' Ductile Iron Pipe		542 San Diego	Edit
17736	010	12" X 60' High-Density Polyethylene Pipe		1546 Miami	Edit

Showing 1 to 10 of 10 entries

[Previous](#) [Next](#)

1. Purpose

The Stock Adjustment screen is an integral part of the Laurel Rental Application. It is designed to provide real-time visibility into the current inventory levels across various warehouse locations. Users can make adjustments to the stock quantities for different pieces of equipment either individually or in bulk. The screen interacts closely with the stock_adjustment database table, ensuring all changes are logged and immediately reflected in the interface.

2. User Interface (UI) Components

1.1 Filters and Search Panel

- *Equipment ID Filter:*
 - **Type:** Text Input
 - **Functionality:** Filters the displayed records by the specific equipment_id.
 - **Validation:** Must match an equipment_id existing in the warehouse_product_list table.
- *Equipment Material Group Filter:*
 - **Type:** Text Input
 - **Functionality:** Filters records by the equipment_material_group, allowing users to focus on specific categories.
 - **Validation:** Must match valid material group categories present in the system.
- *Equipment Name Filter:*
 - **Type:** Text Input
 - **Functionality:** Filters records by the name or description of the equipment.
 - **Validation:** Must match names as listed in the warehouse_product_list.

- *Location Filter:*
 - **Type:** Text Input
 - **Functionality:** Filters the displayed records by the warehouse or location name.
 - **Validation:** Must match a valid location name from the warehouse_list table.
- *Search Box:*
 - **Type:** Global Search Input
 - **Functionality:** Provides a keyword search across all displayed fields, enabling quick access to specific records.

1.2 Data Table Display

- *Equipment ID Column:*
 - **Type:** Numeric Display
 - **Data Source:** equipment_id from the stock_adjustment table.
 - **Functionality:** Displays the unique identifier for each piece of equipment.
- *Equipment Material Group Column:*
 - **Type:** Text Display
 - **Data Source:** equipment_material_group from the stock_adjustment table.
 - **Functionality:** Displays the material group classification for the equipment.
- *Equipment Name Column:*
 - **Type:** Text Display
 - **Data Source:** equipment_name from the stock_adjustment table.
 - **Functionality:** Displays the name or description of the equipment.
- *Quantity Column:*
 - **Type:** Numeric Display
 - **Data Source:** new_quantity from the most recent entry in the stock_adjustment table.
 - **Functionality:** Displays the current quantity of the equipment at the specified location.
- *Location Column:*
 - **Type:** Text Display
 - **Data Source:** location from the stock_adjustment table.
 - **Functionality:** Displays the location where the equipment is stored.

- *Action Column:*
 - **Type:** Action Buttons
 - **Components:**
 - **Edit Button:**
 - **Icon:** Pencil/Edit icon
 - **Functionality:** Opens a modal or inline editor for adjusting the quantity of the selected equipment at the specified location.
 - **Database Interaction:** On submission, updates the stock_adjustment table and refreshes the displayed data.

1.3 Action Buttons

- *Mass Stock Adjustment Button:*
 - **Type:** Action Button
 - **Functionality:** Opens a bulk adjustment interface for modifying quantities of multiple equipment items across locations.
 - **Database Interaction:** Submits bulk changes to the stock_adjustment table and refreshes the screen upon completion.
- *Stock Adjustment Report CSV Button:*
 - **Type:** Action Button
 - **Functionality:** Exports the current view of the stock adjustment data to a CSV file for offline use.
 - **Data Source:** Pulls from the currently displayed data in the table.

3. Database Interactions

3.1 Stock Adjustment Table

- **Table Name:** stock_adjustment
- **Primary Keys:**
 - adjustment_id - Auto-incremented integer unique to each stock adjustment record.
- **Foreign Keys:**
 - equipment_id - References the equipment_id in the warehouse_product_list table.
 - changed_done_by - References the user_id in the user_list table.
- **Triggers:**
 - **On Update:** Updates the new_quantity and changed_done_by fields.
 - **On Insert:** Inserts a new record for each adjustment, ensuring.

4. Business Logic

4.1 Real-Time Updates

- **Logic:** Upon any update (single or mass), the screen fetches the latest data from the stock_adjustment table and refreshes the displayed information. This ensures that users always see the most current data.
- **Concurrency Control:** To avoid race conditions, updates are processed in sequence, with each transaction locking the relevant rows in the database until the operation is complete.

4.2 Validation and Error Handling

- *Validation:*
 - Equipment and location validations are performed both on the client-side and server-side to ensure data integrity.
 - Quantity fields are validated to ensure they contain positive integers, and the reasons for adjustments are verified against a list of predefined reasons.
- *Error Handling:*
 - User-friendly error messages are displayed in case of validation failures or system errors during the adjustment process.

5. Security Considerations

5.1 Access Control

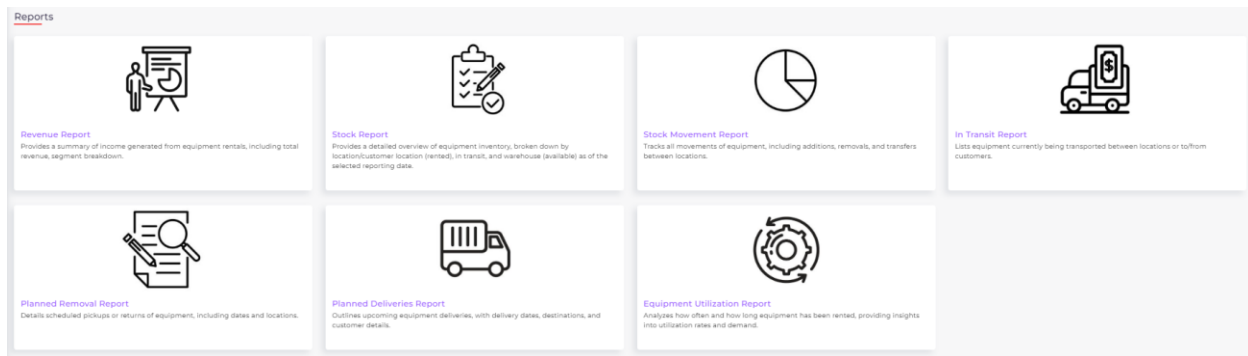
- *Role-Based Access:* Only users with specific have access to the Stock Adjustment screen and its functionalities.
- *Authorization:* Each API request checks the user's role and permissions before allowing any adjustments or data retrieval.

6. Performance Considerations

6.1 Optimization Techniques

- *Indexing:* Indexes are created on frequently queried columns like equipment_id, location to speed up data retrieval.
- *Pagination:* Data is loaded in paginated views (e.g., 10 entries per page) to reduce the load time and improve the user experience, especially when dealing with large datasets.
- *Caching:* Recent queries may be cached to reduce the load on the database, particularly when users are repeatedly filtering similar data.

Reports



1. Revenue Report

Overview:

The Revenue Report provides a detailed summary of income generated from equipment rentals, including total revenue segmented by customer, equipment, and rental duration. Users can filter the report by revenue period, customer name, and equipment material group to gain insights into the financial performance of their rental operations. This report is essential for tracking revenue streams, understanding customer contributions, and optimizing rental strategies.

Key Features and Business Logic:

1. Revenue Calculation:

- The report summarizes the total revenue generated from equipment rentals within the specified period. It accurately reflects the income for each month, ensuring that revenue is split correctly even when contracts span multiple months.
- **Business Logic:**
 - If the rental period (contract start and end dates) spans multiple months, the revenue is distributed evenly across the months. For example, if an order has a total value of \$10,000 for August to September, the report will show \$5,000 for each month when filtered separately.
 - The report calculates revenue based on the number of units rented, the price per unit, and the rental duration within the specified revenue period.

2. Segment Breakdown:

- The report provides a breakdown of revenue by customer, equipment material group, equipment name, and rental duration, helping to identify key revenue drivers.
- **Business Logic:**
 - Segmenting revenue by these categories allows users to pinpoint which customers and equipment contribute most to the overall revenue, enabling more informed business decisions.
 - The rental duration is displayed in months, giving a clear picture of how long each equipment unit was rented, further aiding in assessing equipment utilization and profitability.

3. Customer Insights:

- The report highlights the contribution of each customer to the total revenue, supporting targeted customer management.
- **Business Logic:**
 - By analyzing the revenue generated by each customer, businesses can focus on retaining high-value customers and improving relationships with those who contribute less.
 - The account manager responsible for each customer is also shown, which helps in evaluating the effectiveness of account management strategies.

Columns and Data Sources:

1. **customer_name** - VARCHAR(255)
 - **Description:** The name of the customer associated with the order.
 - **Source:**
 - **Primary:** order_table.customer_id
 - **Referenced Table:** customer_list
 - **Reference Condition:** order_table.customer_id = customer_list.customer_id
 - **Purpose:** Identifies the customer contributing to the revenue.
2. **order_number** - VARCHAR(255)
 - **Description:** The unique identifier for each order.
 - **Source:** order_table.order_id
 - **Purpose:** Provides a reference to the specific order generating the revenue.
3. **equipment_parent_group** - VARCHAR(255)
 - **Description:** The parent group or material group of the equipment.
 - **Source:**
 - **Primary:** order_table.equipment_id
 - **Referenced Table:** warehouse_product_list
 - **Reference Condition:** order_table.equipment_id = warehouse_product_list.equipment_id
 - **Purpose:** Helps categorize the equipment into broader material groups for analysis.
4. **equipment_id** - INT
 - **Description:** The unique identifier for the equipment rented under the order.
 - **Source:**
 - **Primary:** order_table.equipment_id
 - **Referenced Table:** warehouse_product_list
 - **Reference Condition:** order_table.equipment_id = warehouse_product_list.equipment_id
 - **Source:** order_table.equipment_id
 - **Purpose:** Provides a unique reference to the specific equipment unit generating revenue.
5. **equipment_name** - VARCHAR(255)
 - **Description:** The name or description of the equipment rented.
 - **Source:**
 - **Primary:** order_table.equipment_id

- **Referenced Table:** warehouse_product_list
 - **Reference Condition:** order_table.equipment_id = warehouse_product_list.equipment_id
- **Purpose:** Allows users to see exactly which equipment was rented out, aiding in understanding equipment demand.
- 6. **quantity** - INT
 - **Description:** The number of equipment units rented under the order.
 - **Source:** order_table.quantity
 - **Purpose:** Displays the amount of equipment rented, which is used in revenue calculation.
- 7. **price** - DECIMAL(10, 2)
 - **Description:** The price per unit of equipment rented.
 - **Source:** order_table.unit_price
 - **Purpose:** Essential for calculating total revenue from the rental.
- 8. **revenue** - DECIMAL(10, 2)
 - **Description:** The total revenue generated from the rental for the selected period.
 - **Source:**
 - **Primary:** order_table.quantity, order_table.unit_price
 - **Calculated Field:** Revenue is calculated based on the quantity, price, and duration within the selected revenue period.
 - **Calculation Logic:**
 - The revenue is calculated as quantity * unit_price.
 - If the selected revenue period only covers part of the rental period, the revenue is split proportionally by months.
 - **Example:**
 - If a contract is active from August 15, 2024, to September 28, 2024, and the total revenue is \$10,000, filtering the report for August would show \$5,000.
 - **Purpose:** Displays the revenue contribution of each order within the selected period.
- 9. **rental_duration** - VARCHAR(50)
 - **Description:** The duration of the rental, expressed in days.
 - **Source:** Calculated based on order_table.contract_start_date and order_table.contract_end_date.
 - **Calculation Logic:**
 - The duration is calculated as the difference in days between the start and end dates of the contract.
 - **Purpose:** Provides insight into the length of rental agreements, which can be crucial for understanding equipment usage patterns.
- 10. **account_manager** - VARCHAR(255)
 - **Description:** The account manager responsible for the order.
 - **Source:**
 - **Primary:** order_table.account_manager_id
 - **Referenced Table:** account_managers

- **Reference Condition:** `order_table.account_manager_id = account_managers.account_manager_id`
- **Purpose:** Identifies the account manager in charge of the customer, linking performance to personnel management.

