ĐỀ THI THỬ - FLUTTER VỚI FIREBASE FIRESTORE

Thời gian làm bài: 120 phút
Tổng điểm: 100 điểm
Được phép: Sử dụng ChatGPT, tài liệu, Internet

MÔ TẢ DỰ ÁN

Bạn cần xây dựng một ứng dụng Flutter quản lý thư viện sách với Firebase Firestore. Ứng dụng cho phép người dùng đăng ký, đăng nhập, xem danh sách sách, mượn sách, và xem lịch sử mượn sách.

YÊU CẦU CƠ SỞ DỮ LIỆU

Cấu trúc Firestore Collections:

Collection 1: `users`

- `userId` (String, Document ID): ID duy nhất của người dùng
- `email` (String): Email đăng nhập
- `fullName` (String): Họ và tên đầy đủ
- `phoneNumber` (String): Số điện thoại
- `address` (String): Địa chỉ
- `createdAt` (Timestamp): Thời gian tạo tài khoản
- `role` (String): Vai trò ("student", "teacher", "admin")

Collection 2: `books`

- `bookId` (String, Document ID): ID duy nhất của sách
- `title` (String): Tiêu đề sách
- `author` (String): Tác giả
- `isbn` (String): Mã ISBN
- `category` (String): Thể loại ("Novel", "Science", "History", "Technology")
- `description` (String): Mô tả sách
- `totalCopies` (int): Tổng số bản copy
- `availableCopies` (int): Số bản copy còn lại (có thể mượn)
- `publishedYear` (int): Năm xuất bản
- `publisher` (String): Nhà xuất bản
- `createdAt` (Timestamp): Thời gian thêm sách vào hệ thống

Collection 3: `borrows`

- `borrowId` (String, Document ID): ID duy nhất của phiếu mượn
- `userId` (String): ID của người dùng mượn sách (reference đến `users`)
- `bookId` (String): ID của sách được mượn (reference đến `books`)
- `borrowDate` (Timestamp): Ngày mượn
- `dueDate` (Timestamp): Ngày hẹn trả

- `returnDate` (Timestamp, nullable): Ngày thực tế trả sách (null nếu chưa trả)
- `status` (String): Trạng thái ("pending", "borrowed", "returned", "overdue")
- `notes` (String, nullable): Ghi chú

Lưu ý về quan hệ:

- Một `user` có thể mượn nhiều `book` (quan hệ 1-nhiều qua `borrows`)
- Một `book` có thể được mượn bởi nhiều `user` (quan hệ nhiều-nhiều qua `borrows`)
- Khi mượn sách: `availableCopies` của `book` giảm đi 1
- Khi trả sách: `availableCopies` của `book` tăng lên 1

☑ YÊU CẦU CHỨC NĂNG

Phần 1: Thiết lập Firebase (10 điểm)

1. **Cài đặt Firebase trong Flutter** (3 điểm)
    - Thêm các package cần thiết vào `pubspec.yaml`
    - Cấu hình Firebase trong project Flutter
    - Khởi tạo Firebase trong `main.dart`
2. **Tạo Firestore Database** (4 điểm)
    - Tạo Firebase project trên Firebase Console
    - Kích hoạt Firestore Database (chế độ Production hoặc Test)
    - Thiết lập Security Rules cơ bản (cho phép đọc/ghi với authentication)
3. **Tạo Service Class** (3 điểm)
    - Tạo file `lib/services/firebase_service.dart`
    - Tạo class `FirebaseService` với method `initializeFirebase()`

Phần 2: Tạo Model Classes (15 điểm)

1. **Model User** (5 điểm)
    - Tạo file `lib/models/user_model.dart`
    - Class `UserModel` với các field tương ứng collection `users`
    - Implement `fromMap()` và `toMap()` methods
    - Constructor với named parameters
2. **Model Book** (5 điểm)
    - Tạo file `lib/models/book_model.dart`
    - Class `BookModel` với các field tương ứng collection `books`
    - Implement `fromMap()` và `toMap()` methods
    - Constructor với named parameters
3. **Model Borrow** (5 điểm)

- Tạo file `lib/models/borrow_model.dart`
- Class `BorrowModel` với các field tương ứng collection `borrows`
- Implement `fromMap()` và `toMap()` methods
- Xử lý nullable fields (`returnDate`, `notes`)
- Constructor với named parameters

