### **API Documentation: Get Inventory**

#### **Route: /inventory/**

* **Method**: GET
* **Summary**: This endpoint retrieves filtered inventory data from the database based on optional query parameters. It supports dynamic filtering, pagination, and date range filtering for different device and user attributes, such as device model, serial number, user job title, executive status, and more.

#### **Libraries Used:**

1. **FastAPI**: The web framework used to define the route, handle HTTP requests, and automatically generate API documentation.
2. **SQLAlchemy**: ORM (Object Relational Mapping) library used to query the database and perform filtering, counting, and pagination operations on the inventory data.
3. **Pydantic**: For input validation and type checking of request parameters.
4. **Datetime**: To handle and manipulate date values for date range filtering (start\_date, end\_date, device\_receiveddate, etc.).
5. **SQLAlchemy cast & Date**: Used to cast date columns to Date type to facilitate date filtering and comparisons.

#### **Query Parameters:**

* **item\_id** (Optional): Filter inventory by item ID.
* **device\_name** (Optional): Filter inventory by device name.
* **device\_serialnumber** (Optional): Filter inventory by device serial number.
* **device\_type** (Optional): Filter inventory by device type.
* **device\_model** (Optional): Filter inventory by device model.
* **user\_jobtitle** (Optional): Filter inventory by user job title.
* **user\_email** (Optional): Filter inventory by user email.
* **user\_firstname** (Optional): Filter inventory by user first name.
* **user\_lastname** (Optional): Filter inventory by user last name.
* **user\_associateid** (Optional): Filter inventory by user associate ID.
* **user\_branch** (Optional): Filter inventory by user branch.
* **user\_branchoffice** (Optional): Filter inventory by user branch office.
* **user\_city** (Optional): Filter inventory by user city.
* **user\_state** (Optional): Filter inventory by user state.
* **user\_executive** (Optional): Filter inventory by user executive status (Yes/No).
* **user\_shortdept** (Optional): Filter inventory by user short department.
* **device\_shortfmfg** (Optional): Filter inventory by device short manufacturer.
* **device\_laptoptype** (Optional): Filter inventory by device laptop type.
* **device\_receiveddate** (Optional): Filter inventory by device received date.
* **device\_age** (Optional): Filter inventory by device age.
* **device\_yearreceived** (Optional): Filter inventory by device year received.
* **device\_yearrefresh** (Optional): Filter inventory by device year refresh.
* **start\_date** (Optional): Start date for filtering the date range.
* **end\_date** (Optional): End date for filtering the date range.
* **page** (Optional): Page number for pagination (default is 1).
* **per\_page** (Optional): Number of items per page (default is 10, max 100).
* **days** (Optional): Days range to filter inventory based on End-of-Life (EOL) or lease end date.

#### **Response:**

* **inventory**: A list of inventory items matching the query filters.
* **total\_count**: The total number of matching inventory items.
* **total\_pages**: Total number of pages based on the pagination settings.
* **current\_page**: The current page of results.
* **per\_page**: The number of items per page.

#### **Features:**

1. **Dynamic Filtering**: Apply filters on various attributes like device name, serial number, user details, device age, etc.
2. **Pagination**: Results are paginated to manage large data sets, with configurable page size.
3. **Date Range Filtering**: Filter inventory based on device end-of-life (EOL) or lease end date using start\_date and end\_date.
4. **Executive Filter**: Filter results based on user executive status (Yes or No).
5. **Flexible Date Parameters**: Optionally filter inventory by received date, year received, year refresh, and days range for EOL or lease end dates.

#### **Example Request:**

GET /inventory/?device\_name=laptop&user\_city=New%20York&start\_date=2023-01-01&end\_date=2023-12-31&page=2&per\_page=10

#### **Example Response:**

{

"inventory": [

{

"item\_id": "123",

"device\_name": "Laptop A",

"device\_model": "XYZ",

"user\_city": "New York",

"device\_receiveddate": "2023-05-01"

},

{

"item\_id": "124",

"device\_name": "Laptop B",

"device\_model": "ABC",

"user\_city": "New York",

"device\_receiveddate": "2023-06-01"

}

],

"total\_count": 50,

"total\_pages": 5,

"current\_page": 2,

"per\_page": 10}

### **API Documentation: Fetch Distinct Data**

#### **Route: /fetch\_distinct\_data**

* **Method**: GET
* **Summary**: This endpoint fetches distinct values for various inventory and user-related attributes, such as device type, model, branch, city, state, executive status, and more. It provides a list of unique values for each attribute, which can be useful for generating filters or dropdowns in the user interface.

#### **Libraries Used:**

1. **FastAPI**: The web framework used to define the route, handle HTTP requests, and automatically generate API documentation.
2. **SQLAlchemy**: ORM library used to query the database and retrieve distinct values from multiple columns of the inventory data.
3. **SQLAlchemy select and func.distinct**: Used to retrieve distinct values from the specified columns of the Inventory table.
4. **SQLAlchemy filter**: Used to ensure that None values are excluded from the distinct results.
5. **defaultdict**: A dictionary subclass from the Python collections module, used to automatically initialize list values for each attribute.

#### **Response Model:**

* **response\_model**: Dict[str, List[str]] - A dictionary with attribute names as keys and lists of distinct values as the corresponding values.