Business Logic and Special Considerations:

- **Revenue Period Splitting:**
 - The revenue period filter splits the revenue according to the duration of the contract within the selected months. If an order spans multiple months, the revenue is divided equally among those months.
 - **Example:**
 - For an order with a total revenue of \$10,000 spanning August and September, the report will show \$5,000 for each month when filtered individually.
- **Data Integrity:**
 - Ensure that all referenced foreign keys (e.g., `customer_id`, `equipment_id`, `account_manager_id`) are valid and exist in their respective tables to avoid inconsistencies.
- **Export Functionality:**
 - The report can be exported as a CSV file, maintaining the filtered data and calculations as displayed on the screen.

SQL Query Example:

```
SELECT
  c.name AS customer_name, -- customer_list.name
  o.order_id AS order_number, -- order_table.order_id
  w.equipment_material_group AS equipment_parent_group, -- warehouse_product_list.equipment_material_group
  o.equipment_id, -- order_table.equipment_id
  w.equipment_name, -- warehouse_product_list.equipment_name
  o.quantity, -- order_table.quantity
  o.unit_price AS price, -- order_table.unit_price
  CASE
    WHEN MONTH(o.contract_start_date) = MONTH('2024-08-01')
      AND MONTH(o.contract_end_date) = MONTH('2024-08-31') THEN o.unit_price * o.quantity
    WHEN MONTH(o.contract_start_date) = MONTH('2024-08-01') THEN
      (o.unit_price * o.quantity / DATEDIFF(o.contract_end_date, o.contract_start_date)) * DATEDIFF('2024-08-31',
o.contract_start_date)
    WHEN MONTH(o.contract_end_date) = MONTH('2024-08-31') THEN
      (o.unit_price * o.quantity / DATEDIFF(o.contract_end_date, o.contract_start_date)) * DATEDIFF(o.contract_end_date,
'2024-08-01')
    ELSE
      o.unit_price * o.quantity / TIMESTAMPDIFF(MONTH, o.contract_start_date, o.contract_end_date)
  END AS revenue, -- Calculated field based on order_table
  CONCAT(TIMESTAMPDIFF(MONTH, o.contract_start_date, o.contract_end_date), ' Month(s)') AS rental_duration, --
Calculated from order_table
  a.name AS account_manager -- account_managers.name
FROM
  order_table o
```

```

JOIN
    customer_list c ON o.customer_id = c.customer_id
JOIN
    warehouse_product_list w ON o.equipment_id = w.equipment_id
JOIN
    account_managers a ON o.account_manager_id = a.account_manager_id
WHERE
    o.contract_start_date <= '2024-08-31' AND o.contract_end_date >= '2024-08-01';

```

Use Case Example:

- **Scenario:** A user wants to analyze the revenue generated in August 2024.
 - **Steps:**
 1. The user selects the revenue period as August 2024.
 2. The system retrieves orders with a contract start date before August 31, 2024, and an end date after August 1, 2024.
 3. The system calculates the revenue for each order within the specified period.
 4. The report is generated, showing each customer's contribution, the equipment involved, and the duration of the rental.

Table Example:

Filters

Users can select multiple options, and as they type the name of a customer or equipment, it will be displayed as a selectable option in a dropdown list.

- **Revenue Period:** Calendar filter option with ability to select date or date range.
- **Customer Name:** Dropdown with multi-select options. As the user types, available locations will be suggested.
- **Parent Equipment Group:** Dropdown with multi-select options. As the user types, available locations will be suggested.

The **Revenue Period** filter must be selected for the report to populate.

Date Fields

- **Revenue Period:** The period during which the revenue was generated.
- **Customer Name:** Name of the customer associated with the rental.
- **Order Number:** Unique identifier for the rental order.
- **Parent Equipment Group:**
- **Equipment ID:** Unique identifier for the equipment.
- **Equipment Name:** Descriptive name of the equipment.
- **Quantity:** Number of units rented.
- **Price:** Price of unit rented.
- **Revenue:** Total revenue generated from the rental.
- **Rental Duration:** The duration for which the equipment was rented.
- **Account Manager:** The account manager responsible for overseeing the rental transactions

Filters: Revenue Period Filters: Customer Filters: Equipment Material Group

Calendar Selection Dropdown with suggestions Dropdown with suggestions

Customer Name	Order Number	Equipment Parent Group	Equipment ID	Equipment Name	Quantity	Price	Revenue	Rental Duration	Account Manager
ABC Corp	ORD12345	Bulldozer	EQ001	Bulldozer	50	10	\$500	120 days	John Smith
XYZ Inc.	ORD12346	Crane	EQ002	Crane	60	30	\$1,800	415 days	Eva Mark
DEF Ltd.	ORD12347	Generator	EQ003	Generator	100	1	\$100	90 days	John Smith
ABC Corp	ORD12348	Excavator	EQ004	Excavator	10	50	\$500	10 days	John Smith
XYZ Inc.	ORD12349	Forklift	EQ005	Forklift	21	71	\$1,491	600 days	Eva Mark
DEF Ltd.	ORD12350	Drill	EQ006	Drill	1	900	\$900	360 days	John Smith
ABC Corp	ORD12351	Backhoe	EQ007	Backhoe	65	3	\$195	176 days	Eva Mark
XYZ Inc.	ORD12352	Compactor	EQ008	Compactor	98	6	\$588	219 days	Eva Mark
DEF Ltd.	ORD12353	Loader	EQ009	Loader	45	15	\$675	100 days	John Smith
ABC Corp	ORD12354	Mixer	EQ010	Mixer	100	140	\$14,000	780 days	Eva Mark

2. Stock Report

Overview

The Stock Report provides a comprehensive snapshot of the equipment inventory as of a specific date, segmented by location—customer, warehouse, and in transit. The report is designed to track the current status, location, and quantity of each equipment unit. This documentation outlines how the quantity for each location type (customer, in transit, available) is calculated, ensuring precise inventory management.

Data Sources and Relationships

1. Primary Tables:

- **order_table**: Records details of all orders, including deliveries, pick-ups, and returns.
- **warehouse_product_list**: Contains details of all equipment, including unique identifiers, names, and material groups.
- **customer_list**: Stores customer details, including customer names and locations.
- **warehouse_list**: Stores warehouse details, including warehouse names and locations.
- **stock_adjustment_table**: Logs all stock adjustments, providing the current quantity of equipment based on the latest adjustments.

2. Key Fields:

- **order_table.order_id**: Unique identifier for each order.
- **order_table.delivery_date**: Date when equipment was delivered to the customer.

- **order_table.pickup_date**: Date when equipment was picked up from the customer.
- **order_table.check_in_date**: Date when equipment was checked in at the warehouse.
- **order_table.check_out_date**: Date when equipment was checked out from the warehouse.
- **warehouse_product_list.equipment_id**: Unique identifier for each piece of equipment.
- **warehouse_product_list.equipment_name**: Name of the equipment.
- **stock_adjustment_table.new_quantity**: The latest quantity of equipment available after adjustments.
- **customer_list.customer_name**: Name of the customer where the equipment is delivered.
- **warehouse_list.location_name**: Name of the warehouse where the equipment is stored.

Detailed Logic for Quantity Calculation

1. Base Quantity Calculation from Stock Adjustment Table:

- **Purpose**: Determine the current total quantity of each equipment in the system.
- **Source**: stock_adjustment_table
- **Logic**:
 - The quantity is determined based on the latest new_quantity value in the stock_adjustment_table for each equipment_id.
 - This value represents the current available quantity after considering all previous adjustments.
 - **SQL Query Example**:

```
SELECT equipment_id, MAX(new_quantity) AS total_quantity
FROM stock_adjustment_table
GROUP BY equipment_id;
```

2. Quantity with Customers:

- **Purpose**: Calculate the total quantity of equipment currently with customers.
- **Source**: order_table
- **Logic**:
 - Equipment is considered with a customer if the selected report date is between the delivery_date and the pickup_date.
 - Sum the quantities for such records to determine how much is with customers.
 - **SQL Query Example**:

```
SELECT equipment_id, SUM(quantity) AS customer_quantity
FROM order_table
WHERE delivery_date <= '2024-08-23' AND (pickup_date IS NULL OR pickup_date > '2024-08-23')
GROUP BY equipment_id;
```

3. Quantity In Transit:

- **Purpose:** Calculate the total quantity of equipment that is currently in transit.
- **Source:** order_table
- **Logic:**
 - Equipment is considered in transit if the selected report date is between:
 1. check_out_date and delivery_date (moving from warehouse to customer).
 2. pickup_date and check_in_date (moving from customer back to warehouse).
 - Sum the quantities for such records to determine the quantity in transit.
 - **SQL Query Example:**

```
SELECT equipment_id, SUM(quantity) AS transit_quantity
FROM order_table
WHERE (check_out_date <= '2024-08-23' AND delivery_date > '2024-08-23')
OR (pickup_date <= '2024-08-23' AND check_in_date > '2024-08-23')
GROUP BY equipment_id;
```

4. Quantity Available:

- **Purpose:** Calculate the quantity of equipment available in warehouses.
- **Source:** Derived calculation.
- **Logic:**
 - The available quantity is calculated as the difference between the total quantity (from the latest new_quantity in the stock_adjustment_table) and the sum of quantities with customers and in transit.
 - **Formula:**

makefile

Copy code

$\text{available_quantity} = \text{total_quantity} - (\text{customer_quantity} + \text{transit_quantity})$

- **SQL Query Example:**

```
SELECT equipment_id,
       total_quantity - (IFNULL(customer_quantity, 0) + IFNULL(transit_quantity, 0)) AS
       available_quantity
FROM (
  SELECT sa.equipment_id,
         MAX(sa.new_quantity) AS total_quantity,
         SUM(CASE WHEN o.delivery_date <= '2024-08-23' AND (o.pickup_date IS NULL OR
o.pickup_date > '2024-08-23') THEN o.quantity ELSE 0 END) AS customer_quantity,
         SUM(CASE WHEN (o.check_out_date <= '2024-08-23' AND o.delivery_date > '2024-08-
23')
               OR (o.pickup_date <= '2024-08-23' AND o.check_in_date > '2024-08-23') THEN
o.quantity ELSE 0 END) AS transit_quantity
  FROM stock_adjustment_table sa
  LEFT JOIN order_table o ON sa.equipment_id = o.equipment_id
  GROUP BY sa.equipment_id
) AS quantities;
```

5. Location Determination:

- **Customer Location:** If the equipment is with the customer, the location is retrieved from the customer_list using the customer_id.
- **Warehouse Location:** If the equipment is available, the location is determined by the warehouse_id from the order_table and warehouse_list.
- **In Transit:** The location is marked as "In Transit".
- **SQL Query Example:**

```
SELECT CASE
  WHEN delivery_date <= '2024-08-23' AND (pickup_date IS NULL OR pickup_date > '2024-08-23')
    THEN (SELECT customer_name FROM customer_list WHERE customer_id = o.customer_id)
  WHEN (check_out_date <= '2024-08-23' AND delivery_date > '2024-08-23')
    OR (pickup_date <= '2024-08-23' AND check_in_date > '2024-08-23')
    THEN 'In Transit'
  ELSE (SELECT location_name FROM warehouse_list WHERE warehouse_id = o.warehouse_id)
END AS location
FROM order_table o
WHERE equipment_id = '1714';
```

6. Data Display in the Report:

- **Date:** Shows the selected report date.
- **Equipment ID and Name:** Pulled from the warehouse_product_list.
- **Location:** Derived based on the equipment's status, either from customer_list, warehouse_list, or "In Transit".
- **Quantity:** Displayed as the calculated available quantity at the respective location.

