Nightly Batch Processing System

Timesheet Management System

PART 1: FUNCTIONAL OVERVIEW

1. Introduction

The Nightly Batch Processing System handles automated report generation that runs during off-peak hours (overnight). Reports are generated while users are offline, and results are available for review the next business day.

Why Nightly Batch?

Benefit	Description
Off-peak processing	Heavy reports run when system is idle
No user impact	Processing doesn't slow down daytime operations
Consistent schedule	Reports ready at same time each day
Complete data	All daily transactions captured before processing

2. What is Nightly Batch Processing?

Nightly batch processing is a scheduled approach where jobs are queued and executed automatically during a defined overnight window. Unlike real-time processing, users don't wait for results—they review outcomes the next day.

Key Concepts

Term	Meaning
Batch Window	The overnight period when jobs run (e.g., 11 PM - 5 AM)
Job	A single unit of work (e.g., one report generation)
Job Run	One execution of a job with logged start/end times
Dependencies	Jobs that must complete before others can start
Post-Run Review	Morning check of job statuses and results

3. Design Goals

The system is designed with four primary goals to ensure reliable nightly batch execution:

Goal	Description	How Achieved
Resilience	Handle failures gracefully	Auto-retry, failure isolation, state persistence
Flexibility	Easy to add new jobs	Configuration-driven, pluggable architecture
Observability	Full visibility into runs	Start/end timestamps, duration tracking
Recoverability	Fix and retry failures	Morning review, selective retry, error logs

4. Key Capabilities

Capability	Description	Status
Scheduled Execution	Jobs run automatically at configured times	✓
Job Dependencies	Jobs can trigger other jobs	✓
Start/End Logging	Every job logs timestamps	✓
Duration Tracking	Calculate how long each job takes	✓
Automatic Retry	Failed jobs retry with backoff	✓
Post-Run Status	Dashboard shows all job outcomes	✓
Selective Retry	Retry only failed jobs	✓
Multiple Formats	JSON, CSV, PDF, Excel output	✓
Role-Based Masking	Sensitive data protected	✓
Priority Processing	Critical jobs run first	✓
Manual Job Retry API	Retry failed jobs via endpoint	✗
Email Notifications	Alert on job failures	Partial

5. How Jobs Work

What Gets Logged

Field	Description	Used For
job_id	Unique identifier	Tracking and debugging
status	QUEUED/PROCESSING/COMPLETED/FAILED	Status monitoring
created_at	When job was queued	Audit trail
started_at	When processing began	Duration calculation
completed_at	When processing ended	Duration calculation
error_message	Failure reason	Debugging
retry_count	Number of attempts	Retry tracking

Duration Calculation

```
Duration = completed_at - started_at

Example Job Log:
```

```
■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■
Job ID: JOB_ABC123     Type: COUNTY_DAILY
Status: COMPLETED
Created: 2025-12-08 23:00:00
Started: 2025-12-08 23:00:05
Completed: 2025-12-08 23:02:45
Duration: 2 minutes 40 seconds
Records: 5,000 processed
■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■
```

6. Job Dependencies

Jobs can be chained so completing one automatically triggers another.

Pattern	Description	Example
Single (1→1)	One parent triggers one child	County Report → Summary Report
Multiple (N→1)	Multiple parents trigger one child	Weekly + Monthly → Consolidated

Failure Handling

If a parent job FAILS, dependent jobs do NOT trigger. This prevents cascading bad data—failed jobs wait for manual review and retry.

7. Scheduling

Schedule	Time (IST)	Jobs Created
Daily	11:00 PM	County reports, daily summaries
Weekly	Monday 11:00 PM	Weekly aggregations
Monthly	1st of month 11:00 PM	Monthly reports
Quarterly	Jan/Apr/Jul/Oct 1st	Quarterly reviews
Yearly	January 1st	Annual reports

8. Monitoring & Visibility

The morning after each batch run, operations staff review job outcomes via the status dashboard.