Phần 3: Repository Pattern - CRUD Operations (40 điểm)

Tạo file `lib/repositories/library_repository.dart` với class `LibraryRepository`:

3.1. User Repository (12 điểm)

1. **Thêm User** (3 điểm)
   - Method `Future<void> addUser(UserModel user)`
   - Tạo document mới trong collection `users` với document ID là `userId`
   - Set `createdAt` = thời gian hiện tại
2. **Lấy User theo ID** (2 điểm)
   - Method `Future<UserModel?> getUserById(String userId)`
   - Trả về `null` nếu không tìm thấy
3. **Lấy tất cả Users** (2 điểm)
   - Method `Future<List<UserModel>> getAllUsers()`
   - Sắp xếp theo `createdAt` giảm dần (mới nhất trước)
4. **Cập nhật User** (3 điểm)
   - Method `Future<void> updateUser(String userId, Map<String, dynamic> updates)`
   - Chỉ cập nhật các field được truyền vào
5. **Xóa User** (2 điểm)
   - Method `Future<void> deleteUser(String userId)`
   - Kiểm tra xem user có đang mượn sách không (có `borrows` với `status != "returned"`)
   - Nếu đang mượn sách, throw exception "Cannot delete user with active borrows"

3.2. Book Repository (14 điểm)

1. **Thêm Book** (3 điểm)
   - Method `Future<void> addBook(BookModel book)`
   - Tạo document mới trong collection `books`
   - Set `availableCopies = totalCopies` và `createdAt`
2. **Lấy Book theo ID** (2 điểm)
   - Method `Future<BookModel?> getBookById(String bookId)`

3. **Lấy tất cả Books** (2 điểm)
    - Method `Future<List<BookModel>> getAllBooks()`
    - Sắp xếp theo `title` tăng dần
4. **Tìm kiếm Books** (4 điểm)
    - Method `Future<List<BookModel>> searchBooks(String keyword)`
    - Tìm kiếm trong `title`, `author`, hoặc `isbn`
    - Không phân biệt hoa thường
5. **Lọc Books theo Category** (2 điểm)
    - Method `Future<List<BookModel>> getBooksByCategory(String category)`
6. **Cập nhật Book** (1 điểm)
    - Method `Future<void> updateBook(String bookId, Map<String, dynamic> updates)`

3.3. Borrow Repository (14 điểm)

1. **Mượn Sách (Borrow Book)** (5 điểm)
    - Method `Future<void> borrowBook(String userId, String bookId, int daysToBorrow)`
    - Kiểm tra `availableCopies > 0`
    - Kiểm tra user chưa mượn quá 5 sách cùng lúc (status != "returned")
    - Tạo document mới trong `borrows` với:
        - `borrowDate` = hiện tại
        - `dueDate` = `borrowDate` + `daysToBorrow` ngày
        - `status` = "borrowed"
    - Giảm `availableCopies` của book đi 1
2. **Trả Sách (Return Book)** (4 điểm)
    - Method `Future<void> returnBook(String borrowId)`
    - Cập nhật `returnDate` = hiện tại
    - Cập nhật `status` = "returned"
    - Tăng `availableCopies` của book lên 1
3. **Lấy lịch sử mượn của User** (2 điểm)
    - Method `Future<List<BorrowModel>> getBorrowsByUser(String userId)`
    - Sắp xếp theo `borrowDate` giảm dần
4. **Lấy lịch sử mượn của Book** (2 điểm)
    - Method `Future<List<BorrowModel>> getBorrowsByBook(String bookId)`
    - Sắp xếp theo `borrowDate` giảm dần
5. **Lấy sách đang mượn (chưa trả)** (1 điểm)
    - Method `Future<List<BorrowModel>> getActiveBorrows()`

- Lọc `status != "returned"`

Phần 4: UI Implementation (25 điểm)

4.1. Màn hình đăng ký/đăng nhập (5 điểm)

1. **Màn hình đăng ký** (3 điểm)
   - File `lib/screens/register_screen.dart`
   - Form với các field: email, fullName, phoneNumber, address, role (dropdown)
   - Validate input (email format, phone number format)
   - Khi submit: tạo user mới vào Firestore với `userId` tự sinh (UUID hoặc random)
   - Navigate đến màn hình chính sau khi đăng ký thành công

