Project: ATTN

Phase 4: Background & Scheduled Tasks - Completion Report

Date: June 22, 2025

1. Overview

Phase 4 of the ATTN project, focused on establishing the application's background processing and data maintenance logic, is now complete. The primary objective was to define and test the scheduled tasks responsible for maintaining data integrity between the live Redis cache and the persistent PostgreSQL database.

This phase also included a critical architectural review, resulting in a significant simplification of the technology stack for improved maintainability and reduced complexity.

2. Key Accomplishments & Artifacts

2.1. Implementation of Cron Job Logic (`cron.py`)

The core logic for the application's two main housekeeping tasks was fully implemented and tested.

- `sweep_users_task`**:** This function correctly scans Redis for expired user sessions, saves the user data to PostgreSQL for permanent record-keeping, and cleans the expired session keys from Redis.
- `sweep_attendances_task`**:** This function successfully finds completed attendance sessions, gathers all associated student records, saves the complete session data to PostgreSQL, and removes all related keys from Redis.
- **TDD Validation:** Both functions were validated with a comprehensive test suite (`tests/tasks/test_cron.py`) to ensure they perform their duties correctly and handle edge cases as expected.

2.2. Major Architectural Simplification (Removal of Celery)

During this phase, a crucial architectural decision was made to **remove Celery** from the project.

- **Reasoning:** After a thorough analysis of the application's real-world needs, it was determined that the added complexity of a dedicated task queue system like Celery was unnecessary.
    - Fast operations (like Redis writes) are handled instantly by the API.
    - Slow database writes for attendance data are already handled asynchronously by the `sweep_attendances_task`, which acts as a batch processor.

- **Outcome:** This decision significantly simplifies the project's deployment, configuration, and maintenance overhead without sacrificing performance or reliability for our specific use case.

2.3. Finalized Background Task Strategy

The project's strategy for background and scheduled tasks is now clear and robust.

- **Scheduled Tasks:** The `sweep_users_task` and `sweep_attendances_task` will be run periodically using `fastapi-scheduler`, which will be integrated directly into the main FastAPI application. This eliminates the need for a separate Celery Beat service.
- **Asynchronous Operations:** The initial plan to use Celery for handling high-volume requests (e.g., thousands of students submitting attendance) was deemed unnecessary. The application will handle these directly, relying on the speed of Redis and the efficiency of the database clients. A rate-limiting strategy will be employed for security and stability.

3. Current Status

Phase 4 is **100% complete**. The logic for all background maintenance tasks has been implemented and tested, and the overall architecture has been streamlined.

4. Next Steps

With the data layer and background logic in place, we are now fully prepared to begin **Phase 5: The Service Layer**. We will start by creating `teacher_service.py` and implementing the business logic that orchestrates calls to our database clients, Redis clients, and external modules.