TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP THÀNH PHỐ HỒ CHÍ MINH

KHOA CÔNG NGHỆ THÔNG TIN

ĐỀ THI THỰC HÀNH
Môn thi : Lập trình thiết bị di động
Lớp/Lớp học phần: DHKTPM18A
Ngày thi: 15/11/2025
Thời gian làm bài: 75 phút
(Không kể thời gian phát đề)

STT: *MSSV:* *Họ và tên SV:* *Số máy:*

Đề 1: ỨNG DỤNG "GROCERY LIST" (DANH SÁCH ĐI CHỢ)
==========

Tạo ứng dụng "Grocery List" (Expo/React Native) quản lý danh sách món cần mua khi đi chợ/siêu thị.
Ứng dụng lưu dữ liệu offline bằng SQLite và có chức năng đồng bộ một lần từ API mẫu (danh sách gợi ý các món).
App có 1 màn hình chính (danh sách) và 1 modal để thêm/sửa.

Cấu trúc DB đề xuất:

```
grocery_items(
  id INTEGER PRIMARY KEY AUTOINCREMENT,
  name TEXT NOT NULL,
  quantity INTEGER DEFAULT 1,
  category TEXT,
  bought INTEGER DEFAULT 0,
  created_at INTEGER
)
```

Câu 1. (1đ) – Khởi tạo & cấu hình dự án
- Tạo app Expo mới, cài đặt expo-sqlite.
- Tạo repo Git, thực hiện initial commit.
- Tạo file db.ts (hoặc db.js) để mở kết nối SQLite.
- Chạy thử expo start và đảm bảo ứng dụng chạy được, không báo lỗi.
- Commit: feat(Q1): init expo project and sqlite setup
- Tiêu chí đạt: expo start chạy OK, đã kết nối được đến DB (không crash).

Câu 2. (1đ) – Tạo bảng và seed dữ liệu mẫu
- Khi app khởi động, tạo bảng grocery_items nếu chưa tồn tại.
- (Tùy chọn) Seed 2–3 bản ghi mẫu, ví dụ: "Sữa", "Trứng", "Bánh mì" nếu bảng đang trống.
- Commit: feat(Q2): create grocery_items table and seed sample
- Tiêu chí đạt: bảng tồn tại, khởi động app không lỗi, nếu bảng trống lần đầu sẽ có dữ liệu mẫu.

Câu 3. (1đ) – Màn hình danh sách (UI + Hooks)

- Tạo màn hình chính hiển thị danh sách các món cần mua bằng FlatList.
- Mỗi item hiển thị: name, quantity, category, trạng thái bought.
- Sử dụng useState và useEffect để đọc dữ liệu từ SQLite và render lên UI.
- Empty state: nếu không có bản ghi, hiển thị text "Danh sách trống, thêm món cần mua nhé!".
- Commit: feat(Q3): list screen with hooks and empty state
- Tiêu chí đạt: thấy danh sách hoặc empty state, app không bị crash.

Câu 4. (1đ) – Thêm mới bằng Modal (Form)

- Thêm nút "+" trên màn hình chính để mở Modal thêm mới.
- Trong Modal, cho phép nhập: name (bắt buộc), quantity (mặc định 1), category (tùy chọn).
- Khi nhấn Lưu, thực hiện INSERT vào SQLite, sau đó cập nhật lại danh sách trên màn hình chính.
- Validate: name không được rỗng, nếu rỗng phải hiện cảnh báo (Alert hoặc text lỗi).
- Commit: feat(Q4): add modal and insert grocery item
- Tiêu chí đạt: thêm mới thành công, danh sách cập nhật ngay khi thêm, không cần reload thủ công.

Câu 5. (1đ) – Đánh dấu đã mua (Toggle hoàn thành – UPDATE)

- Cho phép chạm vào item để toggle trạng thái bought (0 ↔ 1) trong SQLite.
- UI: thể hiện rõ trạng thái, ví dụ: nếu đã mua thì gạch ngang tên món hoặc hiển thị icon √.
- Commit: feat(Q5): toggle bought state
- Tiêu chí đạt: toggle trơn tru, không crash, không cần reload thủ công toàn app.

