BÁO CÁO PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG

PHƯƠNG PHÁP HƯỚNG ĐỐI TƯỢNG (OOAD/UML)

Hệ thống Báo thức Thông minh với Quiz và QR Code

Ngày 7 tháng 1 năm 2026

Mục lục

Chương 1

Phân tích

1.1 Biểu đồ phân cấp chức năng

1.1.1 Mô tả đặc điểm

Biểu đồ phân cấp chức năng (Function Decomposition Diagram - FDD) trong phương pháp OOAD tập trung vào việc xác định các chức năng từ góc nhìn hướng đối tượng, chuẩn bị cho việc ánh xạ sang Use Case. Hệ thống được chia thành 5 module chính, mỗi module tương ứng với một nhóm Use Case.

1.1.2 Chia module

Dựa trên phân tích codebase (package structure và entry points), hệ thống được chia thành 5 module:

Module 1: Quản lý Báo thức
Module này quản lý vòng đời của báo thức, bao gồm 6 chức năng chính:

- Tạo báo thức mới (với đầy đủ cấu hình)
- Chỉnh sửa báo thức (sửa thông tin, thay đổi mission/QR)
- Xóa báo thức (và hủy lịch hẹn)
- Bật/Tắt báo thức (toggle `isEnabled`)
- Xem danh sách báo thức (với sắp xếp)
- Tạo báo thức nhanh (đổ chuông sau X phút)

Từ góc nhìn OOP, module này được implement bởi:

- **UI:** AlarmScreen, AlarmSettingsScreen
- **ViewModel:** AlarmViewModel, AlarmSettingsViewModel
- **Logic:** AlarmScheduler (tương tác với Android AlarmManager)
- **Data:** AlarmEntity, AppDao (CRUD operations)

Module 2: Quản lý Chủ đề & Câu hỏi
Module này quản lý kho câu hỏi theo cấu trúc phân cấp chủ đề, bao gồm 8 chức năng:

- Tạo chủ đề mới
- Sửa tên chủ đề
- Xóa chủ đề (cascade delete câu hỏi)
- Tìm kiếm chủ đề
- Thêm câu hỏi vào chủ đề
- Chỉnh sửa câu hỏi

- Xóa câu hỏi
- Xem chi tiết câu hỏi

Implement bởi:

- **UI:** TopicScreen, TopicDetailScreen, AddQuestionDialog
- **ViewModel:** TopicViewModel, TopicDetailViewModel
- **Data:** TopicEntity, QuestionEntity, AppDao

Module 3: Thực thi Báo thức
Module này xử lý toàn bộ luồng báo thức reo, bao gồm 6 chức năng chính:

- Kích hoạt báo thức (nhận broadcast từ AlarmManager)
- Hiển thị giao diện báo thức đang reo (full-screen activity)
- Thực hiện Quiz (với 4 chức năng con: chọn câu hỏi SRS, hiển thị đếm giờ, kiểm tra đáp án, cập nhật tiến độ)
- Quét QR Code để tắt
- Snooze báo thức
- Tắt báo thức

Implement bởi:

- **UI:** AlarmRingingActivity, AlarmRingingScreen, QuizScreen
- **ViewModel:** QuizViewModel
- **Logic:** AlarmReceiver, AlarmService (foreground service phát nhạc), QuestionAlgorithmManager (thuật toán SRS)
- **Data:** QuestionEntity, QuestionProgressEntity, HistoryEntity, AlarmHistoryEntity

Module 4: Quản lý QR Code
Module này quản lý kho mã QR/Barcode (tối đa 5 mã), bao gồm 5 chức năng:

- Quét QR/Barcode (camera + ML Kit)
- Lưu mã QR
- Xóa mã QR
- Liên kết QR với báo thức (tối đa 3 mã/báo thức)
- Xác thực mã QR khi tắt báo thức

Implement bởi:

- **UI:** QRCodeScannerScreen, QRCodeManagementDialog, QRCodeSelectionDialog
- **ViewModel:** QRCodeViewModel
- **Data:** QRCodeEntity, AlarmQRLinkEntity, AppDao

Module 5: Thống kê & Báo cáo
Module này cung cấp các báo cáo về hiệu suất học tập và thói quen thức dậy, bao gồm 5 chức năng:

- Xem thống kê độ chính xác theo tuần (biểu đồ đường 7 ngày)
- Xem phân phối trạng thái học tập (biểu đồ tròn New/Learning/Mastered)
- Tính điểm Wake-up Score (0–100)
- Theo dõi streak và điểm số
- Xem lịch sử báo thức

Implement bởi:

- **UI:** StatsScreen (với Line Chart, Pie Chart, Card)
- **ViewModel:** StatsViewModel
- **Data:** StatisticsDao (query phức tạp với GROUP BY, JOIN)

```
%% ơS đồ FDD - Function Decomposition Diagram
%% ệH ốthng Báo úthc Thông minh ớvi Quiz và QR Code

graph TB
    Root[ệH ốthng Báo úthc<br/>Thông minh]

    Root --> M1[1. ảQun lý<br/>Báo úthc]
    Root --> M2[2. ảQun lý<br/>ủCh đề & Câu ỏhi]
    Root --> M3[3. ựThc thi<br/>Báo úthc]
    Root --> M4[4. ảQun lý<br/>QR Code]
    Root --> M5[5. ốThng kê &<br/>Báo cáo]

    %% Module 1: ảQun lý Báo úthc
    M1 --> F1_1[1.1 ạTo báo úthc ởmi]
    M1 --> F1_2[1.2 ỉChnh ửsa báo úthc]
    M1 --> F1_3[1.3 Xóa báo úthc]
    M1 --> F1_4[1.4 ậBt/ắTt báo úthc]
    M1 --> F1_5[1.5 Xem danh sách<br/>báo úthc]
    M1 --> F1_6[1.6 ạTo báo úthc nhanh]

    %% Module 2: ảQun lý ủCh đề & Câu ỏhi
    M2 --> F2_1[2.1 ảQun lý ủch đề]
    M2 --> F2_2[2.2 ảQun lý câu ỏhi]

    F2_1 --> F2_1_1[2.1.1 ạTo ủch đềở mi]
    F2_1 --> F2_1_2[2.1.2 ửSa tên ủch đề]
    F2_1 --> F2_1_3[2.1.3 Xóa ủch đề]
    F2_1 --> F2_1_4[2.1.4 Tìm ếkim ủch đề]

    F2_2 --> F2_2_1[2.2.1 Thêm câu ỏhi<br/>vào ủch đề]
    F2_2 --> F2_2_2[2.2.2 ỉChnh ửsa<br/>câu ỏhi]
    F2_2 --> F2_2_3[2.2.3 Xóa câu ỏhi]
    F2_2 --> F2_2_4[2.2.4 Xem chi ếtit<br/>câu ỏhi]

    %% Module 3: ựThc thi Báo úthc
    M3 --> F3_1[3.1 Kích ạhot báo úthc]
    M3 --> F3_2[3.2 ểHin ịth giao ệdin<br/>báo úthc đang reo]
    M3 --> F3_3[3.3 ựThc ệhin Quiz]
    M3 --> F3_4[3.4 Quét QR Code<brể/> ắtt]
    M3 --> F3_5[3.5 Snooze báo úthc]
    M3 --> F3_6[3.6 ắTt báo úthc]

    F3_3 --> F3_3_1[3.3.1 ọChn câu ỏhi<br/>theo ậthut toán SRS]
    F3_3 --> F3_3_2[3.3.2 ểHin ịth câu ỏhi<br/>và đếm ờgi]
    F3_3 --> F3_3_3[3.3.3 ểKim tra đáp án]
    F3_3 --> F3_3_4[3.3.4 ậCp ậnht ếtin độ<br/>ọhc ậtp SRS]

    %% Module 4: ảQun lý QR Code
    M4 --> F4_1[4.1 Quét QR/Barcode]
    M4 --> F4_2[4.2 ưLu mã QR]
    M4 --> F4_3[4.3 Xóa mã QR]
    M4 --> F4_4[4.4 Liên ếkt QR<br/>ớvi báo úthc]
    M4 --> F4_5[4.5 Xác ựthc mã QR<br/>khi ắtt báo úthc]

    %% Module 5: ốThng kê & Báo cáo
    M5 --> F5_1[5.1 Xem ốthng kê<brđộ/> chính xác<br/>theo ầtun]
    M5 --> F5_2[5.2 Xem phân ốphi<br/>ạtrng thái ọhc ậtp]
    M5 --> F5_3[5.3 Tính điểm<br/>Wake-up Score]
    M5 --> F5_4[5.4 Theo dõi streak<br/>và điểm ốs]
    M5 --> F5_5[5.5 Xem ịlch ửs<br/>báo úthc]

    style Root fill:#e1f5ff
    style M1 fill:#fff4e6
    style M2 fill:#fff4e6
    style M3 fill:#fff4e6
    style M4 fill:#fff4e6
```

