**Project: Monthly Billing Generator**

This project introduces a Python-based Monthly Billing Generator, designed to automate the billing process for a wide range of rented items. Given a list of items with start and stop dates, quantities, and rates, the system intelligently identifies which items are active during a specified target month and calculates their prorated quantities and corresponding revenue based on the number of active days.

By grouping similar items with the same billing attributes, the tool produces a clean, structured summary that includes line-by-line details and total revenue for the month. This ensures billing accuracy, operational efficiency, and transparency for both service providers and clients.

This solution is especially valuable for companies managing co-working spaces, leased assets, equipment rentals, or subscription-based products with variable usage durations.

**Objective:**

To generate a **monthly prorated bill** for a given set of rented items based on their active duration within a target month. The bill includes:

* Quantity used in the month (prorated if partial),
* Rate per item,
* Calculated amount,
* Billing period,
* Total revenue.

**Function Overview**

generate\_monthly\_bill(item\_list: list, target\_month: str) -> dict

**Parameters:**

* item\_list: List of items with their rental details (start\_date, stop\_date, qty, rate, etc.)
* target\_month: A string in "YYYY-MM" format for which billing is to be calculated.

**Returns:**

A dictionary containing:

* line\_items: List of billed items with prorated quantities and corresponding amounts.
* total\_revenue: Total revenue generated in the given month.

**Code Breakdown**

**1. Parse Target Month**

year, month = map(int, target\_month.split('-'))

Splits the input string (e.g., "2024-11") to extract the year and month as integers.

Code:

last\_day = calendar.monthrange(year, month)[1]

month\_start = date(year, month, 1)

month\_end = date(year, month, last\_day)

* Calculates the **first and last day** of the month.
* Needed to calculate the active billing duration for each item.

**2. Grouping Items by Code, Rate, and Active Period**

Code :

groups = {}

Used to **aggregate items** with the same item code, rate, and overlapping billing period.

Code:

for item in item\_list:

start\_date = datetime.strptime(item['start\_date'], '%Y-%m-%d').date()

stop\_date = datetime.strptime(item['stop\_date'], '%Y-%m-%d').date()

* Converts string dates in the items to actual date objects.

**Overlapping Period Check:**

Code:

overlap\_start = max(start\_date, month\_start)

overlap\_end = min(stop\_date, month\_end)

if overlap\_start > overlap\_end:

continue

* Determines the **actual active duration** of the item in the target month.
* Skips items that are not active during the month.

**Prorated Quantity Calculation:**

Code:

active\_days = (overlap\_end - overlap\_start).days + 1

days\_in\_month = (month\_end - month\_start).days + 1

qty = float(item['qty']) if isinstance(item['qty'], str) else float(item.get('qty', 0))

rate = float(item['rate']) if isinstance(item['rate'], str) else float(item.get('rate', 0))

prorated\_qty = qty \* active\_days / days\_in\_month

* Calculates the **portion of the quantity** applicable for the target month based on how many days it was active.
* Converts string values of qty and rate to float safely.

**3. Group Items by Common Attributes**

Code :

group\_key = (item\_code, rate, overlap\_start, overlap\_end)

Groups items with the same:

* Item Code,
* Rate,
* Active period within the month.

Code :

groups[group\_key] = {

'prorated\_qty\_sum': 0.0,

'item\_code': item\_code,

'rate': rate,

'overlap\_start': overlap\_start,

'overlap\_end': overlap\_end

}

groups[group\_key]['prorated\_qty\_sum'] += prorated\_qty

* Accumulates the prorated quantity per unique item group.

**4. Prepare Final Line Items**

Code :

line\_items = []

total\_revenue = 0.0

* Initializes the final bill details and total revenue.

Code :

qty = int(prorated\_qty\_sum) if prorated\_qty\_sum.is\_integer() else round(prorated\_qty\_sum, 2)

code :

line\_item = {

"item\_code": group['item\_code'],

"rate": group['rate'],

"qty": qty,

"amount": round(prorated\_qty\_sum \* group['rate'], 2),

"billing\_period": f"{group['overlap\_start'].strftime('%Y-%m-%d')} to {group['overlap\_end'].strftime('%Y-%m-%d')}"

}

* Each line item includes the item code, rate, prorated quantity, amount, and billing period.

Code :

total\_revenue += line\_item['amount']

* Accumulates total revenue for the month.

**5. Return Final Result**

Code :

return {

"line\_items": line\_items,

"total\_revenue": round(total\_revenue, 2)

}

* Outputs the finalized monthly bill.

**Sample Output for November 2024**

Calling the function:

Code :

result = generate\_monthly\_bill(item\_list, "2024-11")

print(json.dumps(result, indent=4))

Returns a JSON output like:

json

Copy code

{

"line\_items": [

{

"item\_code": "Executive Desk (4\*2)",

"rate": 1080.0,

"qty": 15,

"amount": 16200.0,

"billing\_period": "2024-11-01 to 2024-11-30"

},

...

],

"total\_revenue": 206565.0

}

This script automates the generation of prorated monthly billing for rented items based on their active duration within a specified target month. It processes a list of items with their start and stop dates, quantity, and rate, then calculates the portion of usage applicable for the given month.

For each item, the script:

* Identifies the overlap between the item's rental period and the target month.
* Computes the number of active days within the month.
* Calculates the prorated quantity and corresponding amount based on daily usage.
* Groups items with identical codes, rates, and active periods to consolidate entries.
* Produces a detailed line-item breakdown, showing item code, rate, prorated quantity, amount, and billing period.
* Computes the total revenue for the month.

The result is returned in a clean JSON format, making it easy to integrate into financial systems or reports. This approach ensures accurate, fair, and automated billing for partial-month rentals, supporting dynamic business needs such as co-working space billing, equipment rentals, and subscription-based services.