Project ATTN: Refactoring Report - Phase 1

Subject: Improvement of Data Models and Data Access Layer (DAL)
Date: June 27, 2025
Status: Planning and Implementation

1. Introduction and Goals

This report details the first phase of the ATTN project's refactoring process. The primary objective of this phase is to build the necessary infrastructure to resolve the core data integrity issue within the system and to streamline session management. To this end, the data models stored in Redis will be redesigned, and the Data Access Layer (DAL) will be updated to serve these new models and business logic.

Goals for This Phase:

1. **Define Redis Data Models:** Create new Pydantic models to store critical data in a more structured and complete manner on Redis.
2. **Update the Data Access Layer (DAL):** Refactor the redis_client to support the new data models and to handle user session management efficiently.
3. **Implement Testing:** Write unit tests for the updated DAL functions to verify the system's reliability.

2. Data Model Design Decisions

To solve the system's main problem, the "Foreign Key" error, it is essential that all data required by the periodic cron job is available in a single, consistent location within Redis. The following model changes have been designed for this purpose.

2.1. AttendenceRecordRedis Model

During the processing of an attendance record, it is critical to have all the necessary information to create a new User object if the student does not yet exist in the database.

Evaluation:

- **Option A:** Remove the student_number field and replace it with a user field containing the entire User object.
- **Option B:** Keep the student_number field and add a user_full_name field alongside it.

Decision (Option B):
The student_number will remain the unique primary identifier for the student. In addition, the user_full_name field will be added to the model. The advantages of this approach are:

- **Data Integrity:** When the cron job runs, it will have both the school number

(student_number) and the full name (user_full_name) required to create a new User.
- **Efficiency:** Adding only the required name instead of serializing an entire User object to Redis keeps the data size small and improves performance.

2.2. LessonRedis Model

In the current system, lesson information is stored as an unstructured Redis key. To solve this, a structured Pydantic model has been created.

This model will be saved to Redis associated with an attendence_id when a session begins, allowing the cron job to access all necessary information in a structured manner.

Dual Benefit of This Change:
This model also establishes a discoverable data structure that can be used to improve the student user experience. With this change, we can create an API endpoint where a student can find and join an active session by providing the teacher's name and the lesson name.

3. Data Access Layer (DAL) Refactoring

To support the new data models and logic, significant updates will be made to the redis_client.py file.

Planned Changes (redis_client.py):

- **Pydantic Integration:** Existing methods will be updated to accept Pydantic models, serialize them to JSON for storage, and deserialize them back on read.
- **User Session Management with TTL:**
  - Implement methods to manage the lifecycle of user sessions directly in Redis.
  - save_user_session(user: UserRedis, ttl: int): Saves a user's session data to Redis with a Time-To-Live (TTL). This replaces the need for a separate cron job to clean up user data.
  - get_user_session(user_id: str) -> UserRedis | None: Retrieves a user's session data.
  - delete_user_session(user_id: str): Explicitly deletes a user's session, for instance, on logout.
- **New Methods for Attendance:**
  - save_lesson(attendence_id: str, lesson: LessonRedis): Saves a LessonRedis object for a given attendence_id.

- get_lesson(attendence_id: str) -> LessonRedis | None: Returns lesson information as a LessonRedis object.
- save_attendence_record_user(...): Adds a student's attendance record using the AttendenceRecordRedis model.
- get_attendence_record_users(attendence_id: str) -> list[AttendenceRecordRedis]: Returns a list of all student records for an attendance session.

4. Test Strategy

At the end of this phase, unit tests will be written for the DAL to ensure the correctness of all implemented changes.

Test Scope (tests/db/test_redis_client.py):

1. **LessonRedis and AttendenceRecordRedis Tests:**
   - Verify that LessonRedis and AttendenceRecordRedis objects are successfully saved to and read from Redis.
2. **User Session Management Tests:**
   - Test that save_user_session correctly stores user data and applies the specified TTL.
   - Verify that a key with a TTL expires and is automatically deleted after the specified time.
   - Test that delete_user_session successfully removes the user session key.

These tests will guarantee that the data layer is stable and correct before proceeding to the next phase (cron job refactoring).