Inward Assist Working

CREATE SLOT:

Step 1: User Action

- The user clicks the **"BOOK A SLOT"** button on the UI.

Step 2: API Call to Fetch Dashboard Details

- The UI triggers a call to the **`iav2/dashboard` API** of the **Inward Assist Service**.

Step 3: Check for Cached Data

1. The backend checks if the dashboard details are already cached.
2. **If cached data is available:**
    - Return the cached data directly to the UI.
3. **If no cached data is found:**
    - Populate the cache by making calls to external services.

Step 4: Populate Cache by Calling External Services

1. **Call to Switch Service**:
    - Fetch the warehouse IDs.
2. **Call to WMS Service**:
    - Retrieve warehouse details using the fetched warehouse IDs.
3. **Call to Switch Service Again**:
    - Fetch the business line and group ID.
4. **Call to Shifu Service**:
    - Fetch the business unit (BU) and brand.
5. **Group ID to Capacity Classification Mapping**:
    - Use the **Switch Service** values to map the group ID to the corresponding capacity classification.
6. **Update Cache**:
    - Cache the populated dashboard data.

Step 5: Return Dashboard Data

- Once all details are gathered and cached, the data is returned to the UI.

Step 6: Simultaneous Call to Fetch Purchase Orders (POs)

- While fetching the dashboard, the UI makes another call to the **`iav2/po` API** of the **Inward Assist Backend** to retrieve the list of POs for a vendor.

Step 7: Handling PO Search Criteria

1. **If searching by PO barcode:**
    - The barcode is sent as a query parameter.
2. **If searching by vendor ID only:**
    - The vendor ID is sent as the query parameter.

Step 8: PO Validation

1. Validate the PO barcodes.
2. Fetch the approved POs from the **PO Service**.

Step 9: PO Data Entry and Calculation

1. Create an entry for the POs with their details.
2. Calculate the **pending quantity** using a specific formula.

Step 10: Enriching PO Details

1. Call the **`enrichPODetails` method** to enrich the PO details.
2. **If the PO is fetched for the first time:**
    - Fetch details like **master category**, **brand**, **slotted quantity**, and **capacity classifications** from external services.
3. **If the PO is fetched again:**
    - Populate these details from the **poCacheHelper** cache.

Step 11: Fetch and Enrich Data for PO

1. Fetch all SKUs of the PO from the **PO Service**.

2. Fetch PO-to-Master Category mapping from the **`po_mastercategory_mapping` table**.
    - **If mapping is not found:**
        - Fetch the data from the **CMS Service**.
3. Fetch the business line.