What to Check Each Morning

Check	What to Look For	Action if Problem
Job Count	All expected jobs ran	Investigate missing jobs
Failures	Any FAILED status	Review error, fix, retry
Duration	Jobs within expected time	Investigate slow jobs
Dependencies	Chains completed	Check parent job status
Output Files	Reports generated	Verify file locations

9. Security & Data Protection

Role	Data Access	Masking Applied
ADMIN	All counties, all data	None
SUPERVISOR	Assigned counties	Partial (last 4 SSN)
CASE_WORKER	Single county only	Heavy (SSN hidden)

PART 2: TECHNICAL DETAILS

10. Architecture

The system uses a poll-based database queue pattern optimized for nightly batch execution.

```
                     SCHEDULED TRIGGERS (Cron, 11 PM)
■■■■■■■■■■■■■■■■  ■■■■■■■■■■■■■■■■  ■■■■■■■■■■■■■■■■
■    Daily     ■  ■   Weekly     ■  ■   Monthly    ■
■■■■■■■■■■■■■■■■  ■■■■■■■■■■■■■■■■  ■■■■■■■■■■■■■■■■
       ■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■
                          ▼
               ■■■■■■■■■■■■■■■■■■■■■■■
               ■ScheduledReportSvc   ■ ← Creates jobs, logs created_at
               ■■■■■■■■■■■■■■■■■■■■■■■
                          ▼
■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■
■               PostgreSQL: report_jobs table              ■
■  job_id ■ status      ■ created_at ■ started_at ■ ...    ■
■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■
                          ■ Poll every 5 seconds
                          ▼
               ■■■■■■■■■■■■■■■■■■■■■■■
               ■  BatchJobScheduler  ■ ← Claims jobs, logs started_at
               ■■■■■■■■■■■■■■■■■■■■■■■
                          ▼
               ■■■■■■■■■■■■■■■■■■■■■■■
               ■   Worker Threads    ■
               ■■■■■■■■■■■■■■■■■■■■■■■
                          ▼
               ■■■■■■■■■■■■■■■■■■■■■■■
               ■ BackgroundProcessor ■ ← Processes, logs completed_at
               ■■■■■■■■■■■■■■■■■■■■■■■
```

Design Choices

Choice	Rationale
Database as queue	Jobs persist across restarts; full history retained
Poll-based	Simple; 5-second latency acceptable for batch
Chunked processing	Large nightly datasets processed efficiently
Timestamp logging	Full visibility into job timing for post-run review

11. Tech Stack

Component	Technology	Purpose
Runtime	Java 17	Application runtime
Framework	Spring Boot 3.2	Scheduling, dependency injection
Database	PostgreSQL 13	Job queue, history, data storage
Security	Keycloak + JWT	Authentication and authorization
Scheduler	Spring @Scheduled	Cron-based job triggering

12. End-to-End Flow

Step	Time	Action
1	11:00 PM	Cron triggers ScheduledReportService
2	11:00 PM	14 jobs created with status=QUEUED, created_at logged
3	11:00:05	BatchJobScheduler claims jobs, logs started_at
4	11:00-11:45	Workers process jobs in chunks, update progress
5	On complete	Status=COMPLETED, completed_at logged, deps triggered
6	8:00 AM	Operations reviews dashboard, checks for failures
7	If failed	Review error, fix issue, retry failed jobs only

13. Core Components

Component	File	Responsibility
Scheduler	ScheduledReportService.java	Cron-triggered job creation
Queue Service	JobQueueService.java	Create/claim jobs, update status
Batch Scheduler	BatchJobScheduler.java	Poll for jobs, dispatch workers
Processor	BackgroundProcessingService.java	Execute job logic, log times
Dependencies	JobDependencyService.java	Trigger dependent jobs

14. Configuration

Parameter	Default	Description
batch.scheduler.interval-ms	5000	Poll interval (milliseconds)
batch.executor.core-pool-size	2	Number of worker threads
job-processing.default-chunk-size	1000	Records per chunk
job-processing.max-retries	3	Retry attempts before failing

15. Adding New Jobs

The system is designed to easily accommodate new job types through configuration:

Step	Action
1. Define report type	Add to report type enum or configuration
2. Add to schedule	Add @Scheduled method or YAML config entry
3. Configure dependencies	Add dependency rules in application.yml
4. Test	Trigger manually, verify timestamps and output

Summary

Aspect	Implementation
Pattern	Nightly batch with database queue
Schedule	Cron-triggered (11 PM default)
Processing	Asynchronous, chunked
Visibility	Full timestamp logging (created/started/completed)
Recovery	Post-run morning review, selective retry

Design Goals Achieved

Resilience	Automatic retry, failure isolation, state persistence
Flexibility	Configuration-driven, pluggable jobs, easy additions
Observability	Start/end timestamps, duration tracking, status dashboard
Recoverability	Morning review, selective retry, error details logged