Got it. You want the \*\*available balance to be recomputed in real-time\*\* after each \*\*valid (successful) insert\*\* in the loop. That ensures later entries are evaluated \*\*against the updated state\*\*, not the original balance.

We’ll slightly adjust the use case logic to reflect this.

---

## ✅ REVISED GOAL

- Each entry is checked against the \*\*current available balance\*\*.

- After a \*\*valid insert\*\*, the \*\*balance is updated immediately\*\*.

- All processing is \*\*sequential\*\*, simulating real-time evaluation.

---

## 🔧 Updated Use Case: `BulkInsertEntriesHandler`

```csharp

public class BulkInsertEntriesHandler

{

private readonly IEntryRepository \_entryRepository;

public BulkInsertEntriesHandler(IEntryRepository entryRepository)

{

\_entryRepository = entryRepository;

}

public async Task<(List<Entry>, List<EntryProcessingResultDto>, int)> HandleAsync(BulkInsertEntriesRequest request)

{

var validEntries = new List<Entry>();

var results = new List<EntryProcessingResultDto>();

const int maxLimit = 300;

int currentTotal = await \_entryRepository.GetCurrentTotalAsync();

int availableBalance = maxLimit - currentTotal;

foreach (var dto in request.Entries)

{

if (dto.Value <= availableBalance)

{

var entry = new Entry { Value = dto.Value };

validEntries.Add(entry);

currentTotal += dto.Value;

availableBalance = maxLimit - currentTotal;

results.Add(new EntryProcessingResultDto

{

Value = dto.Value,

IsInserted = true,

Message = "Inserted",

AvailableBalance = availableBalance

});

}

else

{

results.Add(new EntryProcessingResultDto

{

Value = dto.Value,

IsInserted = false,

Message = $"Exceeds available balance of {availableBalance}",

AvailableBalance = availableBalance

});

}

}

if (validEntries.Any())

{

await \_entryRepository.AddEntriesAsync(validEntries);

}

return (validEntries, results, availableBalance);

}

}

```

---

## 📌 Explanation

- \*\*`availableBalance` is recalculated\*\* after each valid insert.

- The response reflects the \*\*real-time state\*\* after each step.

- The final balance is passed back after all processing.

---

## 🧪 Example Scenario

Initial balance: `300`

Incoming values: `[100, 120, 90, 50]`

Evaluation:

1. `100` ✅ → balance becomes `200`

2. `120` ✅ → balance becomes `80`

3. `90` ❌ → rejected (90 > 80)

4. `50` ✅ → balance becomes `30`

### Response:

```json

{

"InsertedCount": 3,

"FinalAvailableBalance": 30,

"Entries": [

{

"value": 100,

"isInserted": true,

"message": "Inserted",

"availableBalance": 200

},

{

"value": 120,

"isInserted": true,

"message": "Inserted",

"availableBalance": 80

},

{

"value": 90,

"isInserted": false,

"message": "Exceeds available balance of 80",

"availableBalance": 80

},

{

"value": 50,

"isInserted": true,

"message": "Inserted",

"availableBalance": 30

}

]

}

```

---

Would you like to also add \*\*per-entry timestamps\*\*, or return the \*\*total of failed entries\*\* too?

public class BulkInsertEntriesHandler

{

private readonly IEntryRepository \_entryRepository;

public BulkInsertEntriesHandler(IEntryRepository entryRepository)

{

\_entryRepository = entryRepository;

}

public async Task<(List<Entry>, List<EntryProcessingResultDto>, int)> HandleAsync(BulkInsertEntriesRequest request)

{

var validEntries = new List<Entry>();

var results = new List<EntryProcessingResultDto>();

const int maxLimit = 300;

int currentTotal = await \_entryRepository.GetCurrentTotalAsync();

int availableBalance = maxLimit - currentTotal;

foreach (var dto in request.Entries)

{

// Check if already exists

bool exists = await \_entryRepository.EntryExistsAsync(dto.Value);

if (exists)

{

results.Add(new EntryProcessingResultDto

{

Value = dto.Value,

IsInserted = false,

Message = "Duplicate entry already exists",

AvailableBalance = availableBalance

});

continue;

}

// Check balance

if (dto.Value <= availableBalance)

{

var entry = new Entry { Value = dto.Value };

validEntries.Add(entry);

currentTotal += dto.Value;

availableBalance = maxLimit - currentTotal;

results.Add(new EntryProcessingResultDto

{

Value = dto.Value,

IsInserted = true,

Message = "Inserted",

AvailableBalance = availableBalance

});

}

else

{

results.Add(new EntryProcessingResultDto

{

Value = dto.Value,

IsInserted = false,

Message = $"Exceeds available balance of {availableBalance}",

AvailableBalance = availableBalance

});

}

}

if (validEntries.Any())

{

await \_entryRepository.AddEntriesAsync(validEntries);

}

return (validEntries, results, availableBalance);

}

}