Step 12: Remove Invalid POs

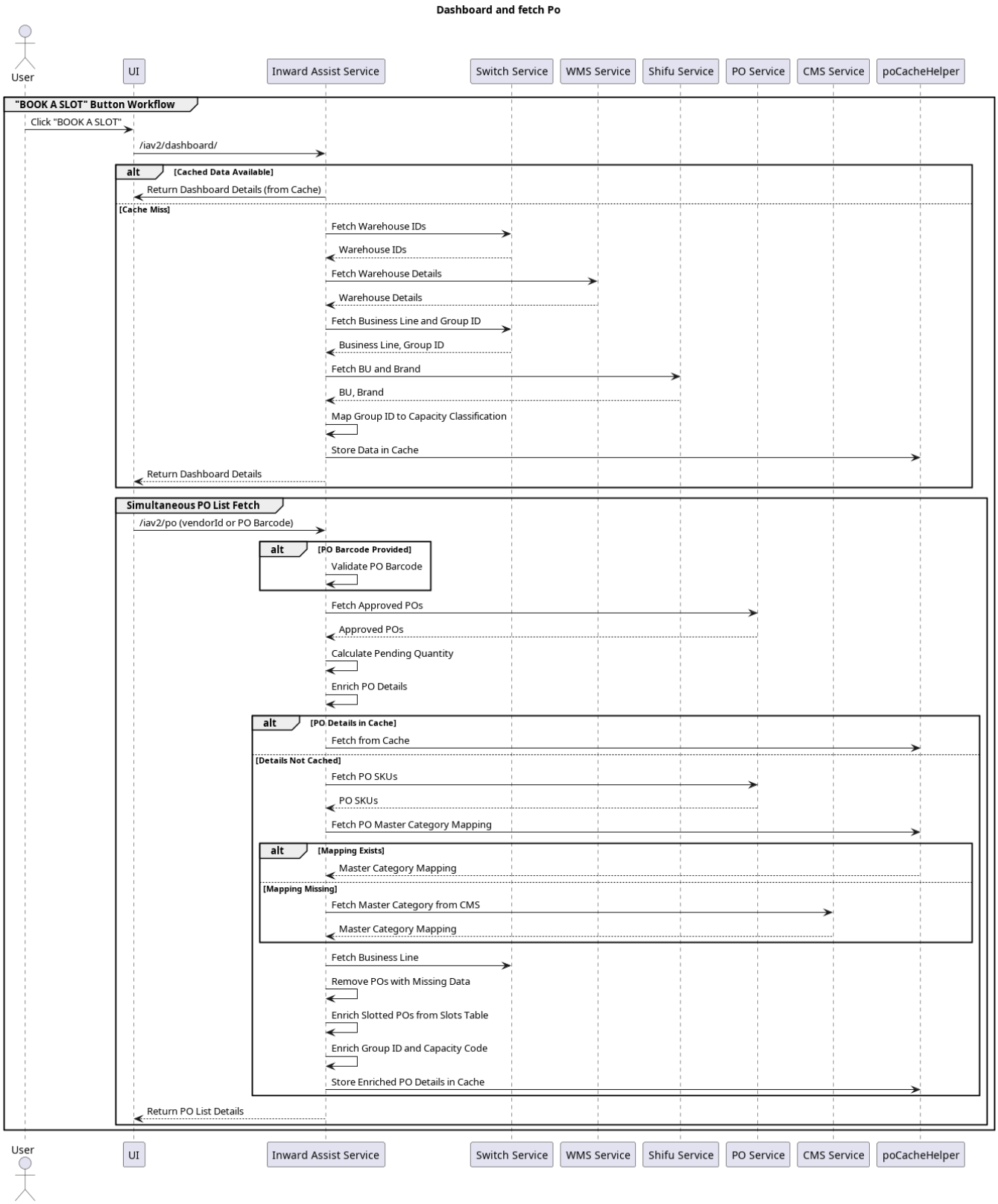
- Exclude POs that lack essential data.

Step 13: Enrich Slotted POs

1. Fetch slotted POs from the **slots table**.
2. Enrich details like **group ID** and **capacity code**.

Step 14: Update Cache and Return

1. Save the enriched details in the **poCacheHelper** cache for faster access in future requests.
2. Return the final enriched PO details to the dashboard.



SLOT BOOKING:

Step 1: User Action

- The user clicks on a PO from the dashboard page on the UI.

Step 2: Backend API Call

- The PO ID is passed to the backend via the **`/iav2/po` API** of the **Inward Assist Service**.
- **Key Parameter Passed**:
  - `isSiblingPONeeded` is set to **`true`**.

Step 3: Fetch PO Details

1. **If the PO is of SAMPLE_PO prioritization**:
   - Return details of only the selected PO.
2. **If the PO is not SAMPLE_PO prioritization**:
   - Fetch details of all **sibling POs** associated with the selected PO.

Step 4: Validate SAMPLE_PO Slotting

1. Check if the **SAMPLE_PO** associated with the selected PO was slotted more than **7 days ago**.
   - **If not slotted 7 days ago**:
     - Show an error on the UI prompting the user to slot the SAMPLE_PO first.
   - **If slotted 7 days ago**:
     - Remove the SAMPLE_PO from the list of sibling POs.

Step 5: Arrange and Return Enriched PO Details

1. Move the clicked PO to the **top of the sibling PO list**.
2. Enrich the PO details as needed.
3. Return the enriched PO list to the UI.

Step 6: User Proceeds to Book Slot

- The user fills in all the required details for booking slots for all sibling POs and clicks the **"Book Slot"** button.

Step 7: Backend Slot Booking API Call

- The backend triggers the **`/iav2/transaction/createSlot` API** of the **Inward Assist Service** to handle the slot booking.

