Project: ATTN

Phase 2: Data Access Layer Implementation - Completion Report

Date: June 22, 2024

Scope of this Phase:

- **Starts After Commit:** "added phase 1"
- **Finishes with Commit:** "including added phase 2"

1. Overview

Phase 2 of the ATTN project, focused on the implementation of the Data Access Layer (DAL), is now complete. The primary objective of this phase was to build robust, fully-tested, and efficient client classes for interacting with the PostgreSQL and Redis databases.

Following a strict Test-Driven Development (TDD) methodology, we successfully created both the AsyncPostgresClient and RedisClient. Every public method in these clients is covered by one or more unit tests, ensuring reliability and adherence to the architectural design.

2. Key Accomplishments & Artifacts

2.1. Test-Driven Development (TDD) Cycle

The development process for this phase strictly followed the "Red-Green-Refactor" TDD cycle:

1. **Red:** Comprehensive test suites (test_db_client.py, test_redis_client.py) were written first, outlining the required functionality for every method. These tests initially failed, providing clear development targets.
2. **Green:** The code for each method in the client classes was written with the specific goal of making the corresponding tests pass.
3. **Refactor:** The code and tests were continuously refined for clarity, efficiency, and correctness, addressing design trade-offs and environmental challenges (such as Windows-specific asyncio loop issues).

2.2. AsyncPostgresClient Implementation

A fully functional and tested client for all PostgreSQL interactions is complete.

- **Functionality:** Handles all CRUD (Create, Read, Update, Delete) operations for Users, Attendances, and AttendanceRecords.
- **Key Features:**

    - **Bulk Operations:** Uses executemany for efficient, high-performance insertion of multiple records at once.
    - **Soft Deletes:** Implemented robust soft-delete logic for both Attendances and AttendanceRecords, preserving data for auditing while ensuring that "deleted" records are excluded from standard queries.
    - **Type Safety:** Seamlessly converts database records to and from Pydantic models, ensuring data integrity at the application level.

2.3. RedisClient Implementation

A fully functional and tested client for managing live session data in Redis is complete.

- **Functionality:** Manages the lifecycle of live user sessions, attendance sessions, and individual attendance records.
- **Key Features:**
    - **Schema Adherence:** Implemented the key schemas (users:*, attendances:*, attendance_records:*) exactly as designed.
    - **Efficient Lookups:** Utilizes direct key lookups for speed and the SCAN command for pattern-based searches to avoid blocking the server.
    - **Correct finish_attendance Logic:** After careful consideration, the finish_attendance method was correctly implemented to update a session's end_time rather than deleting it, deferring cleanup to the planned cron jobs.
    - **Pydantic Serialization:** Leverages .model_dump_json() and .model_validate_json() for reliable serialization and deserialization of objects stored in Redis.

3. Current Status

Phase 2 is **100% complete**. The project now has a fully functional and rigorously tested Data Access Layer.

- **Deliverables:**
    - app/backend/db/db_client.py
    - app/backend/db/redis_client.py
    - tests/test_db_client.py
    - tests/test_redis_client.py

4. Next Steps

With a reliable foundation and data layer, we are now ready to begin **Phase 3: Configuration and External Modules**. The plan is as follows:

1. **Configuration Setup:**
    - Add the necessary URLs and credentials for the external 'Aksis' system to the

.env file.
    - Implement app/backend/config/config.py to securely load these variables from the environment.
2. **Tools and Modules Implementation (TDD):**
    - Apply the Test-Driven Development approach to the tools and modules folders.
    - This will involve writing tests for and then implementing the verifier.py functionalities and the aksis.py and lesson_finder.py modules.