Business Logic and Report Features:

- **Dynamic Date Filtering:**
 - The report supports filtering by a specific date to view inventory levels as of that date. The logic ensures that quantities are adjusted dynamically based on the selected date.
- **Status Tracking:**
 - Equipment status is precisely tracked using key dates (delivery, pick-up, check-in, check-out) to determine whether equipment is available, with customers, or in transit.
- **Location Accuracy:**
 - The report accurately identifies where each piece of equipment is located—be it a customer site, in transit, or in a warehouse—based on the selected date.
- **Quantity Precision:**
 - The report precisely calculates quantities at each location by summing up relevant data from orders and stock adjustments. This ensures accurate stock levels for effective inventory management.
- **Customer Location:**
 - When equipment is with a customer, the location reflects the customer's name, ensuring clear visibility into where rented equipment is situated.
- **Warehouse Location:**

- Available equipment is linked to its warehouse name, providing a clear understanding of stock available for future rentals.

Table Example:

Filters

Users can select multiple locations, statuses, or equipment name as they type the name of a location, it will be displayed as a selectable option in a dropdown list.

- **Date:** Calendar filter option with ability to select date or date range. (Past, current, or future dates)
- **Equipment Name:** Dropdown with multi-select options. As the user types, available locations will be suggested.
- **Location:** Dropdown with multi-select options. As the user types, available locations will be suggested.

The **Date** filter must be selected for the report to populate.

Data Fields

Users will have an overview of current inventory levels across locations, including the number of units available or rented.

- **Date:** The date on which we want to see stock inventory.
- **Equipment ID:** Unique identifier for each piece of equipment.
- **Equipment Name:** Descriptive name of the equipment.
- **Location:** Current location of the equipment (e.g., customer site, warehouse, in transit).
- **Quantity:** Number of units rented, available or in transit.

Sample Report

Filters: Date Filters: Equipment Name Filters: Location

Calendar selection Dropdown with suggestions Dropdown with suggestions

Date	Equipment ID	Equipment Name	Location	Quantity
2024-08-10	EQ001	Bulldozer	Customer Site 1	30
2024-08-10	EQ001	Bulldozer	Warehouse A	35
2024-08-10	EQ001	Bulldozer	In Transit	100
2024-08-10	EQ003	Generator	Customer Site 2	45
2024-08-10	EQ003	Generator	Warehouse A	0
2024-08-10	EQ003	Generator	In Transit	20
2024-08-10	EQ007	Backhoe	Customer Site 3	56
2024-08-10	EQ007	Backhoe	Customer Site 4	46

2024-08-10	EQ007	Backhoe	Warehouse C	0
2024-08-10	EQ007	Backhoe	Warehouse D	90
2024-08-10	EQ007	Backhoe	In Transit	0

3. Stock Movement Report

Overview

The Stock Movement Report provides a detailed record of all equipment movements within the organization, covering additions, removals, and transfers between locations. This report is critical for maintaining accurate inventory records, ensuring transparency in stock handling, and supporting audits of equipment movements. The report integrates data from both rental-related movements (tracked via the delivery process) and stock adjustments, providing a comprehensive view of all stock activities.

Data Sources and Relationships

1. Primary Tables:

- **order_table**: Records details of all orders, including the movement of equipment during the rental process.
- **warehouse_product_list**: Contains details of all equipment, including unique identifiers, names, and types.
- **stock_adjustment_table**: Logs all stock adjustments, capturing changes in equipment quantities at different locations.
- **delivery_table**: Tracks the delivery and pick-up processes of equipment, indicating when equipment is checked out and checked in at the warehouse.
- **customer_list**: Contains customer details, including customer names and addresses, relevant for movements involving customers.
- **warehouse_list**: Stores warehouse details, including warehouse names and locations.

2. Key Fields:

- **order_table.order_id**: Unique identifier for each order.
- **delivery_table.delivery_date**: Date when equipment was delivered to the customer.
- **delivery_table.pickup_date**: Date when equipment was picked up from the customer.
- **delivery_table.check_in_date**: Date when equipment was checked in at the warehouse.
- **delivery_table.check_out_date**: Date when equipment was checked out from the warehouse.
- **warehouse_product_list.equipment_id**: Unique identifier for each piece of equipment.
- **warehouse_product_list.equipment_name**: Name of the equipment.
- **warehouse_product_list.equipment_type**: Type of equipment (e.g., Power Tool, Hand Tool).
- **stock_adjustment_table.new_quantity**: The latest quantity of equipment after adjustments.
- **stock_adjustment_table.reason**: Reason for stock adjustment (e.g., Inspection, Not returned from customer).

- **stock_adjustment_table.movement_done_by**: User responsible for the stock adjustment.
- **customer_list.customer_name**: Name of the customer where the equipment is delivered.
- **warehouse_list.location_name**: Name of the warehouse where the equipment is stored.

Detailed Logic for Stock Movement Calculation

1. Movement Identification:

- **Movement Types**: The report categorizes movements into several predefined types:
 - **Check-in**: Equipment is returned to the warehouse from a customer.
 - **Check-out**: Equipment is dispatched from the warehouse to a customer.
 - **Not Returned from Customer**: Equipment expected to be returned by a customer but not received.
 - **Inspection**: Equipment movements related to inspection processes.
 - **Purchase**: Additions to inventory through purchase.
 - **Warehouse (WH) Differences**: Discrepancies identified during stock audits or adjustments.
 - **Other**: Miscellaneous movements not covered by the above categories.
- **Source**: Movements are derived from both the `delivery_table` (for check-in, check-out, etc.) and `stock_adjustment_table` (for inspection, purchase, etc.).

2. Movement Tracking:

- **Inbound Movements**:
 - **Check-In**: Recorded when equipment is marked as checked in at the warehouse, as noted in the `delivery_table.check_in_date`.
 - **Purchase**: Recorded as inbound when equipment is added to stock via a purchase, logged in the `stock_adjustment_table`.
- **Outbound Movements**:
 - **Check-Out**: Recorded when equipment is dispatched from the warehouse to a customer, based on the `delivery_table.check_out_date`.
 - **Not Returned from Customer**: Recorded if equipment is not returned by the customer on the expected pick-up date.
- **Transfers**:
 - Equipment transfers between warehouses are tracked using `stock_adjustment_table` and are recorded as a transfer movement type.

3. Data Merging:

- The report merges data from the `delivery_table` and `stock_adjustment_table` using:
 - **equipment_id**: To match equipment records.
 - **movement_date**: To track the date of each movement.
 - **movement_type**: To categorize the movement appropriately.
- **SQL Query Example**:

SELECT

```

COALESCE(d.movement_date, sa.movement_date) AS movement_date,
wpl.equipment_id,
wpl.equipment_name,
wpl.equipment_type,
COALESCE(d.quantity, sa.new_quantity) AS quantity,
COALESCE(d.movement_type, sa.reason) AS movement_type,
COALESCE(wl.location_name, cl.customer_name, sa.warehouse_location) AS location_name,
COALESCE(d.reason, sa.reason) AS reason,
COALESCE(d.movement_done_by, sa.movement_done_by) AS movement_done_by
FROM
(SELECT
    delivery_date AS movement_date,
    equipment_id,
    'Outbound' AS movement_type,
    quantity,
    'Check-Out' AS reason,
    check_out_by AS movement_done_by,
    warehouse_id
FROM delivery_table
WHERE check_out_date IS NOT NULL
UNION
SELECT
    check_in_date AS movement_date,
    equipment_id,
    'Inbound' AS movement_type,
    quantity,
    'Check-In' AS reason,
    check_in_by AS movement_done_by,
    warehouse_id
FROM delivery_table
WHERE check_in_date IS NOT NULL) AS d
FULL OUTER JOIN stock_adjustment_table sa ON d.equipment_id = sa.equipment_id
LEFT JOIN warehouse_product_list wpl ON wpl.equipment_id = COALESCE(d.equipment_id,
sa.equipment_id)
LEFT JOIN warehouse_list wl ON wl.warehouse_id = COALESCE(d.warehouse_id, sa.warehouse_id)
LEFT JOIN customer_list cl ON cl.customer_id = sa.customer_id
WHERE COALESCE(d.movement_date, sa.movement_date) BETWEEN '2024-07-30' AND '2024-08-28';

```

4. Reason for Movement:

- For each movement, the reason is recorded, providing context for the movement. This could include reasons like:
 - Inspection:** Equipment was moved for inspection purposes.
 - Not Returned from Customer:** Equipment was not returned as expected by the customer.
- The reason is pulled directly from the stock_adjustment_table.reason or inferred from the delivery_table stage.

5. User Accountability:

- The report captures details about the user who performed the movement, providing accountability for each action.
- Source:**
 - For stock adjustments: stock_adjustment_table.movement_done_by.

- For deliveries/pick-ups: delivery_table.check_in_by, delivery_table.check_out_by.
- **SQL Example:**

```
SELECT movement_done_by
FROM stock_adjustment_table
WHERE equipment_id = '1714' AND movement_date = '2024-08-23';
```

Table Example:

Filters

Users can select multiple pieces of equipment, and as they type the name of an equipment, it will be displayed as a selectable option in a dropdown list.

- **Date:** Calendar filter option with ability to select date or date range.
- **Equipment Name:** Dropdown with multi-select options. As the user types, available locations will be suggested.

The **Date** filter must be selected for the report to populate.

Data Fields

Users will have an overview of all equipment movements between locations, including the date and type of each movement, to track the transfer history and ensure timely and accurate stock management.

- **Date:** The date on which the movement occurred/or period during which the movements occurred.
- **Equipment ID:** Unique identifier for each piece of equipment.
- **Type:** Indicates whether the movement resulted in an increase or decrease in the stock quantity.
- **Quantity:** The number of units involved in the movement.
- **Movement Type:** Type of movement, such as rental, inspection et.
- **Warehouse Location:** The warehouse location of the equipment.
- **Note:** Additional details related to the movement, such as the delivery id for rental-related movements.
- **User:** The user responsible for executing the movement.