1.2 Biểu đồ Use Case

1.2.1 Tổng quan

Use Case Diagram mô tả tương tác giữa các tác nhân (actors) và hệ thống thông qua các trường hợp sử dụng (use cases). Hệ thống có 2 actors chính và 24 use cases được nhóm thành 5 package theo module.

1.2.2 Các Actor

1. **Người dùng (User):** Actor chính, tương tác với hầu hết các use case (tạo/quản lý báo thức, chủ đề/câu hỏi, trả lời quiz, quét QR, xem thống kê).
2. **Android AlarmManager:** Actor hệ thống, gửi broadcast intent khi đến giờ đổ chuông (tương tác với UC13 - Nhận báo thức reo).

1.2.3 Sơ đồ Use Case tổng quát

```
@startuml
'** Use Case Diagram - ệH ốthng Báo úthc Thông minh **

left to right direction
skinparam packageStyle rectangle

actor "ườNgi dùng" as User
actor "Android AlarmManager" as AlarmMgr

rectangle "ệH ốthng Báo úthc Thông minh" {

    package "ảQun lý Báo úthc" {
        usecase "UC01: ạTo báo úthc ớmi" as UC01
        usecase "UC02: ỉChnh ửsa báo úthc" as UC02
        usecase "UC03: Xóa báo úthc" as UC03
        usecase "UC04: ậBt/ắTt báo úthc" as UC04
        usecase "UC05: Xem danh sách báo úthc" as UC05
        usecase "UC06: ạTo báo úthc nhanh" as UC06
    }

    package "ảQun lý ủCh đề & Câu ỏhi" {
        usecase "UC07: ạTo ủch đề mới" as UC07
        usecase "UC08: Thêm câu ỏhi vào ủch đề" as UC08
        usecase "UC09: ỉChnh ửsa câu ỏhi" as UC09
        usecase "UC10: Xóa câu ỏhi" as UC10
        usecase "UC11: Xem chi ếtit ủch đề" as UC11
        usecase "UC12: Tìm ếkim ủch đề" as UC12
    }

    package "ựThc thi Báo úthc" {
        usecase "UC13: ậNhn báo úthc reo" as UC13
        usecase "UC14: ảTr ời câu ỏhi Quiz" as UC14
        usecase "UC15: Quét QR Code để tt" as UC15
        usecase "UC16: Snooze báo úthc" as UC16
        usecase "UC17: ắTt báo úthc" as UC17
    }

    package "ảQun lý QR Code" {
        usecase "UC18: Quét và ưlu mã QR" as UC18
        usecase "UC19: Liên ếkt QR ớvi báo úthc" as UC19
        usecase "UC20: Xóa mã QR" as UC20
    }

    package "ốThng kê & Báo cáo" {
        usecase "UC21: Xem ốthng kê ầtun" as UC21
        usecase "UC22: Xem phân ốphi SRS" as UC22
        usecase "UC23: Xem điểm Wake-up Score" as UC23
        usecase "UC24: Theo dõi Streak" as UC24
    }

    ' Internal use cases
    usecase "ậLp ịlch ệhn ệh ốthng" as Schedule
    usecase "ọChn câu ỏhi theo SRS" as SelectSRS
    usecase "ậCp ậnht ếtin độ hc ậtp" as UpdateProgress
    usecase "Xác ựthc mã QR" as ValidateQR
}

' User relationships
User --> UC01
User --> UC02
User --> UC03
User --> UC04
User --> UC05
```

1.2.4 Chia theo module

Module Quản lý Báo thức (6 use cases):

- UC01: Tạo báo thức mới (`<<include>>` Lập lịch hẹn hệ thống)
- UC02: Chỉnh sửa báo thức (`<<include>>` Lập lịch hẹn, `<<extend>>` UC19 nếu chọn QR)
- UC03: Xóa báo thức
- UC04: Bật/Tắt báo thức (`<<include>>` Lập lịch hẹn)
- UC05: Xem danh sách báo thức
- UC06: Tạo báo thức nhanh (`<<include>>` Lập lịch hẹn)

Module Quản lý Chủ đề & Câu hỏi (6 use cases):

- UC07: Tạo chủ đề mới
- UC08: Thêm câu hỏi vào chủ đề
- UC09: Chỉnh sửa câu hỏi
- UC10: Xóa câu hỏi
- UC11: Xem chi tiết chủ đề
- UC12: Tìm kiếm chủ đề

Module Thực thi Báo thức (5 use cases):

- UC13: Nhận báo thức reo (actor: AlarmManager; `<<extend>>` UC16, UC17)
- UC14: Trả lời câu hỏi Quiz (`<<include>>` Chọn câu hỏi theo SRS, Cập nhật tiến độ học tập)
- UC15: Quét QR Code để tắt (`<<include>>` Xác thực mã QR)
- UC16: Snooze báo thức
- UC17: Tắt báo thức (`<<extend>>` UC14 nếu có quiz, `<<extend>>` UC15 nếu có QR; `<<include>>` Xác thực mã QR)

Module Quản lý QR Code (3 use cases):

- UC18: Quét và lưu mã QR
- UC19: Liên kết QR với báo thức
- UC20: Xóa mã QR

Module Thống kê & Báo cáo (4 use cases):

- UC21: Xem thống kê tuần
- UC22: Xem phân phối SRS
- UC23: Xem điểm Wake-up Score
- UC24: Theo dõi Streak

1.3 Đặc tả Use Case

1.3.1 UC01: Tạo báo thức mới

Tên: Tạo báo thức mới

Mô tả: Người dùng tạo báo thức mới với đầy đủ cấu hình (giờ/phút, nhãn, ngày lặp lại, số câu hỏi, QR code, nhạc chuông, snooze).

Tác nhân: Người dùng (chính)

Tiền điều kiện:

- Ứng dụng đã được cài đặt
- Đã cấp quyền `SCHEDULE_EXACT_ALARM`, `POST_NOTIFICATIONS`

Luồng chính:

1. User nhấn nút "Thêm báo thức mới"trên AlarmScreen
2. Hệ thống navigate đến AlarmSettingsScreen
3. Hệ thống điền giá trị mặc định (giờ/phút hiện tại, nhạc chuông mặc định, snooze: tắt)
4. User chỉnh sửa giờ/phút (Time Picker)
5. User nhập nhãn (Text Field - optional)
6. User chọn ngày lặp lại (Chip Selector: T2–CN hoặc không chọn = 1 lần)
7. User chọn nhạc chuông (Ringtone Picker)
8. User chọn số câu hỏi (Slider: 0–10)

Optional User nhấn "Chọn câu hỏi"→ Hiển thị MissionSelectionDialog

- User chọn toàn bộ Topic HOẶC chọn câu hỏi lẻ
- Hệ thống cập nhật `selectedTopics` và `selectedQuestions`

Optional User nhấn "Chọn QR"→ Hiển thị QRCodeManagementDialog (UC19)

Optional User bật Snooze và chọn thời gian (Slider: 1–60 phút)