Câu 6. (1đ) – Sửa món (EDIT – Modal)

- Cho phép nhấn giữ hoặc thêm nút "Sửa" trên mỗi item để mở Modal chỉnh sửa.
- Modal cho phép thay đổi name, quantity, category của item.
- Khi lưu, thực hiện UPDATE bản ghi tương ứng trong SQLite và refresh list.
- Commit: feat(Q6): edit grocery item via modal
- Tiêu chí đạt: cập nhật đúng bản ghi, UI thể hiện dữ liệu mới rõ ràng.

Câu 7. (1đ) – Xóa món (DELETE) có xác nhận

- Thêm nút xóa cho mỗi item (hoặc sử dụng gesture swipe để xóa).
- Khi bấm xóa, hiện Alert xác nhận "Bạn có chắc chắn muốn xóa món này không?".
- Nếu người dùng đồng ý, thực hiện DELETE khỏi SQLite, danh sách được cập nhật.
- Commit: feat(Q7): delete grocery item with confirm
- Tiêu chí đạt: xóa đúng item, không xóa nhầm; danh sách hiển thị chính xác sau khi xóa.

Câu 8. (1đ) – Tìm kiếm/Filter real-time (Hooks)

- Thêm TextInput Search ở phía trên danh sách.

• Khi gõ vào Search, danh sách được lọc theo name theo thời gian thực (client-side filter hoặc SQL LIKE).
• Sử dụng useMemo/useCallback hợp lý (nếu lọc client-side) để tránh render dư thừa khi có 50–100 bản ghi.
• Commit: feat(Q8): realtime search with hooks optimization
• Tiêu chí đạt: gõ là lọc ngay, scroll vẫn mượt khi có nhiều bản ghi.

Câu 9. (0.5đ) – Fetch API & Đồng bộ "Import once"
• Thêm nút "Import từ API" trên màn hình chính.
• Khi bấm, gọi GET tới endpoint API mẫu trả về danh sách món gợi ý.
• Merge dữ liệu API vào SQLite:
– Nếu name trùng hoàn toàn với bản ghi đã có thì bỏ qua (tránh trùng lặp).
– Nếu API trả về trường completed: map sang bought (true → 1, false → 0).
• UI: có state loading khi đang fetch, hiển thị lỗi nếu fetch thất bại.
• Commit: feat(Q9): import grocery items from API with merge and states
• Tiêu chí đạt: sau khi import, danh sách tăng thêm các item mới hợp lệ; có loading/error state rõ ràng.

Câu 10. (0.5đ) – Tách custom hook + hoàn thiện UI/UX
• Tạo custom hook, ví dụ: useGroceryItems, đóng gói logic:
– load list, insert, update, delete, search, import.
• Dọn dẹp dependencies của useEffect, dùng useCallback cho các hàm thao tác DB.
• Nâng cấp UI/UX:
– Thêm Pull to refresh để reload danh sách.
– Thiết kế Empty state đẹp hơn (icon, màu sắc, mô tả).
– Các Button có trạng thái disabled khi đang loading (ví dụ lúc import).
• Commit: feat(Q10): extract useGroceryItems hook and polish UI/UX
• Tiêu chí đạt: logic gọn trong hook; UI mượt, không warning deps trong console.

Câu 11. (1đ) – EAS build (Preview)
• Dùng EAS build tạo một bản preview cho ứng dụng Grocery List.
• Tạo một file text (ví dụ: EAS_LINK.txt) chứa link preview vừa build, đưa file này vào trong repo.
• Commit: chore(Q11): add EAS preview link text file
• Tiêu chí đạt: có thể truy cập được link preview, file text tồn tại trong repo.

--- Hết ---

Lưu ý:

- Sinh viên được sử dụng mọi nguồn tài liệu để làm bài, kể cả AI (extension, github copilot, chatgpt…)

- Tuy nhiên, không được chép bài của bạn khác, hoặc từ github.

- Gởi và nhận bài đều không được chấm điểm