Sample Report

Filters: Date Filters: Equipment Name

Calendar Selection Dropdown with suggestions

Date	Equipment ID	Equipment Name	Type	Quantity	Movement Type	Warehouse Location	Reason	Movement done by
2024-08-10	EQ001	Bulldozer	Increase	2	Check in	Warehouse A	DEL-1	nnotova
2024-08-11	EQ001	Bulldozer	Decrease	202	Check out	Warehouse B		nnotova
2024-08-11	EQ001	Bulldozer	Decrease	809	WH differences	Warehouse A		nnotova
2024-08-12	EQ001	Bulldozer	Increase	20212	Purchase	Warehouse B		nnotova
2024-08-13	EQ001	Bulldozer	Decrease	200	not returned from customer	Warehouse D	DEL-1	nnotova
2024-08-15	EQ001	Bulldozer	Decrease	28	Inspection	Warehouse B		nnotova
2024-08-17	EQ001	Bulldozer	Decrease	411	Other	Warehouse C		nnotova
2024-08-17	EQ001	Bulldozer	Decrease	9	WH differences	Warehouse B		nnotova
2024-08-18	EQ001	Bulldozer	Decrease	10	Check out	Warehouse C	DEL-1	nnotova
2024-08-30	EQ001	Bulldozer	Increase	4082	Check in	Warehouse G		nnotova

5. In Transit Report

Description:

The In Transit Equipment Report provides real-time tracking of equipment currently in transit between the company's warehouses and customers. This report is crucial for managing logistics, ensuring timely deliveries, and tracking the movement of equipment that has not yet been delivered to a customer or returned to the warehouse. The report can display equipment dispatched from the warehouse and not yet delivered or picked up from a customer and not yet checked back into the warehouse.

Report Contents:

1. Equipment Information:

- **equipment_id** - *INT*
 - The unique identifier for each piece of equipment.
 - **Source:** warehouse_product_list.equipment_id
- **equipment_name** - *VARCHAR(255)*
 - The name of the equipment item.
 - **Source:** warehouse_product_list.equipment_name

2. Order and Delivery Information:

- **customer** - *VARCHAR(255)*
 - The name of the customer associated with the equipment in transit.
 - **Source:** customer_list.customer_name
- **order_number** - *VARCHAR(255)*
 - The order number related to the equipment in transit.
 - **Source:** order_table.order_number

- **delivery_number** - *VARCHAR(255)*
 - The delivery identifier for tracking multiple deliveries under the same order.
 - **Source:** delivery_details.delivery_id
- 3. **Location Information:**
 - **from_location** - *VARCHAR(255)*
 - The location from which the equipment was dispatched. This can be either a warehouse or a customer location.
 - **Source:** delivery_details.from_location (links to warehouse_list.warehouse_name or customer_list.customer_name based on delivery_type)
 - **to_location** - *VARCHAR(255)*
 - The destination to which the equipment is being delivered. This can be either a customer location or a warehouse.
 - **Source:** delivery_details.to_location (links to customer_list.customer_name or warehouse_list.warehouse_name based on delivery_type)
- 4. **Dispatch and Delivery Dates:**
 - **dispatch_date** - *DATE*
 - The date when the equipment was dispatched from the from_location.
 - **Source:** delivery_details.dispatch_date
 - **delivery_date** - *DATE* (nullable)
 - The expected or actual date of delivery to the to_location. This remains NULL until the delivery is confirmed.
 - **Source:** delivery_details.delivery_date
- 5. **Delivery Status:**
 - **status** - *ENUM('In Transit')*
 - Indicates that the equipment is currently in transit, either on its way to the customer or being returned to the warehouse.
 - **Source:** Derived based on the absence of a confirmed delivery_date.

Business Logic:

- **Identifying In-Transit Equipment:**
 - The report includes equipment records where:
 - **For outgoing deliveries:** The equipment has been checked out from a warehouse (dispatch_date is set), but the delivery_date is still NULL.
 - **For return deliveries:** The equipment has been picked up from a customer location, but the check_in_date at the warehouse is still NULL.
- **Location Handling:**
 - The from_location and to_location fields dynamically switch between customer and warehouse names based on the context of the delivery (whether it's being delivered to a customer or returned to the warehouse).
- **Dispatch and Expected Delivery Dates:**

- The `dispatch_date` is the date the equipment leaves the `from_location`, marking the start of its in-transit status. The `delivery_date` is expected but may be updated once the delivery is confirmed.
- **Filtering and Sorting:**
 - Users can filter the report based on `from_location`, `to_location`, `equipment_name`, and `dispatch_date`. This helps in pinpointing specific items in transit, understanding delays, or tracking equipment by date or location.

SQL Example:

```
SELECT
    dd.equipment_id,
    wp.equipment_name,
    cl.customer_name AS customer,
    ot.order_number,
    dd.delivery_id AS delivery_number,
    CASE
        WHEN dd.delivery_type = 'Delivery' THEN wl.warehouse_name
        ELSE cl.customer_name
    END AS from_location,
    CASE
        WHEN dd.delivery_type = 'Delivery' THEN cl.customer_name
        ELSE wl.warehouse_name
    END AS to_location,
    dd.dispatch_date,
    dd.delivery_date,
    'In Transit' AS status
FROM
    delivery_details dd
    JOIN order_table ot ON dd.order_id = ot.order_id
    JOIN warehouse_product_list wp ON dd.equipment_id = wp.equipment_id
    LEFT JOIN warehouse_list wl ON dd.from_location = wl.warehouse_id
    LEFT JOIN customer_list cl ON dd.to_location = cl.customer_id
WHERE
    dd.delivery_type IN ('Delivery', 'Return Delivery')
    AND dd.delivery_date IS NULL;
```

Key Points:

- **Single Source of Truth:** The `delivery_details` table serves as the primary source for identifying in-transit equipment, ensuring consistency across the report.
- **Flexible Location Identification:** The report dynamically adapts location fields based on the delivery type, offering clear insights into where equipment is coming from and going to.
- **Business Relevance:** The report is designed to assist logistics and warehouse managers in ensuring equipment is tracked efficiently, reducing the risk of misplaced or delayed items.
- **Custom Filters:** By allowing filters on `from_location`, `to_location`, `equipment_name`, and `dispatch_date`, the report empowers users to drill down into specific details and manage operations effectively.

Table Example:

Filters

Users can select multiple locations (from/to), and as they type the name of a location, it will be displayed as a selectable option in a dropdown list.

- **From Location:** Dropdown with multi-select options. As the user types, available locations will be suggested.
- **To Location:** Dropdown with multi-select options. As the user types, available locations will be suggested.

No mandatory filters are required, as the report will automatically retrieve 'In Transit' items as of the current date.

Data Field

Users will have an overview of the equipment ship from and to specific location as of today, with information how many days is equipment in transit to track any delayed deliveries.

- **Equipment ID:** Unique identifier for each piece of equipment.
- **Equipment Name:** Descriptive name of the equipment.
- **Customer Name:** Name of customer associated with the order.
- **Order Number:** Unique identifier for the order.
- **Delivery Number:** Unique identifier for tracking the delivery process.
- **From Location:** The location from which the equipment was dispatched.
- **To Location:** The destination location for the equipment.
- **Dispatch Date:** Date when the equipment was dispatched.

Sample Report

Filters: From Location Filters: To Location

Dropdown with suggestions Dropdown with suggestions

Equipment ID	Equipment Name	Customer Name	Order Number	Delivery Number	From Location	To Location	Dispatch Date
EQ001	Bulldozer	ABC Corp	ORD12345	DEL123	Warehouse A	Customer Site 1	2024-08-10
EQ002	Crane	XYZ Inc.	ORD12346	DEL124	Warehouse B	Customer Site 2	2024-08-11
EQ003	Generator	DEF Ltd.	ORD12347	DEL125	Warehouse A	Customer Site 3	2024-08-11
EQ004	Excavator	ABC Corp	ORD12348	DEL126	Warehouse C	Customer Site 4	2024-08-11
EQ005	Forklift	XYZ Inc.	ORD12349	DEL127	Warehouse D	Customer Site 1	2024-08-11
EQ006	Drill	DEF Ltd.	ORD12350	DEL128	Customer Site 1	Warehouse A	2024-08-11
EQ007	Backhoe	ABC Corp	ORD12351	DEL129	Customer Site 2	Warehouse B	2024-08-11
EQ008	Compactor	XYZ Inc.	ORD12352	DEL130	Customer Site 3	Warehouse A	2024-08-09
EQ009	Loader	DEF Ltd.	ORD12353	DEL131	Customer Site 4	Warehouse C	2024-08-10
EQ010	Mixer	ABC Corp	ORD12354	DEL132	Customer Site 1	Warehouse D	2024-08-12

6. Planned Removal Report

Description:

The Planned Removal Report offers comprehensive insights into scheduled returns of equipment from customers, focusing exclusively on "return deliveries" where the equipment is being returned to the warehouse rather than directly to another customer. This report is designed to track equipment scheduled for pickup from customers based on the removal date, ensuring timely returns and optimal inventory management. The report only includes records where the `delivery_type` is "return" and `delivery_to_customer` is False.

Columns and Data Sources:

1. Equipment Information:

- `equipment_id` - *INT*
 - The unique identifier for each piece of equipment.

- **Source:** order_table.equipment_id
- **equipment_name** - *VARCHAR(255)*
 - The name of the equipment item.
 - **Source:** warehouse_product_list.equipment_name
- 2. **Order and Delivery Information:**
 - **customer_name** - *VARCHAR(255)*
 - The name of the customer associated with the scheduled removal.
 - **Source:** customer_list.customer_name (joined via order_table.customer_id)
 - **order_number** - *VARCHAR(255)*
 - The order number related to the scheduled removal.
 - **Source:** order_table.order_number
 - **delivery_number** - *VARCHAR(255)*
 - The delivery identifier for tracking multiple deliveries under the same order.
 - **Source:** delivery_details.delivery_id (joined via order_table.order_id)
- 3. **Location Information:**
 - **from_location** - *VARCHAR(255)*
 - The location from which the equipment is planned to be removed. This field dynamically changes based on the type of delivery:
 - **For standard deliveries** (where delivery_type is "delivery"):
 - **Source:** customer_list.customer_name
 - **Explanation:** This reflects the customer's location since the equipment is being removed from the customer's site.
 - **For return deliveries** (where delivery_type is "return"):
 - **Source:** delivery_details.from_location
 - **Explanation:** This location represents where the equipment is currently located before being returned, which could be either the customer's location or another defined return location.
 - **to_location** - *VARCHAR(255)*
 - The destination to which the equipment is being returned. This can either be a warehouse or another customer location:
 - **For standard deliveries** (where delivery_type is "delivery"):
 - **Source:** delivery_details.to_location linked to warehouse_list.warehouse_name
 - **Explanation:** Represents the warehouse or other predefined location where the equipment is expected to be returned.
 - **For return deliveries** (where delivery_type is "return"):
 - **Source:** delivery_details.to_location
 - **Explanation:** This location could either be a customer or warehouse, depending on where the equipment is being sent after being picked up.
- 4. **Removal Date:**
 - **removal_date** - *DATE*
 - The scheduled date for the equipment to be removed from the customer's location. This date is determined by the contract end date:

- **Source:** order_table.contract_end_date
 - **Explanation:** The contract_end_date from the order_table serves as the planned removal date, marking the expected end of the rental period and signaling when the equipment should be prepared for return. The report filters these dates to include only future dates or past dates where the equipment has not yet been picked up.
5. **Quantity:**
- **quantity** - *INT*
 - The number of equipment units scheduled for removal.
 - **Source:** order_table.quantity

Business Logic and Data Handling:

1. Return Deliveries Only:

- The report exclusively includes deliveries where delivery_type is "return" and delivery_to_customer is False. This ensures the report focuses solely on equipment returning to a warehouse, excluding any items that are being sent to another customer.

2. Scheduled Removals:

- The report includes equipment records where the contract_end_date (used as the removal date) has been reached or is upcoming, and the equipment has not yet been picked up from the customer's location. This ensures that only equipment still awaiting removal is included in the report.