9. Hệ thống tính toán và hiển thị "Đổ chuông sau X giờ Y phút"
10. User nhấn "Lưu"
11. Hệ thống validate dữ liệu (giờ/phút hợp lệ)
12. Hệ thống lưu AlarmEntity vào database (qua AppDao.insertAlarm)
13. Hệ thống lưu các liên kết:
    - `alarm_topic_link` (nếu chọn toàn bộ Topic)
    - `alarm_selected_questions` (nếu chọn câu hỏi lẻ)
    - `alarm_qr_link` (nếu chọn QR)
14. Hệ thống gọi AlarmScheduler.schedule() → đặt lịch hẹn với Android AlarmManager (PendingIntent với `requestCode = alarmId`)
15. Hệ thống hiển thị thông báo "Đã lưu báo thức"
16. Hệ thống navigate back → AlarmScreen
17. AlarmScreen hiển thị báo thức mới trong danh sách

Luồng thay thế:
9a. User chọn "Toàn bộ Topic":

- Hệ thống lưu liên kết vào `alarm_topic_link`
- Tất cả câu hỏi trong Topic sẽ được dùng cho Quiz

9b. User chọn "Câu hỏi lẻ":

- Hệ thống lưu vào `alarm_selected_questions`
- Chỉ các câu hỏi được chọn mới dùng cho Quiz

10a. User chưa có QR nào:

- Hệ thống hiển thị "Chưa có mã QR. Quét ngay?"
- Nếu đồng ý → UC18 (Quét và lưu mã QR)

13a. User nhấn "Hủy":

- Nếu có thay đổi: Hiển thị dialog "Bỏ thay đổi?"

  - Nếu xác nhận: Quay về danh sách
  - Nếu không: Tiếp tục chỉnh sửa
- Nếu không có thay đổi: Quay về ngay

14a. Validation lỗi:

- Hiển thị thông báo lỗi (VD: "Giờ không hợp lệ")
- Quay lại bước 4

17a. Thiếu quyền `SCHEDULE_EXACT_ALARM` (Android 12+):

- Hiển thị dialog "Ứng dụng cần quyền đặt báo thức chính xác"
- Mở Settings để user cấp quyền
- Sau khi quay lại, thử lập lịch lại

Luồng ngoại lệ:

- **E1:** Database lỗi → Hiển thị "Lỗi khi lưu báo thức. Vui lòng thử lại."
- **E2:** AlarmManager không khả dụng → Hiển thị "Không thể đặt lịch hẹn hệ thống."

Hậu điều kiện:

- Báo thức mới được lưu vào database với `isEnabled = true`
- Lịch hẹn đã được đặt trong Android AlarmManager
- User thấy báo thức mới trong danh sách

Minh họa luồng: Tham khảo Activity Diagram UC01 (Hình 1.3).

```
@startuml
'** Activity Diagram - UC01: ạTo báo úthc ớmi **

title Activity Diagram: UC01 - ạTo báo úthc ớmi

|ườNgi dùng|
start
:ấNhn nút "Thêm\nbáo úthc ớmi";

|ệH ốthng|
:ểHin ịth màn hình\ấncu hình báo úthc;Đ

:ền giá ịtr ặmc định:\n- ờGi/Phút ệhin ạti\n- ạNhc chuông ặmc định\n- Snooze
    : ắtt;

|ườNgi dùng|
repeat
    :ỉChnh ửsa thông tin;
    note right
        - ờGi, Phút
        - Nhãn (label)
        - Ngày ặlp ại
        - ạNhc chuông
        - ốS câu ỏhi
        - Snooze
    end note

    if (ọChn Mission?) then (yes)
        |ệH ốthng|
        :ểHin ịth dialog\ọnchn câu ỏhi;

        :Load danh sách\nTopics & Questions;

        |ườNgi dùng|
        fork
            :ọChn toàn ộb Topic;
        fork again
            :ọChn câu ỏhi ẻl;
        fork again
            :ọChn câu ỏhi ặmc định;
        end fork

        |ệH ốthng|
        :ậCp ậnht danh sách\ncâu ỏhi đã ọchn;
    endif

    if (ọChn QR Code?) then (yes)
        |ệH ốthng|
        :ểHin ịth dialog\ọnchn QR;

        :Load danh sách\nQR đã ưlu;

        |ườNgi dùng|
        if (Có QR ẫsn?) then (yes)
            :ọChn ốti đa 3 QR;
        else (no)
            :Quét QR ớmi;
            |ệH ốthng|
            :ởM camera;
            |ườNgi dùng|
            :Quét mã QR/Barcode;
            |ệH ốthng|
            :ưLu QR vào database;
            :ọChn QR ừva ưlu;
        endif
```

1.3.2 UC14: Trả lời câu hỏi Quiz

Tên: Trả lời câu hỏi Quiz

Mô tả: User trả lời đủ số câu hỏi đúng để tắt báo thức, hệ thống chọn câu hỏi theo thuật toán SRS và cập nhật tiến độ học tập.

Tác nhân: Người dùng (chính)

Tiền điều kiện:

- Báo thức đang reo (UC13 đã kích hoạt)
- Báo thức có `questionCount > 0`
- Đã có câu hỏi được chọn (Topics/Questions)

Luồng chính:

1. User nhấn "Tắt"trên AlarmRingingScreen
2. Hệ thống kiểm tra `questionCount`, nếu > 0 thì navigate đến QuizScreen
3. Hệ thống đọc cấu hình báo thức (alarmId, questionCount)
4. Hệ thống tạo AlarmHistoryEntity (scheduledTime, firstRingTime, snoozeCount=0)
5. Hệ thống gọi QuestionAlgorithmManager.generateMissionQuestions(alarmId, questionCount)
   - Đọc danh sách câu hỏi đã chọn (manual + topics)
   - Đọc QuestionProgressEntity của từng câu
   - Tính điểm ưu tiên: câu chưa học (500), câu đến hạn (1000+), câu khác (difficultyScore)
   - Sắp xếp và chọn Top N câu
6. Hệ thống chuyển đổi sang QuestionData (với 4 đáp án đã shuffle)
7. Hệ thống khởi tạo: `correctCount = 0`, `targetCount = questionCount`
8. [**Loop**] Cho đến khi `correctCount >= targetCount`:
   (a) Hệ thống hiển thị câu hỏi hiện tại
   (b) Hệ thống bắt đầu đếm ngược 15 giây (circular timer progress)
   (c) User xem câu hỏi và chọn 1 trong 4 đáp án
   (d) [**Race condition:**] User chọn đáp án HOẶC hết 15 giây (timeout)
   (e) Hệ thống dừng timer
   (f) Hệ thống kiểm tra đáp án
   (g) **NẾU đúng:**
      - Hiển thị "Đúng"(màu xanh)
      - `correctCount++`
      - Gọi QuestionAlgorithmManager.processAnswer(questionId, isCorrect=true, timeSpentMs):
        - `correctStreak++`
        - `easinessFactor += 0.1` (max 3.0)
        - `interval = interval * easinessFactor` (hoặc 1 nếu lần đầu)
        - `nextReviewDate = NOW() + interval (ngày)`
      - Cập nhật TopicStats: `userEloScore += 10`
   (h) **NẾU sai:**
      - Hiển thị "Sai"(màu đỏ)
      - Gọi QuestionAlgorithmManager.processAnswer(questionId, isCorrect=false, timeSpentMs):
        - `correctStreak = 0`
        - `easinessFactor -= 0.2` (min 1.3)

- `interval = 1`
  - Cập nhật TopicStats: `userEloScore -= 5` (min 0)

(i) Hệ thống ghi HistoryEntity (questionId, isCorrect, answeredAt, timeToAnswerMs)

(j) Hệ thống chờ 1 giây (để user thấy kết quả)