Step 8: Log Transaction and Publish to Topic

1. Create an entry in the **`capacity_transaction` table** with the status set to **IN_PROGRESS**.
2. Create a job and publish the job details to the **`create_slot` topic** for processing.

Step 9: Consume the Message and Process Transaction

1. The message is consumed by the Inward Assist service.
2. Validate the transaction payload for:
   - **Date validation**.
   - **Sibling PO validation**.

Step 10: Handle Validation Failures

- If the transaction entry is invalid:
  - Mark the transaction as **FAILED** in the **`capacity_transaction` table**.

Step 11: Proceed with Slot Booking

- **If validation succeeds**, process the transaction by performing the following:

1. **Validate Capacity ID**:
   - Check if the capacity ID for the requested slot is valid.
2. **Check for Existing Slot for Invoice**:
   - If a slot already exists for the invoice:
     - Fail the job.
   - If no slot exists, proceed to the next step.
3. **Fetch Capacity Details**:
   - Retrieve capacity data from the **`capacity_storage` table**.
4. **Block Capacity**:
   - **If insufficient storage is available**:
     - Interrupt the job and fail the transaction.
   - **If sufficient storage is available**:
     - Block the required capacity and update the **`capacity_storage` table** by deducting the reserved capacity.

Step 12: Fetch PO Details

1. Fetch PO details from the **PO Service** using the PO ID.
2. **If PO does not exist**:
    - Interrupt the job and fail the transaction.
3. **If PO exists**:
    - Proceed to create a slot entry.

Step 13: Update Slot Entry

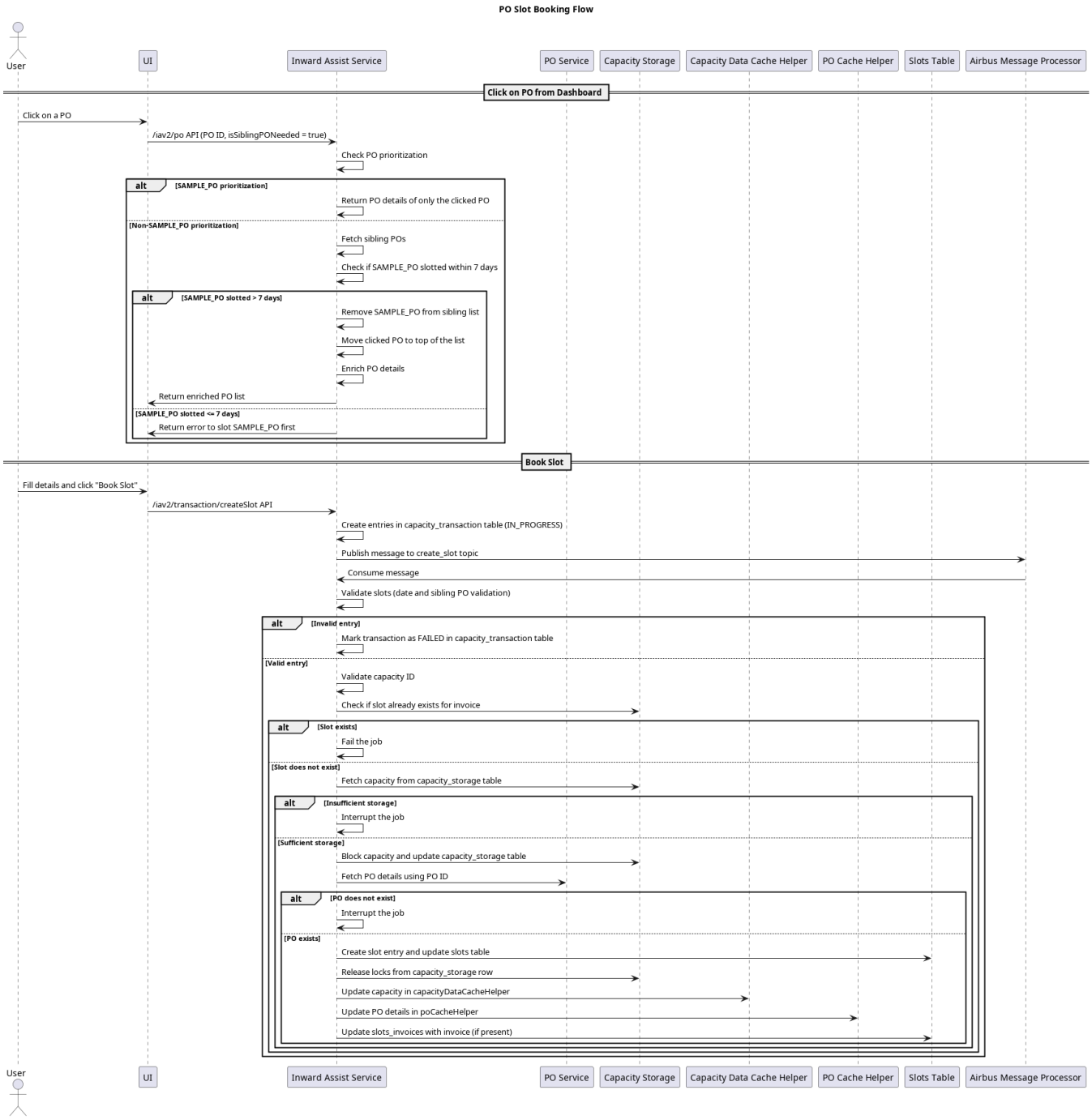
1. Create a slot entry in the **`slots` table**.
2. Update the database with the slot details.