3. Exclusion of Completed Pickups:

- Equipment that has already been picked up from the customer (as indicated by a confirmed pickup_date in the delivery_details table) is excluded from this report. This prevents the inclusion of equipment that has already been returned to the warehouse.

4. Location Handling:

- The from_location always references the customer_list.customer_name, representing the location from where the equipment is to be picked up.
- The to_location references warehouse_list.warehouse_name, identifying the warehouse where the equipment is scheduled to be returned.

5. Date Handling:

- The removal_date is directly derived from the contract_end_date in the order_table. This date signifies when the equipment is expected to be removed from the customer's location.

SQL Example:

```
SELECT
    ot.equipment_id,
    wp.equipment_name,
    cl.customer_name,
    ot.order_number,
    dd.delivery_id AS delivery_number,
    CASE
        WHEN dd.delivery_type = 'return' THEN dd.from_location
        ELSE cl.customer_name
    END AS from_location,
    CASE
```

```

        WHEN dd.delivery_type = 'return' THEN dd.to_location
        ELSE wl.warehouse_name
    END AS to_location,
    ot.contract_end_date AS removal_date,
    ot.quantity
FROM
    order_table ot
    JOIN delivery_details dd ON ot.order_id = dd.order_id
    JOIN warehouse_product_list wp ON ot.equipment_id = wp.equipment_id
    JOIN customer_list cl ON ot.customer_id = cl.customer_id
    LEFT JOIN warehouse_list wl ON dd.to_location = wl.warehouse_id
WHERE
    ot.contract_end_date >= CURDATE()
    AND dd.pickup_date IS NULL;

```

Key Points:

- **Source Identification:** Clear linkage of each report column to its corresponding table and field, ensuring that the report's structure is traceable and easily understandable.
- **Real-Time Monitoring:** Filters equipment by the current or future removal date and excludes any equipment that has already been picked up, allowing for accurate, real-time monitoring of equipment scheduled for removal.
- **Conditional Location Logic:** Uses business logic to determine whether the from_location and to_location should be based on customer or warehouse data, depending on the delivery type.

Table Example:

Filters

Users can select multiple options, and as they type the name of a location or customer name, it will be displayed as a selectable option in a dropdown list.

- **Removal Date:** Calendar filter option with ability to select date or date range.
- **Customer Name: Dropdown** with multi-select options. As the user types, available locations will be suggested.
- **From Location: Dropdown** with multi-select options. As the user types, available locations will be suggested.

The **Removal Date** filter must be selected for the report to populate.

Data Fields

Users will have an overview of the equipment ship from and to specific location as of today, with information how many days is equipment in transit to track any delayed deliveries.

- **Removal Date:** The date when the removal is planned.
- **Customer Name:** Name of the customer from whom the equipment is to be removed.
- **Order Number:** Unique identifier for the order associated with the removal.
- **Equipment ID:** Unique identifier for the equipment.
- **Equipment Name:** Descriptive name of the equipment.
- **From Location:** The location from which the equipment will be picked up.
- **To Location:** The destination location for the equipment after removal.

- **Quantity:** The quantity of equipment scheduled for removal.

Sample Report

Filters: Removal Date

Calendar selection

Filters: Customer Name

Dropdown with suggestions

Filters: From Location

Dropdown with suggestions

Removal Date	Customer Name	Order Number	Equipment ID	Equipment Name	Delivery Number	From Location	To Location	Quantity
2024-08-12	ABC Corp	ORD12345	EQ001	Bulldozer	DEL123	Customer Site 1	Warehouse A	2
2024-08-13	XYZ Inc.	ORD12346	EQ002	Crane	DEL124	Customer Site 2	Warehouse B	202
2024-08-11	DEF Ltd.	ORD12347	EQ003	Generator	DEL125	Customer Site 3	Warehouse A	809
2024-08-14	ABC Corp	ORD12348	EQ004	Excavator	DEL126	Customer Site 4	Warehouse C	20212
2024-08-12	XYZ Inc.	ORD12349	EQ005	Forklift	DEL127	Customer Site 1	Warehouse D	200
2024-08-14	DEF Ltd.	ORD12350	EQ006	Drill	DEL128	Customer Site 5	Warehouse B	28
2024-08-13	ABC Corp	ORD12351	EQ007	Backhoe	DEL129	Customer Site 6	Warehouse E	411
2024-08-12	XYZ Inc.	ORD12352	EQ008	Compactor	DEL130	Customer Site 7	Warehouse F	9
2024-08-11	DEF Ltd.	ORD12353	EQ009	Loader	DEL131	Customer Site 8	Warehouse C	10
2024-08-14	ABC Corp	ORD12354	EQ010	Mixer	DEL132	Customer Site 9	Warehouse G	4082

7. Planned Deliveries Report

Description:

The Planned Deliveries Report provides a comprehensive overview of upcoming equipment deliveries, focusing on deliveries where the `delivery_type` is "delivery" or where `delivery_to_customer` is `True`. This report is crucial for managing and tracking equipment that is scheduled to be delivered to customers based on the contract start date. By ensuring that all planned deliveries are visible and managed efficiently, this report helps identify potential delays and optimize logistics.

Columns and Data Sources:

1. Delivery Information:

- **delivery_date** - DATE
 - The scheduled date for delivering the equipment to the customer.

- **Source:** order_table.contract_start_date
- **Explanation:** This date is set based on the contract start date, which serves as the planned delivery date when equipment is expected to reach the customer.
- 2. **Customer Information:**
 - **customer_name** - VARCHAR(255)
 - The name of the customer who will receive the delivery.
 - **Source:** customer_list.customer_name (linked via order_table.customer_id)
- 3. **Order and Delivery Identifiers:**
 - **order_number** - VARCHAR(255)
 - The unique identifier for the order associated with the planned delivery.
 - **Source:** order_table.order_number
 - **delivery_number** - VARCHAR(255)
 - The identifier for the specific delivery event within the order.
 - **Source:** delivery_details.delivery_id (linked via order_table.order_id)
- 4. **Equipment Information:**
 - **equipment_id** - INT
 - The unique identifier for each piece of equipment being delivered.
 - **Source:** delivery_details.equipment_id
 - **equipment_name** - VARCHAR(255)
 - The name of the equipment being delivered.
 - **Source:** warehouse_product_list.equipment_name
- 5. **Location Information:**
 - **from_location** - VARCHAR(255)
 - The origin location from which the equipment is being delivered. This could be a warehouse or a previous customer's location:
 - **Source:** warehouse_list.warehouse_name or customer_list.customer_name (depending on where the delivery is being dispatched from, linked via delivery_details.from_location)
 - **to_location** - VARCHAR(255)
 - The destination where the equipment is being delivered:
 - **Source:** customer_list.customer_name or warehouse_list.warehouse_name (depending on the delivery type, linked via delivery_details.to_location)
- 6. **Quantity:**
 - **quantity** - INT
 - The quantity of equipment units being delivered.
 - **Source:** delivery_details.quantity

Business Logic and Data Handling:

- **Inclusion Criteria:**
 - The report includes deliveries where delivery_type is "delivery" or where delivery_to_customer is True. This ensures that the report focuses on equipment that is actively being delivered to customers, including both direct deliveries and deliveries as part of a return process where the equipment is sent directly to another customer.

- **Scheduled Deliveries:**
 - The report displays deliveries that are scheduled based on the `contract_start_date` from the `order_table`. This date reflects when the equipment should arrive at the customer's location, allowing for precise tracking and management of delivery schedules.
- **Location Handling:**
 - The `from_location` and `to_location` fields are populated based on the source and destination of the delivery. For example, if the equipment is being delivered from a warehouse, `from_location` will reference `warehouse_list.warehouse_name`. If it is being transferred from one customer directly to another, `from_location` will reference the previous `customer_list.customer_name`, and `to_location` will reflect the new customer's location.
- **Real-Time Data Accuracy:**
 - The report dynamically filters and displays only those deliveries that are planned within the specified delivery date range, ensuring that the information is up-to-date and actionable.

SQL Example:

```
SELECT
    ot.contract_start_date AS delivery_date,
    cl.customer_name,
    ot.order_number,
    dd.delivery_id AS delivery_number,
    wp.equipment_name,
    ot.equipment_id,
    CASE
        WHEN dd.from_location IS NOT NULL THEN wl.warehouse_name
        ELSE cl_from.customer_name
    END AS from_location,
    CASE
        WHEN dd.to_location IS NOT NULL THEN cl.customer_name
        ELSE wl_to.warehouse_name
    END AS to_location,
    ot.quantity
FROM
    order_table ot
    JOIN delivery_details dd ON ot.order_id = dd.order_id
    JOIN warehouse_product_list wp ON ot.equipment_id = wp.equipment_id
    JOIN customer_list cl ON ot.customer_id = cl.customer_id
    LEFT JOIN warehouse_list wl ON dd.from_location = wl.warehouse_id
    LEFT JOIN customer_list cl_from ON dd.from_location = cl_from.customer_id
    LEFT JOIN warehouse_list wl_to ON dd.to_location = wl_to.warehouse_id
WHERE
    (dd.delivery_type = 'delivery' OR dd.delivery_to_customer = TRUE)
    AND ot.contract_start_date >= CURDATE();
```

Key Points:

- **Scheduled Deliveries Focus:** The report strictly includes planned deliveries where equipment is either being newly delivered to a customer or being transferred directly between customers without returning to a warehouse.
- **Location Management:** Accurately tracks the origin (from_location) and destination (to_location) of deliveries, whether they involve warehouses or direct customer-to-customer transfers.
- **Real-Time Monitoring:** Filters deliveries by the planned contract_start_date, ensuring the report reflects current and upcoming deliveries for effective planning and logistics management.

Table Example:

Filters

Users can select multiple options, and as they type the name of a location or customer name, it will be displayed as a selectable option in a dropdown list.

- **Delivery Date:** Calendar filter option with ability to select date or date range.
- **Delivery Location: Dropdown** with multi-select options. As the user types, available locations will be suggested.
- **Customer Name: Dropdown** with multi-select options. As the user types, the available customer's name will be suggested.

The **Delivery Date** filter must be selected for the report to populate.

Data Fields

Users will have an overview of the equipment ship from and to specific location as of today, with information how many days is equipment in transit to track any delayed deliveries.

- **Delivery Date:** Date when the delivery is scheduled (coming from the contract start date).
- **Customer Name:** Name of the customer who will receive the equipment.
- **Order Number:** Unique identifier for the order associated with the delivery.
- **Equipment ID:** Unique identifier for the equipment.
- **Equipment Name:** Descriptive name of the equipment.
- **Shipping Type:** Specifies whether the removal is handled by internal resources or a hired service.
- **From Location:** The location from where the equipment will be dispatched.
- **To Location:** The destination location for the equipment.
- **Quantity:** The quantity of equipment scheduled for delivery.