(k) **NẾU** `correctCount < targetCount`: Chuyển sang câu hỏi tiếp theo

9. **[End Loop]** Đã đủ số câu đúng
10. Hệ thống cập nhật AlarmHistory (dismissalTime = NOW(), isDismissed = true)
11. Hệ thống cập nhật UserStats (totalPoints += 10, currentStreak, totalAlarmsDismissed++)
12. Hệ thống gọi AlarmService.stopService() (dừng phát nhạc)
13. Hệ thống đóng QuizScreen, finish AlarmRingingActivity
14. Hệ thống quay về MainActivity

Luồng thay thế:
2a. questionCount = 0:

- Hệ thống bỏ qua Quiz, tắt báo thức ngay lập tức
- Tiếp tục bước 9 (cập nhật AlarmHistory)

5a. Không có câu hỏi nào (danh sách rỗng):

- Hệ thống hiển thị "Không có câu hỏi. Đặt lại cấu hình."
- Tắt báo thức ngay lập tức

8.4a. User không tương tác và hết 15 giây nhiều lần:

- `correctCount` không tăng → Quiz kéo dài vô hạn cho đến khi trả lời đúng đủ
- (Đây là design cố ý để buộc user tỉnh táo)

Luồng ngoại lệ:

- **E1:** User force-stop ứng dụng → Service bị kill, AlarmHistory không được cập nhật (dismissalTime = null, isDismissed = false)
- **E2:** Máy tắt nguồn giữa chừng → Tương tự E1, dữ liệu một phần bị mất

Hậu điều kiện:

- Báo thức đã tắt, nhạc dừng
- AlarmHistory có bản ghi đầy đủ (dismissalTime, isDismissed=true)
- Tiến độ SRS của các câu hỏi đã trả lời được cập nhật
- UserStats được cập nhật (điểm, streak)

Minh họa luồng: Tham khảo Activity Diagram UC14 (Hình 1.4).

```
@startuml
'** Activity Diagram - UC14: åTr ời câu ỏhi Quiz **

title Activity Diagram: UC14 - åTr ời câu ỏhi Quiz

|ườNgi dùng|
start
:ậNhn thông báo\nbáo úthc reo;

|ệH ốthng|Đọ
:c ấcu hình báo úthc\n(ốs câu ỏhi);

if (ốS câu ỏhi = 0?) then (yes)
    :ấTt báo úthc ngay;
    stop
else (no)
    :ạTo AlarmHistory;Đọ

    :c câu ỏhi đã ọchn\n(manual + topics);Đọ

    :c ếtin độ SRS\ủnca các câu ỏhi;

    :Tính điểm ưu tiên\n(SRS algorithm);
    note right
        - Câu ưcha ọhc: 500 điểm
        - Câu đến ạhn ôn: 1000 + ờthi gian quá ạhn
        - Câu khác: dùng difficultyScore
    end note

    :ắSp ếxp và ọchn\nTop N câu ỏhi;

    :ởKhi ạto ộb đếm\n(correctCount = 0);

    |ườNgi dùng|
    repeat
        |ệH ốthng|
        :ểHin ịth câu ỏhi;
        :ắBt đầu đếm ượngc\n(15 giây);

        |ườNgi dùng|
        :Xem câu ỏhi và\ncác đáp án;

        fork
            :ọChn đáp án;
        fork again
            |ệH ốthng|
            :ếHt ờgi;Đ
            :ánh ấdu timeout;
        end fork

        |ệH ốthng|
        :ừDng timer;

        if Đ(áp án đúng?) then (yes)
            :ểHin ịth Đ"úng" (xanh);
            :correctCount++;

            :ậCp ậnht SRS:\n- ăTng correctStreak\n- ăTng easinessFactor\n-
               ăTng interval;

            :Tính điểm ELO\ủnca Topic (+10);
        else (no)
            :ểHin ịth "Sai" đỏ();

            :ậCp ậnht SRS:\n- Reset correctStreak = 0\n- åGim easinessFactor
               \n- Reset interval = 1;
```

1.3.3 UC04: Bật/Tắt báo thức

Tên: Bật/Tắt báo thức

Mô tả: User bật hoặc tắt báo thức nhanh chóng bằng cách toggle switch.
Tác nhân: Người dùng (chính)
Tiền điều kiện:

- Đã có ít nhất 1 báo thức trong danh sách

Luồng chính:

1. User toggle switch của báo thức trên AlarmCard
2. Switch chuyển trạng thái (bật tắt)
3. AlarmScreen gọi AlarmViewModel.toggleAlarm(alarmId, isEnabled)
4. AlarmViewModel đọc AlarmEntity từ database (AppDao.getAlarmById)
5. AlarmViewModel cập nhật `alarm.isEnabled = isEnabled`
6. AlarmViewModel gọi AppDao.updateAlarm(alarm)
7. **NẾU** `isEnabled = true` (Bật):
   (a) AlarmViewModel gọi AlarmScheduler.schedule(alarm)
   (b) AlarmScheduler tính thời gian đổ chuông tiếp theo (xử lý trường hợp giờ/phút < hiện tại → đặt sang ngày mai)
   (c) AlarmScheduler gọi AlarmManager.setAlarmClock(time, pendingIntent) với `requestCode = alarmId`
   (d) Hệ thống hiển thị switch màu xanh + text "Báo thức sẽ reo sau X giờ Y phút"
8. **NẾU** `isEnabled = false` (Tắt):
   (a) AlarmViewModel gọi AlarmScheduler.cancel(alarm)
   (b) AlarmScheduler tạo PendingIntent với cùng `requestCode = alarmId`
   (c) AlarmScheduler gọi AlarmManager.cancel(pendingIntent)
   (d) Hệ thống hiển thị switch màu xám

Luồng thay thế:
7a. Thiếu quyền `SCHEDULE_EXACT_ALARM`:

- Hiển thị dialog yêu cầu cấp quyền
- Mở Settings
- Sau khi quay lại, nếu đã cấp quyền → Thử lập lịch lại
- Nếu vẫn không cấp → Switch quay về trạng thái tắt, hiển thị lỗi

Hậu điều kiện:

- Trạng thái `isEnabled` được cập nhật trong database
- Lịch hẹn hệ thống được đặt (nếu bật) hoặc hủy (nếu tắt)

1.3.4 UC18: Quét và lưu mã QR

Tên: Quét và lưu mã QR

Mô tả: User quét QR/Barcode bằng camera và lưu vào hệ thống (tối đa 5 mã).
Tác nhân: Người dùng (chính)
Tiền điều kiện:

- Đã cấp quyền `CAMERA`
- Số lượng QR đã lưu < 5

Luồng chính:

1. User nhấn "Quét mã mới"trong QRCodeManagementDialog
2. Hệ thống navigate đến QRCodeScannerScreen
3. Hệ thống khởi động camera (CameraX)
4. Hệ thống hiển thị camera preview + khung hình quét + text hướng dẫn
5. User đưa mã QR/Barcode vào khung hình
6. Hệ thống phân tích frame bằng ML Kit (BarcodeScanning)
7. ML Kit phát hiện barcode
8. Hệ thống trích xuất: `codeValue`, `codeType` ("QR"hoặc "BARCODE")
9. Hệ thống dừng camera
10. Hệ thống hiển thị "Đã quét: [codeValue]"
11. Hệ thống yêu cầu user đặt tên (TextField)
12. User nhập tên (VD: "Mã tủ lạnh")
13. User nhấn "Lưu"
14. Hệ thống gọi QRCodeViewModel.saveQRCode(name, codeValue, codeType)
15. QRCodeViewModel kiểm tra số lượng QR hiện tại (AppDao.getQRCodeCount())
16. **NẾU** < 5:
    (a) QRCodeViewModel kiểm tra trùng lặp (AppDao.getQRCodeByValue(codeValue))
    (b) **NẾU** chưa tồn tại:
    - Hệ thống lưu QRCodeEntity (name, codeValue, codeType, createdAt)
    - Hệ thống hiển thị "Đã lưu mã thành công"
    - Hệ thống navigate back → QRCodeManagementDialog
    - Dialog hiển thị danh sách QR (có mã mới)
17. User tick chọn mã vừa lưu
18. User nhấn "Xong"
19. Hệ thống cập nhật `selectedQRCodeIds` trong AlarmSettingsViewModel
20. Hệ thống quay về AlarmSettingsScreen