Step 14: Release Locks and Update Caches

1. Release any locks on rows in the **`capacity_storage` table**.
2. Update the **`capacityDataCacheHelper`** with the new capacity details.
3. Update the **`poCacheHelper`** with the updated PO details for faster future access.

Step 15: Handle Invoice in Slot Request

- If the slot request includes an invoice, update the **`slots_invoices` table** with the invoice details.

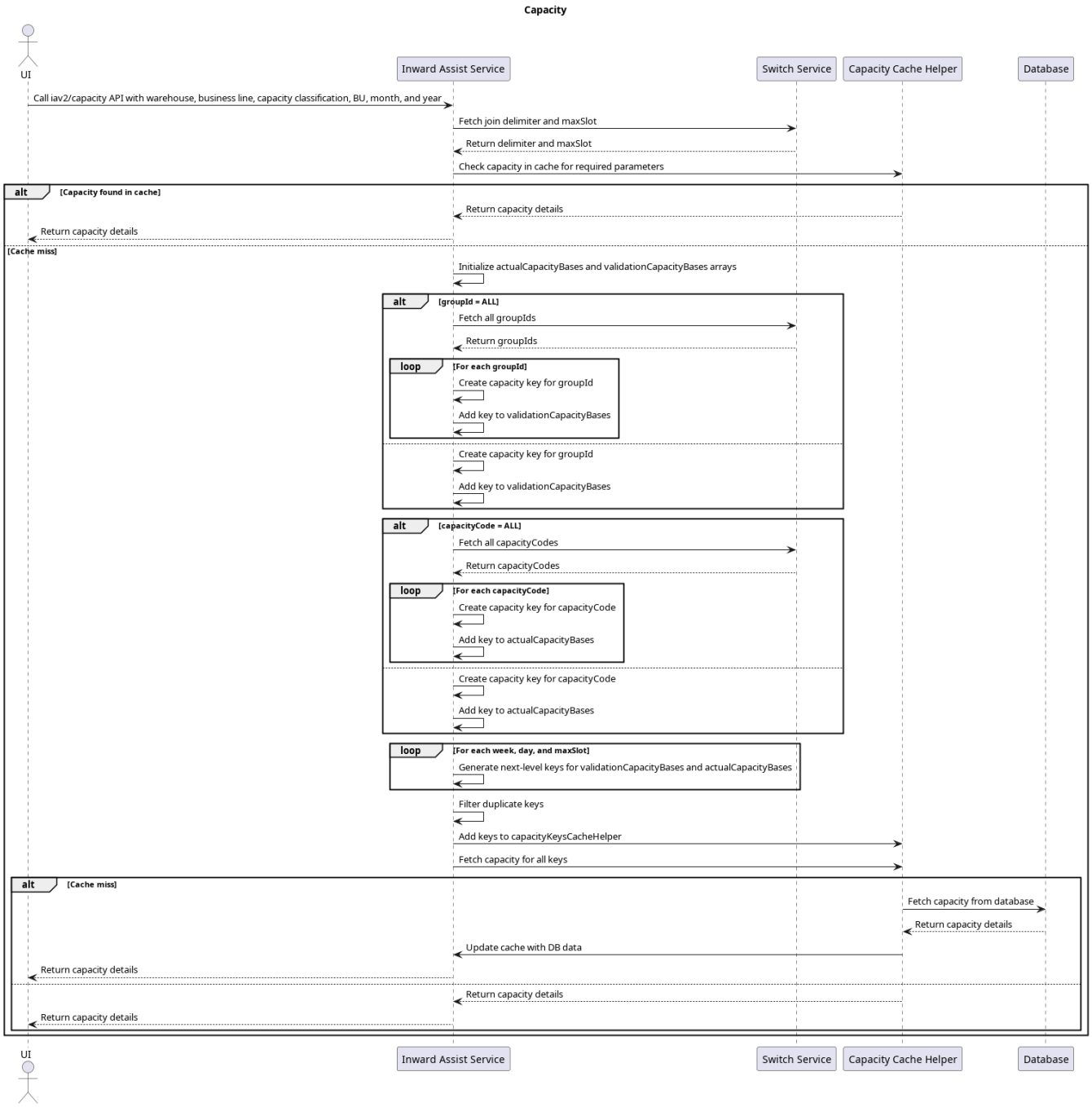


FETCHING CAPACITY DETAILS:

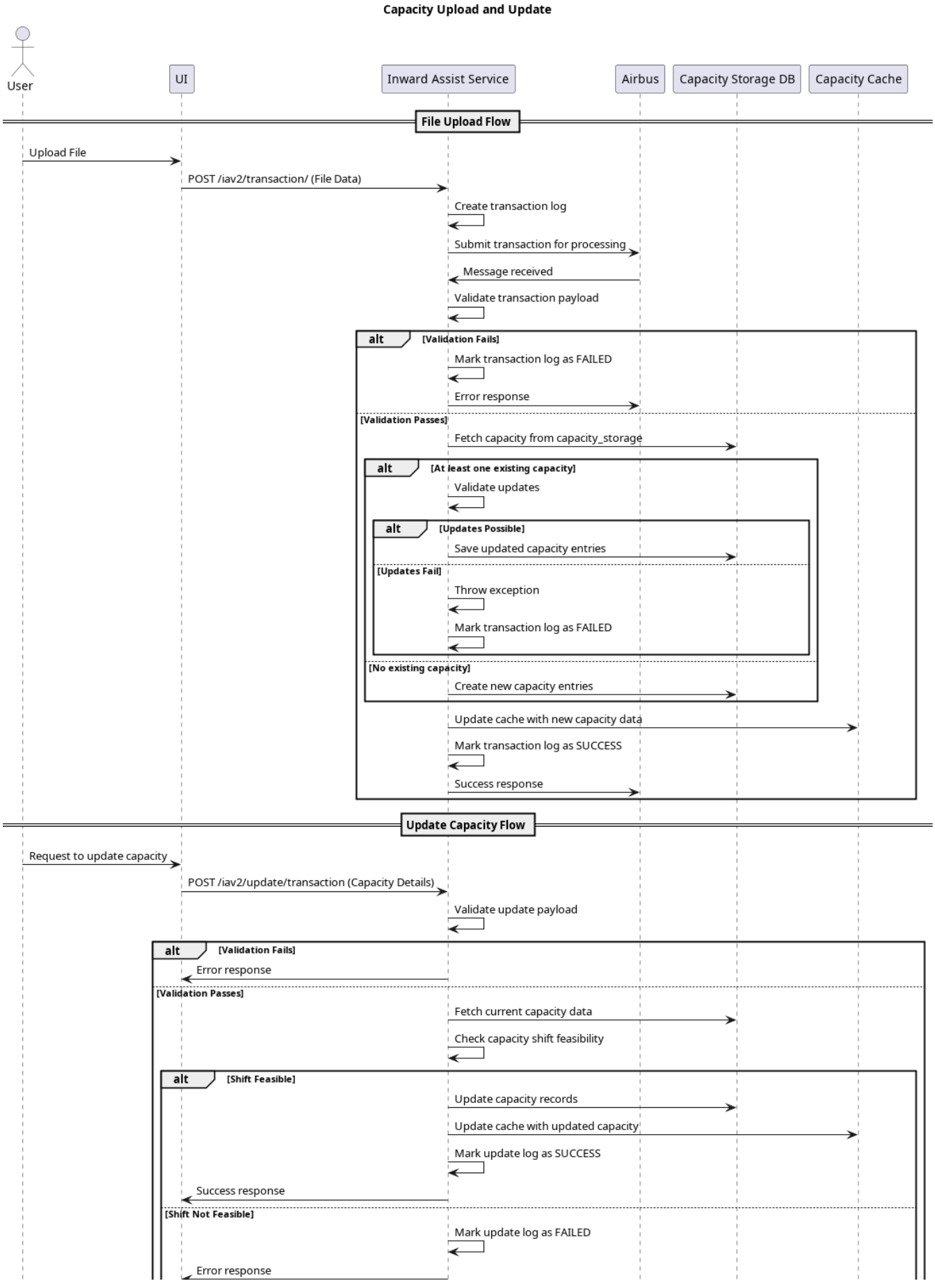
The **Inward Assist Service** provides two distinct APIs for fetching capacity details:

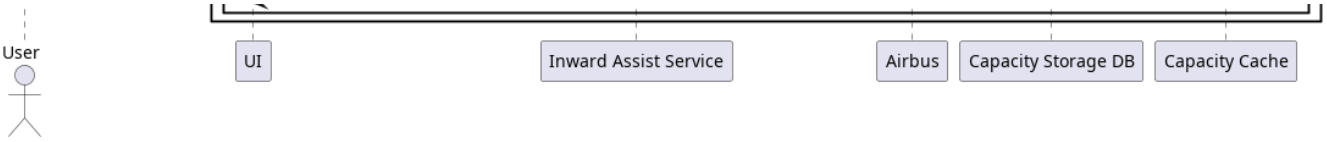
1. **Single Capacity Fetch API**
    - Designed to retrieve capacity for a single key at a time.
    - Suitable for scenarios where precise and specific capacity information is required.
2. **Bulk Capacity Fetch API**
    - Optimized for retrieving capacity for multiple keys in a single call.
    - Primarily used in scenarios like **sibling PO slotting**, where multiple POs need to be processed simultaneously.

By leveraging the **Bulk Capacity Fetch API**, the system efficiently handles batch processing, ensuring seamless performance even when dealing with numerous POs in parallel.



CAPACITY UPLOAD AND UPDATE:


Capacity Upload and Update
User
UI
Inward Assist Service
Airbus
Capacity Storage DB
Capacity Cache
File Upload Flow
Upload File
POST /iav2/transaction/ (File Data)
Create transaction log
Submit transaction for processing
Message received
Validate transaction payload
alt
[Validation Fails]
Mark transaction log as FAILED
Error response
[Validation Passes]
Fetch capacity from capacity_storage
alt
[At least one existing capacity]
Validate updates
alt
[Updates Possible]
Save updated capacity entries
[Updates Fail]
Throw exception
Mark transaction log as FAILED
[No existing capacity]
Create new capacity entries
Update cache with new capacity data
Mark transaction log as SUCCESS
Success response
Update Capacity Flow
Request to update capacity
POST /iav2/update/transaction (Capacity Details)
Validate update payload
alt
[Validation Fails]
Error response
[Validation Passes]
Fetch current capacity data
Check capacity shift feasibility
alt
[Shift Feasible]
Update capacity records
Update cache with updated capacity
Mark update log as SUCCESS
Success response
[Shift Not Feasible]
Mark update log as FAILED
Error response


User
UI
Inward Assist Service
Airbus
Capacity Storage DB
Capacity Cache