Sample Report

Filters: Delivery Date

Filters: Customer Name

Filters: Delivery Location

Calendar selection

Dropdown with suggestions

Dropdown with suggestions

Delivery Date	Customer Name	Order Number	Equipment ID	Equipment Name	Delivery Number	From Location	To Location	Quantity
2024-08-12	ABC Corp	ORD12345	EQ001	Bulldozer	DEL123	Customer Site 1	Warehouse A	2
2024-08-13	XYZ Inc.	ORD12346	EQ002	Crane	DEL124	Customer Site 2	Warehouse B	202
2024-08-11	DEF Ltd.	ORD12347	EQ003	Generator	DEL125	Customer Site 3	Warehouse A	809
2024-08-14	ABC Corp	ORD12348	EQ004	Excavator	DEL126	Customer Site 4	Warehouse C	20212
2024-08-12	XYZ Inc.	ORD12349	EQ005	Forklift	DEL127	Customer Site 1	Warehouse D	200
2024-08-14	DEF Ltd.	ORD12350	EQ006	Drill	DEL128	Customer Site 5	Warehouse B	28
2024-08-13	ABC Corp	ORD12351	EQ007	Backhoe	DEL129	Customer Site 6	Warehouse E	411
2024-08-12	XYZ Inc.	ORD12352	EQ008	Compactor	DEL130	Customer Site 7	Warehouse F	9
2024-08-11	DEF Ltd.	ORD12353	EQ009	Loader	DEL131	Customer Site 8	Warehouse C	10
2024-08-14	ABC Corp	ORD12354	EQ010	Mixer	DEL132	Customer Site 9	Warehouse G	4082

8. Utilization Report

Description:

The Equipment Utilization Report provides a detailed analysis of equipment usage, highlighting rental frequency, duration, revenue generated, and the proportion of time equipment spends in the warehouse versus rented out. This report is essential for understanding the efficiency of equipment utilization, optimizing rental operations, and maximizing revenue.

Columns and Data Sources:

1. Equipment Information:

- **equipment_id** - *VARCHAR(255)*
 - A unique identifier for each piece of equipment.
 - **Source:** warehouse_product_list.equipment_id
- **equipment_name** - *VARCHAR(255)*
 - The name of the equipment.
 - **Source:** warehouse_product_list.equipment_name

2. Utilization Metrics:

- **duration_of_use** - *VARCHAR(255)*

- The total time (in hours or days) that the equipment has been rented out during the specified report period.
 - **Source:** Calculated based on `delivery_details.delivery_date` and `delivery_details.pickup_date`
 - **Explanation:** The difference between the `pickup_date` and the `delivery_date` from the `delivery_details` table determines how long the equipment was in use for each rental period.
3. **Order Information:**
- **order_number** - *VARCHAR(255)*
 - The unique identifier of the order associated with the equipment rental.
 - **Source:** `order_table.order_number`
4. **Revenue Insights:**
- **revenue_generated** - *DECIMAL(10, 2)*
 - The total revenue generated from the equipment during the report period.
 - **Source:** `order_table.order_amount`
 - **Explanation:** The revenue is calculated based on the rental amount from the order linked to the equipment. If the rental period spans multiple months, the revenue is split proportionally across those months.
5. **Warehouse vs. Rented Metrics:**
- **% warehouse** - *DECIMAL(5, 2)*
 - The percentage of time the equipment spent in the warehouse during the report period.
 - **Source:** Calculated based on the stock report and `stock_adjustment_table`
 - **Explanation:** This metric is calculated by determining the time the equipment was not rented (i.e., available in the warehouse). It is derived from the `stock_adjustment_table` and the status changes recorded in the stock report.
 - **% rented** - *DECIMAL(5, 2)*
 - The percentage of time the equipment was rented out during the report period.
 - **Source:** Calculated based on the stock report and `delivery_details`
 - **Explanation:** This metric represents the proportion of time the equipment was rented out compared to its availability. It is calculated by comparing the rental duration against the total available time within the report period.
6. **Paid Period:**
- **paid_period** - *VARCHAR(255)*
 - The paid rental period for the equipment, indicating the span from the contract start date to the contract end date.
 - **Source:** `order_table.contract_start_date` and `order_table.contract_end_date`
 - **Explanation:** This column reflects the official rental period for which the customer is billed, regardless of the actual time the equipment was in use.

Business Logic and Data Handling:

- **Percentage Metrics Calculation:**

- The % warehouse and % rented metrics are calculated using data from the stock report. The time the equipment spends in different statuses (e.g., in transit, rented, or available) is measured and converted into percentages. These percentages show the balance between equipment being available for rent (in the warehouse) and actually rented out.
- **Rental Duration:**
 - The duration_of_use metric is calculated by finding the difference between the delivery_date and the pickup_date from the delivery_details table. This value is then used to assess the total time the equipment was actively rented out during the reporting period.
- **Revenue Allocation:**
 - If a rental period spans multiple months or the reporting period, revenue is split proportionally across those periods. For instance, if an order covers both August and September, and the total revenue is \$10,000, the report will allocate \$5,000 to each month, provided that both months are included in the report period.
- **Paid Period Handling:**
 - The paid_period reflects the contractual dates during which the equipment is billed. This column provides a clear distinction between the official billing period and the actual usage duration, which may differ due to early returns or extended rentals.

SQL Example:

```
SELECT
    wp.equipment_id,
    wp.equipment_name,
    CONCAT(DATEDIFF(dd.pickup_date, dd.delivery_date), ' hours') AS duration_of_use,
    ot.order_number,
    SUM(ot.order_amount) AS revenue_generated,
    ROUND((SUM(CASE WHEN sr.status = 'warehouse' THEN 1 ELSE 0 END) / COUNT(*)) * 100, 2) AS '% warehouse',
    ROUND((SUM(CASE WHEN sr.status = 'rented' THEN 1 ELSE 0 END) / COUNT(*)) * 100, 2) AS '% rented',
    CONCAT(ot.contract_start_date, ' to ', ot.contract_end_date) AS paid_period
FROM
    order_table ot
    JOIN delivery_details dd ON ot.order_id = dd.order_id
    JOIN warehouse_product_list wp ON ot.equipment_id = wp.equipment_id
    LEFT JOIN stock_report sr ON wp.equipment_id = sr.equipment_id
WHERE
    ot.contract_start_date BETWEEN '2024-07-30' AND '2024-08-28'
GROUP BY
    wp.equipment_id, wp.equipment_name, ot.order_number, ot.contract_start_date, ot.contract_end_date
ORDER BY
    wp.equipment_id;
```

Key Points:

- **Utilization Focus:** The report primarily tracks how long and how frequently equipment is used (rented) during the specified period.
- **Revenue and Efficiency Insights:** By combining utilization data with revenue and warehouse availability, the report provides comprehensive insights into the financial and operational efficiency of the equipment fleet.

- **Time-Based Analysis:** The use of both `duration_of_use` and `paid_period` allows for a comparison between the actual rental duration and the period for which the customer is billed, highlighting any discrepancies or opportunities for optimizing rental contracts.

Table Example:

Filters

Users can select multiple options, and as they type the name of an equipment or customer name, it will be displayed as a selectable option in a dropdown list.

- **Reporting Period:** Calendar filter option with ability to select date or date range.
- **Equipment Name: Dropdown** with multi-select options. As the user types, available locations will be suggested.
- **Customer Name: Dropdown** with multi-select options. As the user types, the available customer's name will be suggested.

The **Report Period** filter must be selected for the report to populate. influences metrics like **rental duration**, **paid period**, **revenue generated**, and percentage calculations to reflect utilization within the selected dates.

Data Fields

Users will have an overview of the equipment ship from and to specific location as of today, with information how many days is equipment in transit to track any delayed deliveries.

- **Reporting Date:** Defines the period for analyzing equipment usage.
- **Equipment ID:** Unique identifier for the equipment.
- **Equipment Name:** Descriptive name of the equipment.
- **Rental Duration:** Total time the equipment has been rented out during the selected period.
- **Revenue Generated:** Total revenue generated from the rental of the equipment.
- **Percentage in Warehouse:** Percentage of equipment in the warehouse.
- **Percentage Rented:** Percentage of the equipment that is currently rented out.
- **Paid Period:** The difference between the contract end date and the actual removal date, representing the paid period for the rental.

Sample Report

Filters: Report Period

Calendar selection

Filters: Customer Name

Dropdown with suggestions

Filters: Equipment Name

Dropdown with suggestions

Equipment ID	Equipment Name	Duration of Use	Order Number	Revenue Generated	% Warehouse	% Rented	Paid Period
EQ001	Bulldozer	120 days	ORD12345	\$50,000	30%	70%	120 days
EQ002	Crane	15 days	ORD12346	\$100,000	70%	10%	12 days
EQ003	Generator	10 days.	ORD12347	\$20,000	10%	90%	9 days
EQ004	Excavator	90 days	ORD12348	\$13,000	12%	88%	70 days

EQ005	Forklift	34 days	ORD12349	\$75,000	49%	50%	34 days
EQ006	Drill	21 days.	ORD12350	\$21,000	7%	93%	20 days
EQ007	Backhoe	80 days	ORD12351	\$1,000	21%	59%	76 days
EQ008	Compactor	110 days	ORD12352	\$37,000	78%	12%	110 days
EQ009	Loader	170 days	ORD12353	\$45,000	13%	87%	180 days
EQ010	Mixer	14 days	ORD12354	\$77,000	68%	22%	7 days

Page Name: Authorizations

User List

Show 10 entries

+ Add User

USER	EMAIL	CONTACT NUMBER	PERMISSION	ACTIONS
Amy Davis	amydavis@laurel.com	+17894562301	Create Order	
Chloe Rodriguez	chloerodriguez@laurel.com	+17894329870	Check In Maintain Master Data	
David Miller	davidmiller@laurel.com	+19878901234	Check In	
Emily Johnson	emilyjohnson@laurel.com	+19876543210	Pick Up	
Emma Wilson	emmawilson@laurel.com	+13456789012	Maintain Master Data Repair	
Isabella Brown	isabellabrown@laurel.com	+19876543210	Stock Adjustment Maintain Master Data	
James Peter	jamespeter@laurel.com	+18245678901	Stock Adjustment Create Order	
John Williams	johnwilliams@laurel.com	+12345678901	Maintain Master Data	
Liam Martinez	liammartinez@laurel.com	+19876543210	Maintain Master Data	
Michael Brown	michaelbrown@laurel.com	+18901234567	Stock Adjustment Create Order	

Showing 1 to 10 of 15 entries

Previous 1 2 Next

1. Purpose:

The User Management screen is designed for administrators to manage user accounts and their associated permissions within the rental process application. This page allows admins to add new users, edit existing users, and manage the permissions assigned to each user. Permissions correspond to specific tasks or roles in the application, and these permissions are stored in the permissions column of the user_list table in the database.

Component: Top Bar

- *Position:* Across the top of the interface.
- *Items:*
 - *Search Box:* Allows for searching within workflows or users.
 - *User Profile:* Displayed in the top right corner with the user's email and status (e.g., "Available").
 - *Logo:* Positioned at the top left corner, providing branding for the application.

2. UI Components

2.1. User List Panel

Component: User Table

- *Position:* Central area of the screen.
- *Structure:*
 - *Columns:*
 - **User:** Displays the name of the user.
 - **Email:** The user's email address.
 - **Contact Number:** The user's contact phone number.
 - **Permissions:** Displays the list of permissions assigned to the user, represented as tags.
 - **Actions:** Provides options to edit or delete the user.
 - *Pagination:*
 - **Position:** Bottom of the table.
 - **Function:** Allows navigation through multiple pages of users. The number of entries per page can be adjusted using a dropdown.

Component: Actions Column

- *Position:* Rightmost column of the user table.
- *Structure:*
 - **Edit Button:** A pencil icon that opens the "Edit User" modal, allowing the admin to modify user details and permissions.
 - **Delete Button:** A trash icon that allows the admin to delete the user from the system.
- *Interaction:*
 - **Edit Button:** Clicking this button opens the "Edit User" modal where the admin can change the user's name, email, contact number, and permissions.
 - **Delete Button:** Clicking this button prompts the admin to confirm the deletion of the user. If confirmed, the user is removed from the `user_list` table.