Luồng thay thế:
7a. Không phát hiện được mã:

- Hệ thống tiếp tục quét (loop bước 6–7)
- User điều chỉnh vị trí/góc độ

15a. Đã đủ 5 mã:

- Hiển thị "Bạn chỉ có thể lưu tối đa 5 mã. Vui lòng xóa bớt mã cũ."
- Quay về QRCodeManagementDialog

16.2a. Mã đã tồn tại:

- Hiển thị "Mã này đã được lưu với tên "[existingName]"@
- User có thể chọn mã cũ đó thay vì quét mới

13a. User nhấn "Hủy":

- Hệ thống bỏ mã vừa quét
- Quay về QRCodeManagementDialog

Luồng ngoại lệ:

- **E1:** Thiếu quyền Camera → Hiển thị "Ứng dụng cần quyền Camera để quét mã", mở Settings

- **E2:** Camera không hoạt động → Hiển thị "Không thể mở camera. Vui lòng kiểm tra thiết bị."

Hậu điều kiện:

- Mã QR mới được lưu vào bảng `qr_codes`
- Mã QR đã được chọn cho báo thức hiện tại (trong memory, chưa lưu DB)

1.3.5 UC21: Xem thống kê tuần

Tên: Xem thống kê tuần
Mô tả: User xem biểu đồ độ chính xác trả lời 7 ngày gần nhất.
Tác nhân: Người dùng (chính)
Tiền điều kiện:

- User đã trả lời ít nhất 1 câu hỏi (có dữ liệu trong bảng `history`)

Luồng chính:

1. User nhấn tab "Thống kê"trên Bottom Navigation
2. MainScreen navigate đến StatsScreen
3. StatsScreen observe StatsViewModel.weeklyAccuracy (StateFlow)
4. StatsViewModel gọi StatisticsDao.getWeeklyAccuracy(sevenDaysAgo)
5. StatisticsDao thực thi SQL query:

```
SELECT date(answeredAt/1000, 'unixepoch', 'localtime') as day,
       SUM(CASE WHEN isCorrect = 1 THEN 1 ELSE 0 END) as correct,
       COUNT(*) as total
FROM history
WHERE answeredAt > (NOW() - 7 days)
GROUP BY day
ORDER BY day ASC
```

6. StatisticsDao trả về Flow<List<DailyStat»
7. StatsViewModel xử lý dữ liệu:
   - Tạo list 7 ngày (từ 6 ngày trước đến hôm nay)
   - Với mỗi ngày: NẾU có data thì `accuracy = correct / total`, NGƯỢC LẠI `accuracy = 0`
   - Format nhãn: "Nay", "06", "05"...
8. StatsViewModel emit Flow với List<Pair<String, Float»
9. StatsScreen collect Flow
10. StatsScreen vẽ LineChart (biểu đồ đường) với 7 điểm (trục X: nhãn ngày, trục Y: % đúng 0–100)
11. User xem biểu đồ

Đồng thời hiển thị:

- **SRS Distribution (Pie Chart):** Query GROUP BY status (New/Learning/Mastered), vẽ biểu đồ tròn 3 phần
- **Wake-up Score:** Lấy 5 lần `alarm_history` gần nhất, tính điểm: `100 - (snoozeCount*10) - (delayMinutes*0.5)`, hiển thị điểm trung bình

- **User Stats:** Hiển thị Tổng điểm, Streak hiện tại, Streak tốt nhất, Tổng báo thức đã tắt

Luồng thay thế:
5a. Không có dữ liệu history:

- Query trả về list rỗng
- StatsViewModel trả về 7 ngày với `accuracy = 0`
- Biểu đồ hiển thị đường thẳng ở 0%
- Hiển thị text "Chưa có dữ liệu. Hoàn thành Quiz đầu tiên để xem thống kê!"

Hậu điều kiện:

- User thấy biểu đồ và các chỉ số thống kê

Chương 2

Thiết kế

2.1 Thiết kế Use Case

Phần này mô tả thiết kế chi tiết cho các use case quan trọng nhất, tập trung vào luồng tương tác giữa các lớp (UI/Screen → ViewModel/Control → Repository/Dao → DB/Service).

2.1.1 Sequence Diagram - UC01: Tạo báo thức mới

Luồng tương tác khi user tạo báo thức mới:

1. **User → AlarmScreen:** Nhấn "Thêm báo thức"
2. **AlarmScreen → AlarmSettingsScreen:** Navigate với `alarmId = -1`
3. **AlarmSettingsScreen → AlarmSettingsViewModel:** Khởi tạo ViewModel, gọi `init()`
4. **AlarmSettingsViewModel:** Điền giá trị mặc định, emit `uiState`
5. **User → AlarmSettingsScreen:** Nhập thông tin (giờ/phút/label/days), chọn Mission/QR
6. **AlarmSettingsScreen → AlarmSettingsViewModel:** Gọi `updateHour()`, `updateMinute(`
   `updateMission()`, `updateSelectedQRCodes()`
7. **User → AlarmSettingsScreen:** Nhấn "Lưu"
8. **AlarmSettingsScreen → AlarmSettingsViewModel:** Gọi `saveAlarm()`
9. **AlarmSettingsViewModel → AppDao:** Gọi `insertAlarm(alarmEntity)` (Coroutine)
10. **AppDao → Room Database:** INSERT INTO `alarms` VALUES (...)
11. **Room Database → AppDao:** Trả về `newAlarmId`
12. **AlarmSettingsViewModel → AppDao:** Gọi `clearSelectedQuestionsForAlarm(alarmId`
    `clearAlarmTopicLinks(alarmId)`
13. **AlarmSettingsViewModel → AppDao:** Loop: `insertAlarmTopicLink(alarmId, topicId)` cho mỗi topic đã chọn
14. **AlarmSettingsViewModel → AppDao:** Loop: `insertSelectedQuestion(alarmId, questionId)` cho mỗi câu hỏi lẻ
15. **AlarmSettingsViewModel → AppDao:** Loop: `insertAlarmQRLink(alarmId, qrId)` cho mỗi QR đã chọn
16. **AlarmSettingsViewModel → AlarmScheduler:** Gọi `schedule(alarmEntity)`
17. **AlarmScheduler → Android AlarmManager:** Gọi `setAlarmClock(time, pendingIntent` với `requestCode = alarmId`
18. **AlarmScheduler → AlarmSettingsViewModel:** Trả về Success
19. **AlarmSettingsViewModel:** Cập nhật `uiState.isSaved = true`