#### **Response Structure:**

The response will be a JSON object where each key represents an inventory or user attribute, and the associated value is a list of distinct values for that attribute. Below are the fields returned:

* **device\_type**: List of distinct device types.
* **device\_model**: List of distinct device models.
* **user\_shortdept**: List of distinct user short department names.
* **device\_shortmfg**: List of distinct device manufacturers.
* **device\_laptoptype**: List of distinct laptop types.
* **user\_branch**: List of distinct user branches.
* **user\_branchoffice**: List of distinct user branch office names.
* **user\_city**: List of distinct user cities.
* **user\_state**: List of distinct user states.
* **device\_receiveddate**: List of distinct device received dates.
* **device\_age**: List of distinct device ages.
* **device\_yearreceived**: List of distinct device years received.
* **device\_yearrefresh**: List of distinct device years for refresh.
* **user\_executive**: List of distinct values for user executive status, where True is mapped to "Yes" and False is mapped to "No".

#### **Example Response:**

{

"device\_type": ["Laptop", "Desktop", "Tablet"],

"device\_model": ["XPS 13", "MacBook Pro", "Surface Pro"],

"user\_shortdept": ["IT", "HR", "Finance"],

"device\_shortmfg": ["Dell", "Apple", "Microsoft"],

"device\_laptoptype": ["Ultrabook", "Gaming"],

"user\_branch": ["NY", "LA", "SF"],

"user\_branchoffice": ["New York Office", "Los Angeles Office"],

"user\_city": ["New York", "San Francisco", "Los Angeles"],

"user\_state": ["NY", "CA"],

"device\_receiveddate": ["2023-01-01", "2023-06-15", "2024-02-01"],

"device\_age": ["1 year", "2 years", "3 years"],

"device\_yearreceived": ["2023", "2022", "2021"],

"device\_yearrefresh": ["2026", "2025", "2024"],

"user\_executive": ["Yes", "No"]

}

#### **Features:**

1. **Distinct Values Retrieval**: Fetches distinct values for a wide range of attributes related to devices and users.
2. **Excludes Null Values**: Ensures that None (null) values are excluded from the distinct lists.
3. **Executive Status Conversion**: Converts boolean values for user\_executive into more user-friendly values, such as "Yes" and "No".
4. **Grouped Data**: Returns a dictionary where each key is a column name, and the corresponding value is a list of sorted, distinct values.
5. **Data for Filtering**: Useful for generating filter options or dropdown menus based on available inventory and user data.

#### **Example Request:**

GET /fetch\_distinct\_data

#### **Example Response:**

{

"device\_type": ["Laptop", "Desktop", "Tablet"],

"device\_model": ["XPS 13", "MacBook Pro", "Surface Pro"],

"user\_shortdept": ["IT", "HR", "Finance"],

"device\_shortmfg": ["Dell", "Apple", "Microsoft"],

"device\_laptoptype": ["Ultrabook", "Gaming"],

"user\_branch": ["NY", "LA", "SF"],

"user\_branchoffice": ["New York Office", "Los Angeles Office"],

"user\_city": ["New York", "San Francisco", "Los Angeles"],

"user\_state": ["NY", "CA"],

"device\_receiveddate": ["2023-01-01", "2023-06-15", "2024-02-01"],

"device\_age": ["1 year", "2 years", "3 years"],

"device\_yearreceived": ["2023", "2022", "2021"],

"device\_yearrefresh": ["2026", "2025", "2024"],

"user\_executive": ["Yes", "No"]

}

### **API Documentation: Export Inventory to Excel**

#### **Route: /export\_inventory\_to\_excel**

* **Method**: GET
* **Summary**: This endpoint exports all the inventory data from the database into an Excel file. The data is retrieved from the Inventory table and converted into a DataFrame using the pandas library. If there is data in the table, it is exported to an Excel file and returned as a downloadable file.

#### **Libraries Used:**

1. **FastAPI**: The web framework used to define the route, handle HTTP requests, and stream the response.
2. **SQLAlchemy**: ORM library used to query the database and retrieve all inventory records from the Inventory table.
3. **pandas**: Used to handle the conversion of database query results into a DataFrame, which is then written to an Excel file.
4. **openpyxl**: Excel file writer engine used by pandas to generate .xlsx files.
5. **BytesIO**: Used to store the in-memory Excel file and stream it back to the user.

#### **Response Model:**

* **Response**: The API will return an Excel file in .xlsx format, which will be available for download. If no data is found, a message will be returned indicating that the Inventory table is empty.

#### **Response Structure:**

1. **File Download**: If data is available in the Inventory table, an Excel file (inventory\_data.xlsx) will be returned as a downloadable attachment with the correct MIME type for Excel.
2. **No Data Message**: If no records are found in the Inventory table, the response will be a JSON object with a message "No data found in the Inventory table.".

#### **Features:**

1. **Excel Export**: Exports all inventory data to a .xlsx Excel file.
2. **Data Conversion**: The data is converted into a pandas DataFrame for easy export to Excel.
3. **Streaming Response**: The file is returned as a streaming response, so the user can download it directly.
4. **Empty Table Check**: If no records are available in the Inventory table, a friendly message is returned.

#### **Example Request:**

GET /export\_inventory\_to\_excel

#### **Example Response:**

* **If inventory data is present**: An Excel file named inventory\_data.xlsx is generated and downloaded by the client.
* **If no data is available**: The response will be a JSON message.

{

"message": "No data found in the Inventory table."

}