Project ATTN: Refactoring Report - Phase 2

Subject: Adapting the Service Layer to the New Data Architecture
Date: June 28, 2025
Status: Planning

1. Introduction and Goals

Following the successful completion of Phase 1, we now have a robust and tested Data Access Layer (DAL). The primary objective of this second phase is to reconfigure the **Service Layer** (auth, teacher_service, student_service), which contains the application's core business logic, to use the new redis_client methods created in Phase 1.

By the end of this phase, the application's core functions (session management, starting/ending lessons, joining attendance) will be operating on the new, more efficient, and error-free architecture. This is the most critical refactoring step of the project and will resolve the data integrity issues of the original system at their root.

Goals for This Phase:

1. **Modernize Session Management:** Completely eliminate the sweep_users cron job by delegating session management to Redis's native TTL (Time-To-Live) mechanism.
2. **Refactor the Teacher Service:** Update the lesson start and end processes to use the new structured LessonRedis model.
3. **Improve the Student Service:** Make the process of joining attendance more user-friendly and guarantee data integrity with the AttendanceRecordRedis model.

2. Planned Changes and Tasks

2.1. Session Management (auth service and related areas)

This task will simplify the lifecycle of user sessions and eliminate the inefficient cron job.

- **login Function:**
  - **Current State:** Adds user data to Redis without an expiration.
  - **New State:** A UserSessionRedis object will be created. The login logic will then check the user's role. Based on the role, it will select a corresponding TTL value from the configuration file (config.py), for example: TEACHER_SESSION_TTL_SECONDS = 3600 for teachers and STUDENT_SESSION_TTL_SECONDS = 600 for students. The user session object and the appropriate TTL will be passed to the

redis_client.save_user_session() method. This ensures that sessions for different user types expire after their designated durations.

- **logout Function:**
  - **Current State:** Potentially only performs client-side cleanup or no server-side action.
  - **New State:** It will call redis_client.delete_user_session() with the user's school number, immediately terminating the user's session record in Redis.

2.2. Teacher Service (teacher_service.py)

This task will simplify the lesson management logic and make it more reliable.

- **start_attendance Function:**
  - **Current State:** Calls redis_client.add_to_active_lesson_index to create an unstructured index.
  - **New State:** It will create a LessonRedis object with the necessary information (teacher name, lesson name, etc.). This object will be sent to the redis_client.save_lesson() method, which will atomically create both the main lesson data and the secondary index for discoverability.
- **finish_attendance Function:**
  - **Current State:** Calls redis_client.remove_from_active_lesson_index.
  - **New State:** It will retrieve the relevant LessonRedis object or its information and call redis_client.delete_lesson(). This method will clean up both the primary record of the lesson and its reference in the index in a single step.

2.3. Student Service (student_service.py)

This task will improve the student experience and completely resolve the data integrity issue.

- **find_active_lessons (New or Refactored Function):**
  - **Purpose:** To allow students to join lessons easily.
  - **Functionality:** It will take lesson_name and teacher_name parameters from the API and call the redis_client.find_lessons_by_name() method. If there are multiple active lessons with the same name, it will return a list of all matching lessons so the student can choose the correct one.
- **join_attendance Function:**
  - **Current State:** Creates an AttendanceRecord without the student_full_name information.
  - **New State:** It will create a complete AttendanceRecordRedis object, including the joining student's user_full_name. This object will be passed to the redis_client.add_attendance_record() method. This change guarantees that the sweep_attendances cron job, which will be updated in the final phase, will

have all the data it needs.

3. Expected Outcome

Once this phase is complete, the application's service layer will be fully revamped. The system will be simpler, more efficient, and, most importantly, free from data integrity issues. The sweep_users cron job will be officially decommissioned, and the overall complexity of the project will be significantly reduced.