20. **AlarmSettingsScreen → User:** Hiển thị "Đã lưu", navigate back
21. **AlarmScreen → User:** Hiển thị danh sách (có báo thức mới)

```
@startuml
'** Sequence Diagram - UC01: ạTo báo úthc ớmi **

title Sequence Diagram: UC01 - ạTo báo úthc ớmi

actor "ườNgi dùng" as User
participant "AlarmScreen" as Screen
participant "AlarmSettingsScreen" as SettingsScreen
participant "AlarmSettingsViewModel" as VM
participant "AppDao" as Dao
participant "AlarmScheduler" as Scheduler
participant "AlarmManager" as SysAlarm

User -> Screen : ấNhn "Thêm báo úthc"
activate Screen

Screen -> SettingsScreen : navigate(alarmId = -1)
activate SettingsScreen

SettingsScreen -> VM : init()
activate VM
VM -> VM : Đền giá ịtr ặmc định
VM -> VM : updateTimeUntilAlarm()
VM --> SettingsScreen : uiState
deactivate VM

SettingsScreen --> User : ểHin ịth form ấcu hình
deactivate SettingsScreen

User -> SettingsScreen : ậNhp ờgi/phút/nhãn
activate SettingsScreen
SettingsScreen -> VM : updateHour(hour)
activate VM
VM -> VM : updateTimeUntilAlarm()
VM --> SettingsScreen : uiState updated
deactivate VM
deactivate SettingsScreen

User -> SettingsScreen : ọChn ngày ặlp ại
activate SettingsScreen
SettingsScreen -> VM : toggleDay(dayCode)
activate VM
VM --> SettingsScreen : uiState updated
deactivate VM
deactivate SettingsScreen

opt ọChn Mission
    User -> SettingsScreen : ấNhn "ọChn câu ỏhi"
    activate SettingsScreen
    SettingsScreen -> VM : ểHin ịth MissionDialog
    User -> VM : ọChn topics/questions
    VM -> VM : updateMission(count, questions, topics)
    VM --> SettingsScreen : selectedQuestions updated
    deactivate SettingsScreen
end

opt ọChn QR Code
    User -> SettingsScreen : ấNhn "ọChn QR"
    activate SettingsScreen
    SettingsScreen -> VM : ểHin ịth QRDialog
    User -> VM : ọChn QR codes
    VM -> VM : updateSelectedQRCodes(qrIds)
    VM --> SettingsScreen : selectedQRCodeIds updated
    deactivate SettingsScreen
end
```

2.1.2 Sequence Diagram - UC13+UC14: Báo thức reo và Quiz

Luồng tương tác khi báo thức đổ chuông và user làm quiz (luồng phức tạp nhất):

Giai đoạn 1: Kích hoạt báo thức

1. **Android AlarmManager → AlarmReceiver:** Gửi broadcast `onReceive(intent)` đúng giờ
2. **AlarmReceiver → AppDao:** Gọi `getAlarmById(alarmId)` (Coroutine với `goAsync()`)
3. **AppDao → Room Database:** SELECT * FROM `alarms` WHERE `alarmId = ?`
4. **Room Database → AppDao:** Trả về `AlarmEntity`
5. **AlarmReceiver:** Kiểm tra `daysOfWeek`:
   - Nếu có lặp lại: Gọi `AlarmScheduler.schedule(alarm)` → đặt lịch tiếp theo
   - Nếu 1 lần: Gọi `AppDao.updateAlarmEnabledStatus(alarmId, false)`
6. **AlarmReceiver → AlarmService:** Gọi `startForegroundService(intent)`
7. **AlarmService:** Khởi động, gọi `createNotificationChannel()`, `playAlarmSound(ringtoneU` (MediaPlayer loop)
8. **AlarmService → AppDao:** Gọi `getQRLinkCountForAlarm(alarmId)` để kiểm tra có QR không
9. **AlarmService:** Tạo notification với `fullScreenIntent` → khởi động AlarmRingingActivity
10. **AlarmRingingActivity → User:** Hiển thị màn hình reo + phát nhạc

Giai đoạn 2: User chọn tắt và làm Quiz

1. **User → AlarmRingingActivity:** Nhấn "Tắt"
2. **AlarmRingingActivity:** Kiểm tra điều kiện: Nếu có QR thì yêu cầu quét, nếu có Quiz thì navigate → QuizScreen
3. **QuizScreen → QuizViewModel:** Gọi `setAlarmId(alarmId)`
4. **QuizViewModel → AppDao:** Gọi `getAlarmById(alarmId)` để lấy `questionCount`
5. **QuizViewModel → AppDao:** Gọi `insertAlarmHistory(alarmHistory)` → nhận `alarmHistoryId`
6. **QuizViewModel → QuestionAlgorithmManager:** Gọi `generateMissionQuestions(alar count)`
7. **QuestionAlgorithmManager → AppDao:** Gọi `getSelectedQuestionsForAlarmOnce(ala`, `getQuestionsFromLinkedTopics(alarmId)`, `getProgressForQuestions(questionIds)`
8. **QuestionAlgorithmManager:** Tính điểm ưu tiên, sắp xếp, chọn Top N
9. **QuestionAlgorithmManager → QuizViewModel:** Trả về `List<MissionQuestion>`
10. **QuizViewModel:** Chuyển đổi sang `QuestionData`, khởi tạo timer, emit `uiState`
11. **QuizScreen → User:** Hiển thị câu hỏi đầu tiên + timer 15s

Giai đoạn 3: Loop trả lời câu hỏi

1. **User → QuizScreen:** Chọn đáp án
2. **QuizScreen → QuizViewModel:** Gọi `onOptionSelected(answerId)`
3. **QuizViewModel:** Dừng timer, kiểm tra đáp án
4. **QuizViewModel → QuestionAlgorithmManager:** Gọi `processAnswer(questionId, isCorrect, timeMs, alarmHistoryId)`
5. **QuestionAlgorithmManager → AppDao:** Gọi `insertHistory(historyEntity)`, `getProgressForQuestions([questionId])`
6. **QuestionAlgorithmManager:** Tính toán SRS mới:

- Nếu đúng: `correctStreak++`, `easinessFactor += 0.1`, `interval *= easinessFactor`
- Nếu sai: `correctStreak = 0`, `easinessFactor -= 0.2`, `interval = 1`

7. **QuestionAlgorithmManager → AppDao:** Gọi `updateQuestionProgress(progress)`, `updateTopicStats(topicStats)`
8. **QuizViewModel:** Cập nhật `correctCount`, kiểm tra điều kiện hoàn thành
9. **NẾU** chưa đủ: Chuyển câu tiếp theo, emit `uiState`, startTimer() → Quay lại bước 1
10. **NẾU** đã đủ: Tiếp tục giai đoạn 4

Giai đoạn 4: Hoàn thành và tắt

1. **QuizViewModel → AppDao:** Gọi `updateAlarmHistory(dismissalTime = NOW(), isDismissed = true)`
2. **QuizViewModel → AppDao:** Gọi `updateUserStats(...)` (cập nhật points, streak)
3. **QuizScreen → AlarmService:** Gọi `stopService()`
4. **AlarmService:** Dừng MediaPlayer, release, stopForeground()
5. **QuizScreen:** Navigate back, finish AlarmRingingActivity
6. **User:** Quay về MainActivity (màn hình chính)

```
@startuml
'** Sequence Diagram - UC13 & UC14: ậNhn báo ứthc reo và åTr ởli Quiz **

title Sequence Diagram: UC13+UC14 - Báo ứthc reo và Quiz

actor "ườNgi dùng" as User
participant "AlarmManager" as SysAlarm
participant "AlarmReceiver" as Receiver
participant "AppDao" as Dao
participant "AlarmScheduler" as Scheduler
participant "AlarmService" as Service
participant "AlarmRingingActivity" as RingingAct
participant "QuizScreen" as QuizScr
participant "QuizViewModel" as QuizVM
participant "QuestionAlgorithmManager" as AlgoMgr
participant "MediaPlayer" as Player

== Kích ạhot Báo ứthc ==

SysAlarm -> Receiver : onReceive(intent)
activate Receiver

Receiver -> Dao : getAlarmById(alarmId)
activate Dao
Dao --> Receiver : AlarmEntity
deactivate Dao

alt Báo ứthc ặlp ạli (có daysOfWeek)
    Receiver -> Scheduler : schedule(alarm)
    activate Scheduler
    Scheduler -> SysAlarm : setAlarmClock(nextTime)
    Scheduler --> Receiver : OK
    deactivate Scheduler
else Báo ứthc 1 ần
    Receiver -> Dao : updateAlarmEnabledStatus(alarmId, false)
    activate Dao
    Dao --> Receiver : OK
    deactivate Dao
end

Receiver -> Service : startForegroundService(intent)
activate Service

Service -> Service : createNotificationChannel()
Service -> Service : playAlarmSound(ringtoneUri)
activate Player
Player --> Service : Đang phát ạnhc
deactivate Player

Service -> Dao : getQRLinkCountForAlarm(alarmId)
activate Dao
Dao --> Service : qrCodeCount
deactivate Dao

Service -> Service : createNotification(hasQRCodes)

Service -> RingingAct : startActivity(fullScreenIntent)
activate RingingAct

RingingAct --> User : ểHin ịth màn hình\nbáo ứthc reo + phát ạnhc
deactivate RingingAct

deactivate Service
deactivate Receiver

== ườNgi dùng ochn ắtt báo ứthc ==
```