Component: Search Box

- *Position:* Top right corner of the user table.
- *Function:* Allows admins to search for users by name, email, or contact number.
- *Interaction:* Typing into the search box filters the table to show only users matching the search criteria.

2.2. Add User Modal

Component: Add User Modal

- *Trigger:* Activated by clicking the "+ Add User" button located in the top right corner of the User Management screen.
- *Structure:*
 - *Fields:*
 - **User Name:** Input field for the user's full name.

- **Email:** Input field for the user's email address.
 - **Contact Number:** Input field for the user's contact number.
- Buttons:
 - **Close:** Closes the modal without saving changes.
 - **Save:** Saves the new user to the `user_list` table in the database.
- Interaction:*
 - **Save Button:** When clicked, the system validates the input fields and then adds the new user to the database. The new user is displayed in the user table upon successful addition.

2.3. Edit User Modal

Component: Edit User Modal

- *Trigger:* Activated by clicking the Edit button in the Actions column for a specific user.
- *Structure:*
 - Fields:
 - **User Name:** Input field pre-populated with the current user's name.
 - **Email:** Input field pre-populated with the current user's email address.
 - **Contact Number:** Input field pre-populated with the current user's contact number.
 - **User Permissions:** A dropdown field that allows the admin to add or remove permissions for the user. The dropdown is populated with available permissions, and only those permissions that the user does not already have are shown in the dropdown.
 - Buttons:
 - **Close:** Closes the modal without saving changes.
 - **Save:** Updates the user's information and permissions in the `user_list` table.
 - Interaction:
 - **Adding Permissions:** Admins can start typing in the permissions field to filter and select additional permissions. Selected permissions are added as tags in the field and simultaneously updated in the `permissions` column in the database.
 - **Removing Permissions:** Each permission tag has an "x" button that, when clicked, removes the permission from the user. This action also updates the `permissions` column in the database.

3. Functional Requirements

4.1. User Management

- *Adding a New User:*
 - **Trigger:** Clicking the "+ Add User" button.
 - **Input Fields:** The admin must fill in the user's name, email, and contact number.
 - **Validation:** The system checks that the email is unique and that all required fields are filled.
 - **Database Update:** Upon clicking "Save," a new record is created in the `user_list` table with the provided details. The `permissions` column is initially empty unless specified during editing.

- *Editing an Existing User:*
 - **Trigger:** Clicking the Edit button for a user.
 - **Fields:** The admin can modify the user's name, email, contact number, and permissions.
 - **Permission Management:** Permissions are edited by adding or removing tags in the dropdown field. The system updates the permissions column in the user_list table to reflect these changes.
 - **Database Update:** Upon clicking "Save," the user's record in the user_list table is updated with the new details and permissions.
- *Deleting a User:*
 - **Trigger:** Clicking the Delete button for a user.
 - **Confirmation:** The system prompts the admin to confirm the deletion.
 - **Database Update:** Upon confirmation, the user is removed from the user_list table.

4.2. Permission Management

- *Adding Permissions:*
 - **Trigger:** Permissions are added in the Edit User modal by typing in the permissions field and selecting from the dropdown.
 - **Database Update:** The selected permission name is appended to the permissions column for the user, with multiple permissions separated by commas.
- *Removing Permissions:*
 - **Trigger:** Permissions are removed in the Edit User modal by clicking the "x" on the permission tag.
 - **Database Update:** The corresponding permission name is removed from the permissions column for the user in the user_list table.

4.3. User Search and Filtering

- *Search Functionality:*
 - **Trigger:** Typing in the Search box filters the displayed users in the table.
 - **Database Query:** The search function queries the user_list table, filtering results based on user name, email, or contact number.

User List

[+ Add user](#)

Show 10 entries

Search:

USER	EMAIL	CONTACT NUMBER	PERMISSION	ACTIONS
Amy Davis	amydavis@laurel.com	+17894562301	Create Order	
Chloe Rodriguez	chloerodriguez@laurel.com	+17654329870	Check In Maintain Master Data	
David Miller	davidmiller@laurel.com	+15678901234	Check In	
Emily Johnson	emilyjohnson@laurel.com	+19876543210	Pick Up	
Emma Wilson	emmawilson@laurel.com	+13456789012	Maintain Master Data Apan	
Isabella Brown	isabellabrown@laurel.com	+19876543210	Stock Adjustment Maintain Master Data	
James Peter	jamespeter@laurel.com	+18245456456	Stock Adjustment Create Order	
John Williams	johnwilliams@laurel.com	+12098675432	Maintain Master Data	
Liam Martinez	liammartinez@laurel.com	+19876543210	Maintain Master Data	
Michael Brown	michaelbrown@laurel.com	+16504321098	Stock Adjustment Create Order	

Showing 1 to 10 of 15 entries

[Previous](#)
[1](#)
[2](#)
[Next](#)

User List

[+ Add user](#)

Show 10 entries

Search:

USER	EMAIL	CONTACT NUMBER	PERMISSION	ACTIONS
Amy Davis	amydavis@laurel.com	+17894562301	Create Order	
Chloe Rodriguez	chloerodriguez@laurel.com	+17654329870	Check In Maintain Master Data	
David Miller	davidmiller@laurel.com			
Emily Johnson	emilyjohnson@laurel.com			
Emma Wilson	emmawilson@laurel.com		Maintain Master Data Apan	
Isabella Brown	isabellabrown@laurel.com		Stock Adjustment Maintain Master Data	
James Peter	jamespeter@laurel.com		Stock Adjustment Create Order	
John Williams	johnwilliams@laurel.com		Maintain Master Data	
Liam Martinez	liammartinez@laurel.com		Maintain Master Data	
Michael Brown	michaelbrown@laurel.com		Stock Adjustment Create Order	

Showing 1 to 10 of 15 entries

[Previous](#)
[1](#)
[2](#)
[Next](#)

Add User

USER NAME:

Amy Davis

EMAIL:

amydavis@laurel.com

CONTACT NO:

+17894562301

[Close](#)
[Save](#)

Page Name: Workflow

User List

[+ Add user](#)

Show 10 entries

Search:

USER	EMAIL	CONTACT NUMBER	PERMISSION	ACTIONS
Amy Davis	amydavis@laurel.com	+17894562301	Create Order	
Chloe Rodriguez	chloerodriguez@laurel.com	+17654329870	Check In Maintain Master Data	
David Miller	davidmiller@laurel.com	+15678901234	Check In	
Emily Johnson	emilyjohnson@laurel.com	+19876543210	Pick Up	
Emma Wilson	emmawilson@laurel.com	+13456789012	Maintain Master Data Apan	
Isabella Brown	isabellabrown@laurel.com	+19876543210	Stock Adjustment Maintain Master Data	
James Peter	jamespeter@laurel.com	+18245456456	Stock Adjustment Create Order	
John Williams	johnwilliams@laurel.com	+12098675432	Maintain Master Data	
Liam Martinez	liammartinez@laurel.com	+19876543210	Maintain Master Data	
Michael Brown	michaelbrown@laurel.com	+16504321098	Stock Adjustment Create Order	

Showing 1 to 10 of 15 entries

[Previous](#)
[1](#)
[2](#)
[Next](#)

1. Purpose:

The Workflow Permissions page is designed for administrators to manage user roles and permissions related to the rental workflow process. This interface allows admins to assign users to specific stages of the rental workflow, ensuring that only authorized personnel receive relevant notifications and can

act on them. When access is granted or removed for a task, the changes are reflected in the user_list database table by updating the permissions column, which stores multiple role names separated by commas.

2. UI Components

Component: Top Bar

- *Position:* Across the top of the interface.
- *Items:*
 - *Search Box:* Allows for searching within workflows or users.
 - *User Profile:* Displayed in the top right corner with the user's email and status (e.g., "Available").
 - *Logo:* Positioned at the top left corner, providing branding for the application.

2.1. Workflow Permissions Panel

Component: Workflow Permissions Panel

- *Position:* Central and most prominent area of the screen.
- *Structure:*
 - *Task List:* A vertical list of predefined tasks representing stages in the rental process.
 - *Assignment Columns:*
 - *Task Name Column:*
 - Content: Task names are displayed in bold (e.g., "Create Order", "Create Delivery", "Inspection").
 - Width: Fixed width to maintain alignment with the assignment dropdowns.
 - *Assignment Box Column:*
 - Content: Each row contains a dropdown field where assigned users are displayed as tags.
 - Width: Flexible to accommodate various screen sizes.
- *Task Rows:*
 - *Appearance:*
 - Each row represents a distinct task in the rental workflow.
 - Task Background: Alternating row colors or subtle borders to distinguish between different tasks.

- *Assigned Users:* Users are displayed as tags within the dropdown field.
- *Interactions:*
 - *Adding Users:*
 - Clicking within the dropdown field opens an input area where the admin can start typing a user's name.
 - Autocomplete: As the admin types, a dropdown appears showing only users from the `user_list` database table who are marked as active (`active = True`).
 - Selection: The admin can select a user from this list, and the user is then added as a tag within the dropdown. Upon selection, the corresponding permission name (representing the task) is added to the `permissions` column for that user in the `user_list` table.
 - *Removing Users:*
 - Each user tag has an "x" button that, when clicked, removes the user from the task assignment. Upon removal, the corresponding permission name is deleted from the `permissions` column for that user in the `user_list` table.

2.2. User Assignment Tags

Component: User Tags

- *Appearance:*
 - **Tag Color:** Default to purple for all user tags, with white text for contrast.
 - **Tag Shape:** Rounded corners for a modern, polished look.
- *Interaction:*
 - **Adding Tags:** Typing in the dropdown field triggers an autocomplete, and selecting a user from the dropdown adds them as a tag. This action also updates the `permissions` column for the user in the database to include the task name.
 - **Removing Tags:** Each tag has a clickable "x" to remove the user from the assignment. This action also updates the `permissions` column for the user in the database to remove the task name.
- *Dynamic Updating:*
 - Changes (additions/removals) are saved immediately, ensuring the current state reflects the actual workflow assignments and corresponding updates in the `user_list` database.

3. Functional Requirements

3.1. Role Assignment Mechanism

Adding Users to a Task:

- *Trigger:* Clicking within the dropdown field of a task row.

- *User Search and Selection:*
 - **Real-time Search:** As the administrator types a user's name, the system queries the backend to retrieve matching users from the `user_list` database table.
 - **Filter:** Only users who are marked as active (`active = True`) in the `user_list` table are displayed in the dropdown.
 - **Autocomplete Dropdown:** The dropdown shows only active users. No other users are listed or can be added.
 - **User Selection:** Clicking a user from the dropdown adds them to the task, immediately creating a visual tag in the assignment field. Simultaneously, the task name (representing the permission) is appended to the `permissions` column for the user in the `user_list` table, with multiple permissions separated by commas.
- *Validation:*
 - **Active Check:** The system ensures that only active users (`active = True`) from the `user_list` can be added to the task.
 - **Duplicate Check:** Prevents the same user from being assigned multiple times to the same task.
- *Immediate Update:*
 - **Backend Sync:** The update is instantly reflected in the backend, ensuring no delays or discrepancies in workflow management. This also triggers the system to add the task name (representing the permission) to the `permissions` column for the user in the `user_list` table.