2.1.3 Sequence Diagram - UC04: Bật/Tắt báo thức

Luồng tương tác khi user toggle switch:

1. **User → AlarmScreen:** Toggle switch của báo thức
2. **AlarmScreen → AlarmViewModel:** Gọi `toggleAlarm(alarmId, isEnabled)`
3. **AlarmViewModel → AppDao:** Gọi `getAlarmById(alarmId)`
4. **AppDao → Room Database:** SELECT * FROM `alarms` WHERE `alarmId = ?`
5. **Room Database → AppDao:** Trả về `AlarmEntity`
6. **AlarmViewModel:** Tạo `updatedAlarm = alarm.copy(isEnabled = isEnabled)`
7. **AlarmViewModel → AppDao:** Gọi `updateAlarm(updatedAlarm)`
8. **AppDao → Room Database:** UPDATE `alarms` SET `isEnabled = ?` WHERE `alarmId = ?`
9. **NẾU** `isEnabled = true`:
   (a) **AlarmViewModel → AlarmScheduler:** Gọi `schedule(updatedAlarm)`
   (b) **AlarmScheduler:** Tính thời gian đổ chuông, tạo PendingIntent
   (c) **AlarmScheduler → Android AlarmManager:** Gọi `setAlarmClock(time, pendingIntent)`
   (d) **AlarmViewModel:** Emit `uiState` updated
   (e) **AlarmScreen → User:** Switch màu xanh + "Báo thức sẽ reo sau X giờ Y phút"
10. **NẾU** `isEnabled = false`:
    (a) **AlarmViewModel → AlarmScheduler:** Gọi `cancel(updatedAlarm)`
    (b) **AlarmScheduler:** Tạo PendingIntent với cùng `requestCode`
    (c) **AlarmScheduler → Android AlarmManager:** Gọi `cancel(pendingIntent)`
    (d) **AlarmViewModel:** Emit `uiState` updated
    (e) **AlarmScreen → User:** Switch màu xám

```
@startuml
'** Sequence Diagram - UC04: ậBt/ắTt báo úthc **

title Sequence Diagram: UC04 - ậBt/ắTt báo úthc

actor "ườNgi dùng" as User
participant "AlarmScreen" as Screen
participant "AlarmViewModel" as VM
participant "AppDao" as Dao
participant "AlarmScheduler" as Scheduler
participant "AlarmManager" as SysAlarm

User -> Screen : Toggle switch ủca báo úthc
activate Screen

Screen -> VM : toggleAlarm(alarmId, isEnabled)
activate VM

VM -> Dao : getAlarmById(alarmId)
activate Dao
Dao --> VM : AlarmEntity
deactivate Dao

VM -> VM : updatedAlarm = alarm.copy(isEnabled = isEnabled)

VM -> Dao : updateAlarm(updatedAlarm)
activate Dao
Dao -> Dao : UPDATE alarms SET isEnabled = ?
Dao --> VM : Success
deactivate Dao

alt isEnabled = true (ậBt báo úthc)
    VM -> Scheduler : schedule(updatedAlarm)
    activate Scheduler

    Scheduler -> Scheduler : Tính ờthi gian đổ chuông
    note rightế
        Nu ờgi/phút < ệhin ạti
        thì đặt sang ngày mai
    end note

    Scheduler -> SysAlarm : setAlarmClock(time, pendingIntent)
    activate SysAlarm
    note right
        PendingIntent.requestCode = alarmIdĐả
        m ảbo ỗmi alarm có intent riêng
    end note
    SysAlarm --> Scheduler : Đã đặt ịlch ẹhn
    deactivate SysAlarm

    Scheduler --> VM : Success
    deactivate Scheduler

    VM --> Screen : uiState updated
    Screen --> User : Switch ậbt (màu xanh)\n"Báo úthc ẽs reo sau X ờgi Y
        phút"

else isEnabled = false (ắTt báo úthc)
    VM -> Scheduler : cancel(updatedAlarm)
    activate Scheduler

    Scheduler -> Scheduler : createPendingIntent(alarm)
    note rightạ
        To PendingIntent ớvi
        cùng requestCode = alarmId
    end note
```

2.1.4 Tổng quan các Sequence Diagram khác

Ngoài các sequence diagram đã mô tả chi tiết, hệ thống còn có các sequence diagram khác:

- **UC08 - Thêm câu hỏi vào chủ đề:** User → TopicScreen → TopicDetailScreen → TopicDetailViewModel → AppDao (insertQuestion)
- **UC18+UC19 - Quét và liên kết QR Code:** User → QRCodeManagementDialog → QRCodeScannerScreen → CameraPreview → ML Kit → QRCodeViewModel → AppDao
- **UC21 - Xem thống kê tuần:** User → StatsScreen → StatsViewModel → StatisticsDao (query phức tạp với GROUP BY)

Tất cả các sequence diagram đều tuân theo kiến trúc MVVM:

- **UI Layer:** Screen (Composable) quan sát StateFlow từ ViewModel
- **ViewModel Layer:** Xử lý logic UI, gọi Repository/Dao, emit state
- **Logic Layer:** Business logic (AlarmScheduler, QuestionAlgorithmManager, AlarmService)
- **Data Layer:** Dao/Repository truy cập Room Database

2.2 Thiết kế cơ sở dữ liệu

2.2.1 Mô hình quan hệ

Hệ thống sử dụng Room Database (SQLite) với 12 bảng chính, phản ánh trực tiếp từ các Entity class. Database name: `app_database`, version: 2.

Các bảng chính:

1. **alarms:** Lưu thông tin báo thức (PK: alarmId)
2. **topics:** Lưu chủ đề (PK: topicId)
3. **questions:** Lưu câu hỏi (PK: questionId, FK: ownerTopicId)
4. **qr_codes:** Lưu mã QR (PK: qrId)
5. **alarm_topic_link:** Junction table (PK: alarmId + topicId)
6. **alarm_selected_questions:** Liên kết alarm-question lẻ (PK: selectionId)
7. **alarm_qr_link:** Junction table (PK: alarmId + qrId)
8. **question_progress:** Tiến độ SRS (PK: questionId)
9. **topic_stats:** Thống kê chủ đề (PK: topicId)
10. **history:** Lịch sử trả lời (PK: historyId, FK: questionId, alarmHistoryId)
11. **alarm_history:** Lịch sử báo thức (PK: historyId, FK: alarmId)
12. **UserStats:** Thống kê người dùng (PK: userId)

Quan hệ chính:

- ALARMS (1) (N) ALARM_TOPIC_LINK (N) TOPICS (1): Many-to-Many
- ALARMS (1) (N) ALARM_SELECTED_QUESTIONS: One-to-Many
- ALARMS (1) (N) ALARM_QR_LINK (N) QR_CODES (1): Many-to-Many
- ALARMS (1) (N) ALARM_HISTORY: One-to-Many
- TOPICS (1) (N) QUESTIONS: One-to-Many
- QUESTIONS (1) (1) QUESTION_PROGRESS: One-to-One (optional)
- QUESTIONS (1) (N) HISTORY: One-to-Many
- ALARM_HISTORY (1) (N) HISTORY: One-to-Many

Ràng buộc toàn vẹn:

- **PK:** Tất cả bảng có Primary Key (auto-increment hoặc composite)
- **FK:** Foreign Key với `ON DELETE CASCADE` (xóa parent tự động xóa child)
- **NOT NULL:** Các trường quan trọng (topicName, prompt, codeValue, hour, minute)
- **UNIQUE:** topicName (trong TOPICS), codeValue (trong QR_CODES)
- **INDEX:** Tự động tạo index cho FK, thêm index thủ công cho các cột hay query

```
%% ERD - Entity Relationship Diagram
%% Mô hình ữd ệliu ủca ệh ốthng Báo úthc Thông minh

erDiagram
    ALARMS ||--o{ ALARM_TOPIC_LINK : "có"
    ALARMS ||--o{ ALARM_SELECTED_QUESTIONS : "ọchn"
    ALARMS ||--o{ ALARM_QR_LINK : "ửs ụdng"
    ALARMS ||--o{ ALARM_HISTORY : "ạto ra"

    TOPICS ||--o{ ALARM_TOPIC_LINK : đượ"c ọchn ởbi"
    TOPICS ||--o{ QUESTIONS : "ứcha"
    TOPICS ||--o| TOPIC_STATS : "có"

    QUESTIONS ||--o{ ALARM_SELECTED_QUESTIONS : đượ"c ọchn"
    QUESTIONS ||--o| QUESTION_PROGRESS : "có"
    QUESTIONS ||--o{ HISTORY : "ạto ịlch ửs"

    QR_CODES ||--o{ ALARM_QR_LINK : đượ"c dùng ởbi"

    ALARM_HISTORY ||--o{ HISTORY : "ứcha"

    USER_STATS ||--|| USER : "ộthuc ềv"

    ALARMS {
        int alarmId PK
        int hour
        int minute
        string label
        set daysOfWeek
        int questionCount
        boolean isEnabled
        string ringtoneUri
        int snoozeDuration
        boolean snoozeEnabled
    }

    TOPICS {
        int topicId PK
        string topicName
    }

    QUESTIONS {
        int questionId PK
        int ownerTopicId FK
        string prompt
        list options
        string correctAnswer
    }

    ALARM_TOPIC_LINK {
        int alarmId PK_FK
        int topicId PK_FK
    }

    ALARM_SELECTED_QUESTIONS {
        int selectionId PK
        int alarmId FK
        int questionId
        int topicId
    }

    QR_CODES {
        int qrId PK
        string name
        string codeValue
        string codeType
```

2.2.2 Mapping Class Bảng

Entity Class → Table: Mapping 1:1 trực tiếp, mỗi data class có annotation `@Entity(tableName = "...")` ánh xạ tới bảng.

Bảng mapping chi tiết:

Class (Kotlin)	Table (SQLite)	TypeConverter
AlarmEntity	alarms	Set<String> String
TopicEntity	topics	-
QuestionEntity	questions	List<String> JSON
QRCodeEntity	qr_codes	-
AlarmTopicLink	alarm_topic_link	-
AlarmSelectedQuestionEntity	alarm_selected_questions	-
AlarmQRLinkEntity	alarm_qr_link	-
QuestionProgressEntity	question_progress	Date Long
TopicStatsEntity	topic_stats	-
HistoryEntity	history	Date Long
AlarmHistoryEntity	alarm_history	Date Long
UserStatsEntity	UserStats	-

TypeConverter chi tiết:

- **Set<String> String:** `daysOfWeek` lưu dạng "T2,T3,T4". Converter: `joinToString(",")` và `split(",").toSet()`.
- **List<String> String JSON:** `options` lưu dạng JSON array (`["A", "B", "C"]`). Converter: Gson `toJson()` và `fromJson()`.
- **Date Long:** Các trường thời gian (`answeredAt`, `nextReviewDate`, `scheduledTime`...) lưu dạng timestamp (milliseconds). Converter: `date.time` và `Date(timestamp)`.

ViewModel Entity:

ViewModel quản lý UI state (ephemeral data), không lưu trực tiếp vào DB. Khi save, ViewModel convert UI state sang Entity:

```
// UI State (trong ViewModel)
data class AlarmSettingData(
    val id: Int,
    val hour: Int,
    val minute: Int,
    val label: String,
    val daysOfWeek: Set<String>,
    val selectedQuestions: List<MissionQuestion>,
    val selectedQRCodeIds: List<Int>,
    ...
)

// Convert sang Entity trước khi lưu DB
val alarmEntity = AlarmEntity(
    alarmId = if (state.id == -1) 0 else state.id,
    hour = state.hour,
    minute = state.minute,
    label = state.label,
```

```
    daysOfWeek = state.daysOfWeek,
    questionCount = state.questionCount,
    isEnabled = true,
    ringtoneUri = state.ringtoneUri,
    ...
)
alarmDao.insertAlarm(alarmEntity)
```

Class Diagram thiết kế:

```
@startuml
'** Design Class Diagram - Mô hình ếthit ếk chi ếtit **

title Design Class Diagram - ệH ốthng Báo ứthc Thông minh

skinparam classAttributeIconSize 0

package "UI Layer" {
    class AlarmScreen {
        - viewModel: AlarmViewModel
        + displayAlarmList()
        + onToggleAlarm(id, enabled)
        + onDeleteAlarm(id)
    }

    class AlarmSettingsScreen {
        - viewModel: AlarmSettingsViewModel
        + displaySettings()
        + onSaveAlarm()
        + onSelectMission()
        + onSelectQR()
    }

    class QuizScreen {
        - viewModel: QuizViewModel
        + displayQuestion()
        + onAnswerSelected(answerId)
        + showTimer()
    }

    class TopicScreen {
        - viewModel: TopicViewModel
        + displayTopicList()
        + onAddTopic()
        + onSearchTopic(query)
    }

    class StatsScreen {
        - viewModel: StatsViewModel
        + displayWeeklyChart()
        + displaySrsDistribution()
        + displayWakeUpScore()
    }
}

package "ViewModel Layer" {
    class AlarmViewModel {
        - alarmDao: AppDao
        - alarmScheduler: AlarmScheduler
        + alarms: StateFlow<List<AlarmData>>
        + toggleAlarm(id, enabled)
        + deleteAlarm(id)
        + findNextAlarmTime()
    }

    class AlarmSettingsViewModel {
        - alarmDao: AppDao
        + uiState: StateFlow<AlarmSettingData>
        + loadAlarm()
        + saveAlarm()
        + updateMission()
        + updateSelectedQRCodes()
    }

    class QuizViewModel {
        - dao: AppDao
```

Design Class Diagram thể hiện đầy đủ các lớp trong 4 tầng:

- **UI Layer:** AlarmScreen, AlarmSettingsScreen, QuizScreen, TopicScreen, StatsScreen
- **ViewModel Layer:** AlarmViewModel, AlarmSettingsViewModel, QuizViewModel, TopicViewModel, StatsViewModel, QRCodeViewModel
- **Logic Layer:** AlarmScheduler, AlarmReceiver, AlarmService, QuestionAlgorithmManager
- **Data Layer:** AppDao, StatisticsDao, AppDatabase, các Entity classes

Quan hệ giữa các lớp:

- UI → ViewModel (aggregation)
- ViewModel → Logic (dependency)
- ViewModel → Data (dependency)
- Logic → Data (dependency)
- Logic Logic (dependency)
- Database provides Dao (realization)

Giả định/TBD

- **Giả định 1:** Người dùng luôn cấp quyền `CAMERA`, `POST_NOTIFICATIONS`, `SCHEDULE_EXACT_ALAR` Thực tế cần xử lý từ chối quyền tốt hơn.
- **Giả định 2:** Chỉ có 1 người dùng (userId = 1). Không hỗ trợ multi-profile.
- **Giả định 3:** Thuật toán SRS sử dụng công thức đơn giản dựa trên SM-2. Chưa tính forgetting curve chi tiết.
- **TBD 1:** Chưa có sequence diagram cho tất cả 24 use cases. Đã mô tả chi tiết 4 use cases quan trọng nhất (UC01, UC04, UC13+UC14, UC21).
- **TBD 2:** Chưa có backup/restore dữ liệu. Cần implement Export/Import trong phiên bản tiếp theo.
- **TBD 3:** Chưa hỗ trợ câu hỏi dạng hình ảnh. Cần thêm `imageUri` vào QuestionEntity.