Removing Users from a Task:

- *Trigger:* Clicking the "x" on a user tag.
 - *Immediate Update:*
 - **Backend Sync:** The removal is instantly reflected in the backend, ensuring no delays or discrepancies in workflow management. This also triggers the system to remove the task name (representing the permission) from the `permissions` column for the user in the `user_list` table.

3.2. Task Management

- *Task List Structure:*
 - **Static Task List:** The tasks listed on the Workflow Permissions page are predefined and cannot be modified by the user.
 - **Task Order:** The tasks are displayed in the order they occur in the rental process, from initiation (e.g., "Create Order") to finalization (e.g., "Inspection").
 - **Expandable Rows (Future Consideration):** Each task row could be expanded to show additional details, such as the number of users assigned or a brief task description.

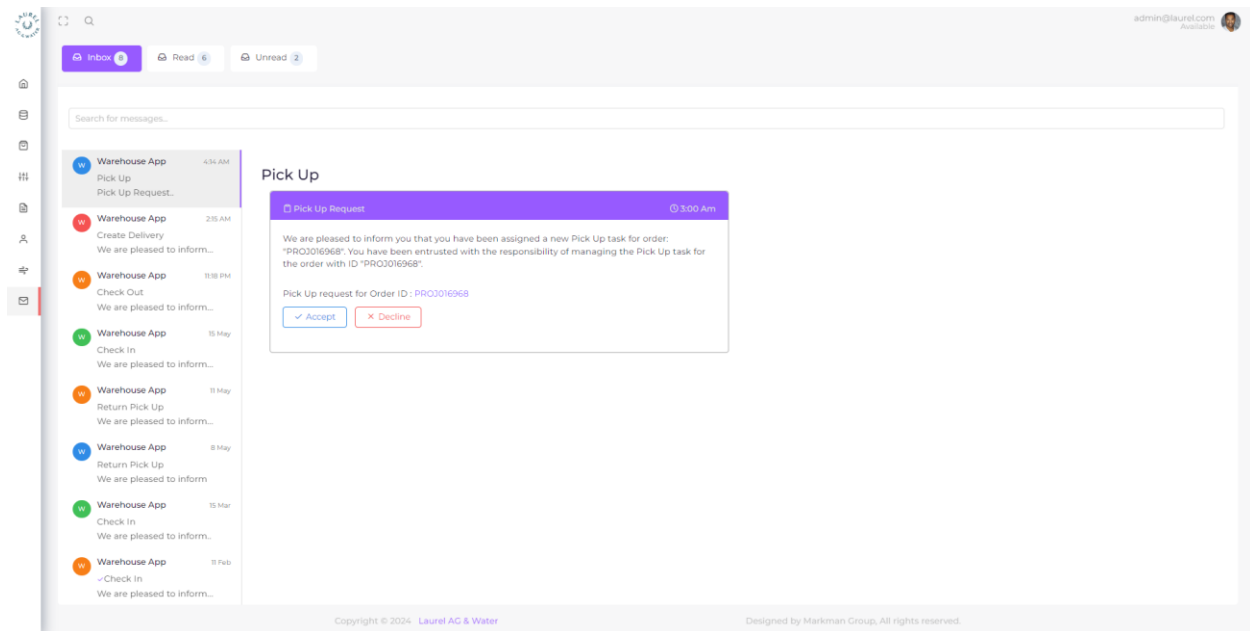
3.3. User Search and Autocomplete

- *Search Input:*
 - **Real-time Query:** As the user types, each keystroke triggers a query to the backend to retrieve matching users from the `user_list` table in the database.
 - **Filter:** The system only retrieves and displays users who are marked as active (`active = True`).
- *Autocomplete Dropdown:*
 - **Appearance:** The dropdown appears directly beneath the input field, overlaying any tags that may already exist.
 - **Content:** Displays user names along with additional identifiers like department or role if necessary to avoid confusion.
 - **Interaction:** Navigable by keyboard (arrow keys and Enter) and mouse.

3.4. Access Control

- *Admin-Only Access:*
 - **Restrictions:** Only users with the admin role can access the Workflow Permissions page and make changes.
 - **Access Validation:** Before displaying the page, the system checks the user's role to ensure they have administrative privileges.
 - **Error Handling:** If a non-admin user attempts to access the page, they are redirected to the home screen or shown an access denied message.

Page Name: Inbox



1. Purpose

The Inbox page is designed to efficiently manage and respond to various rental tasks communicated through system-generated messages. This interface enables users to access and act on messages related to different types of requests, each tailored to specific operational needs, ensuring a seamless workflow and enhancing user engagement with actionable insights.

4. Description of UI Components

The UI components maintain the design structure focusing on navigation, message listing, and detail viewing, all of which are crucial for efficient task management and user interaction.

1.2 Message List Panel

Component: Message List

- *Position*: Left side of the main content area.
- *Content*: A list of messages, each displaying:
 - *Icon*: An icon indicating the message type (e.g., delivery, pickup).
 - *Subject Line*: First line of the message (not full).
 - *Timestamp*: The time or date the message was received.
 - *Inbox/Read/Unread*: Option to filter messages
 - *Search*: A search box to search through the inbox e.g., by entering order id, etc.

1.3 Message Detail View

Component: Message Detail

- *Position*: Right side of the main content area.
- *Content*: Detailed information about the selected message, including:
 - *Header*: Message title and timestamp.
 - *Body*: Detailed message content.
 - *Action Buttons*: Relevant buttons like "Create Order", "Accept" etc. based on the message type.
 - *Hyperlinks*: Hyperlinked data such as Order IDs for quick navigation to detailed screens.

5. Message Types and User Interactions

2.1 Create Delivery Request

- *Purpose*: Initiates the creation of a new delivery order.
- *When*: Immediately upon order creation, with reminders sent 30 and 15 calendar days before the contract start date.
- *Control*: Notification not sent if deliveries created for full order amount/quantity.
- *Message Content*:
 - "Hello! A new delivery task awaits your expertise. Please review and create a delivery order for Order ID: [Hyperlinked Order ID]. We rely on your prompt action to keep things moving smoothly."
- *Action Buttons*:
 - *Create Delivery Order*: Directs the user to a form for creating a new delivery order. This action does not prefill any details, encouraging complete user-driven entry to ensure accuracy and detail adherence.
- *Hyperlinks*:
 - *Order ID*: Direct navigation to the order's detailed view for full context before proceeding.

2.2 Pick Up Request

- *Purpose*: Alerts the user to manage a pickup of goods.
- *When*: Immediately upon delivery order creation, with reminders sent 5 calendar days before, as well as on the contract start date.
- *Control*: Notification not sent if step completed.
- *Message Content*:
 - "You have been selected to manage a pickup for Delivery ID: [Hyperlinked Delivery ID]. Please confirm your availability and prepare to execute this task."
- *Action Buttons*:
 - *Accept*: Completes user consent to proceed, loading the pickup ticket step with all necessary details prefilled for quick action.

- *Review*: Navigate user to pick up stage without prefilled details.
- *Hyperlinks*:
 - *Delivery ID*: Direct navigation to the delivery's detailed view for full context before proceeding.

2.3 Check Out Request

- *Purpose*: Facilitates the checkout process for items ready for dispatch.
- *When*: Immediately upon delivery order creation, with reminders sent 5 calendar days before the contract start date, as well as on the contract start date.
- *Control*: Notification not sent if step completed.
- *Message Content*:
 - "A checkout task requires your attention for Delivery ID: [Hyperlinked Delivery ID]. Please ensure all items are accounted for and proceed to confirm the checkout."
- *Action Buttons*:
 - *Accept*: Navigates to a checkout screen with item details prefilled, ready for verification and confirmation.
 - *Review*: Navigate user to check out stage without prefilled details.
- *Hyperlinks*:
 - *Delivery ID*: Direct navigation to the delivery's detailed view for full context before proceeding.

2.4 Delivered to Customer Task

- *Purpose*: Confirms the successful delivery of items to a customer.
- *When*: Immediately upon delivery order creation, with reminders sent 5 calendar days before the delivery date, and again on the contract start date.
- *Control*: Notification not sent if step completed.
- *Message Content*:
 - "Please confirm the successful delivery of items for Delivery ID: [Hyperlinked Delivery ID]. Your confirmation is vital for our records and customer satisfaction."
- *Action Buttons*:
 - *Accept*: Leads to a confirmation screen with delivery details prefilled for final verification.
 - *Review*: Navigate user to delivered to customer stage without prefilled details.
- *Hyperlinks*:
 - *Delivery ID*: Direct navigation to the delivery's detailed view for full context before proceeding.

2.5 Create Return Deliveries Request

- *Purpose*: Manages the initiation of return deliveries from customers.
- *When*: Starting 30 and 15 calendar days before the contract end date.
- *Control*: Notification not sent if step completed.

- *Message Content:*
 - "A return delivery task has been initiated for Delivery ID: [Hyperlinked Delivery ID]. Please create the necessary documentation to process this return effectively."
- *Action Buttons:*
 - *Create Return Delivery:* Directs the user to a form for creating a new delivery order. This action does not prefill any details, encouraging complete user-driven entry to ensure accuracy and detail adherence.
 - *Review:* Navigate user to return delivery stage without prefilled details.
- *Hyperlinks:*
 - *Delivery ID:* Direct navigation to the delivery's detailed view for full context before proceeding.

2.6 Return Pickup Request

- *Purpose:* Manages the pickup of returned items from a customer.
- *When:* Five calendar days before and on the contract end date.
- *Control:* Notification is not sent if the step is completed and the return delivery is directed to the customer (refer to the rental process).
- *Message Content:*
 - "You are tasked with overseeing the pickup of returned items for Delivery ID: [Hyperlinked Delivery ID]. Please prepare and confirm when you are ready to proceed."
- *Action Buttons:*
 - *Accept:* Proceeds with a detailed task screen, prefilled with necessary pickup details to facilitate quick action.
 - *Review:* Navigate user to return pick up stage without prefilled details.
- *Hyperlinks:*
 - *Delivery ID:* Direct navigation to the delivery's detailed view for full context before proceeding.

2.7 Check In Request

- *Purpose:* Ensures the proper check-in of items into warehouse inventory.
- *When:* Five calendar days before and on the contract end date.
- *Control:* Notification is not sent if the step is completed and the return delivery is directed to the customer (refer to the rental process).
- *Message Content:*
 - "A check-in request has been issued for Delivery ID: [Hyperlinked Delivery ID]. Please verify all items and confirm their storage in our system."
- *Action Buttons:*
 - *Accept:* Navigates to a check-in screen with all relevant details prefilled for efficient processing.
 - *Review:* Navigate user to return check in stage without prefilled details.
- *Hyperlinks:*

- *Delivery ID*: Direct navigation to the delivery's detailed view for full context before proceeding.

2.8 Inspection Request

- *Purpose*: Calls for the inspection of items either returned or recently delivered.
- *When*: Five calendar days before and on the contract end date.
- *Control*: Notification is not sent if the step is completed and the return delivery is directed to the customer (refer to the rental process).
- *Message Content*:
 - "Your expertise is required for an inspection task concerning Delivery ID: [Hyperlinked Delivery ID]. Please review the items and provide your findings."
- *Action Buttons*:
 - *Accept*: Directs to an inspection detail screen with necessary information prefilled to assist in the inspection process.
 - *Review*: Navigate user to return inspection stage without prefilled details.
- *Hyperlinks*:
 - *Delivery ID*: Direct navigation to the delivery's detailed view for full context before proceeding.

Appendix



Rental Process
Laurel.pdf



rental app.docx



Attribute
Mapping.xlsx