2. **Màn hình đăng nhập** (2 điểm)
   - File `lib/screens/login_screen.dart`
   - Form với email (hoặc userId)
   - Khi submit: lấy user từ Firestore theo email hoặc userId
   - Lưu thông tin user vào SharedPreferences
   - Navigate đến màn hình chính

4.2. Màn hình danh sách sách (8 điểm)

1. **Hiển thị danh sách sách** (4 điểm)
   - File `lib/screens/books_list_screen.dart`
   - Dùng `StreamBuilder` để lắng nghe real-time updates từ Firestore
   - Hiển thị danh sách books trong `ListView` hoặc `GridView`
   - Mỗi item hiển thị: title, author, category, availableCopies/totalCopies
   - Có AppBar với tiêu đề "Danh sách sách"

2. **Tìm kiếm và lọc** (2 điểm)
   - TextField tìm kiếm (gọi `searchBooks()` khi người dùng nhập)
   - Dropdown lọc theo category (gọi `getBooksByCategory()`)

3. **Chi tiết sách** (2 điểm)
   - Khi tap vào một book: Navigate đến `book_detail_screen.dart`
   - Hiển thị đầy đủ thông tin sách
   - Hiển thị nút "Mượn sách" (chỉ hiện nếu `availableCopies > 0`)
   - Hiển thị lịch sử mượn của sách này

4.3. Màn hình mượn sách (6 điểm)

1. **Form mượn sách** (4 điểm)
   - Trong `book_detail_screen.dart`, khi nhấn "Mượn sách":
   - Hiển thị Dialog/BottomSheet để chọn số ngày mượn (1-30 ngày)

- Gọi `borrowBook()` từ repository
- Hiển thị SnackBar thông báo thành công/thất bại
- Refresh danh sách sách để cập nhật `availableCopies`

2. **Xử lý lỗi** (2 điểm)

- Hiển thị thông báo lỗi phù hợp khi:
  - Hết sách (`availableCopies = 0`)
  - User đã mượn quá 5 sách
  - Lỗi mạng

4.4. Màn hình lịch sử mượn sách (6 điểm)

1. **Danh sách sách đang mượn** (3 điểm)

- File `lib/screens/my_borrows_screen.dart`
- Hiển thị danh sách sách mà user hiện tại đang mượn
- Mỗi item hiển thị: book title, borrowDate, dueDate, status
- Highlight nếu quá hạn (dueDate < hiện tại và status != "returned")

2. **Nút trả sách** (2 điểm)

- Mỗi item có nút "Trả sách"
- Khi nhấn: Confirm dialog, sau đó gọi `returnBook()`
- Refresh danh sách sau khi trả

3. **Lịch sử đã trả** (1 điểm)

- Tab hoặc section riêng hiển thị sách đã trả
- Hiển thị returnDate

Phần 5: Xử lý Real-time và Error Handling (10 điểm)

1. **Real-time Updates** (5 điểm)

- Sử dụng `StreamBuilder` trong màn hình danh sách sách
- Danh sách tự động cập nhật khi có thay đổi trong Firestore
- Xử lý các state: loading, error, empty, data

2. **Error Handling** (5 điểm)

- Try-catch cho tất cả các operations với Firestore
- Hiển thị thông báo lỗi user-friendly
- Xử lý các trường hợp:
  - Mất kết nối mạng
  - Permission denied
  - Document không tồn tại
  - Invalid data format

YÊU CẦU KỸ THUẬT

1. **Code Organization** (Bắt buộc)
    - Tổ chức code theo cấu trúc: `lib/models/`, `lib/services/`, `lib/repositories/`, `lib/screens/`
    - Tách biệt logic (repository) và UI (screen)
    - Sử dụng named routes hoặc Navigator.push
2. **State Management** (Khuyến khích)
    - Có thể sử dụng Provider, setState, hoặc bất kỳ state management nào đã học
    - Đảm bảo UI update khi dữ liệu thay đổi
3. **UI/UX** (Khuyến khích)
    - UI đẹp, dễ sử dụng
    - Loading indicators khi đang tải dữ liệu
    - Empty states khi không có dữ liệu
    - Confirm dialogs cho các hành động quan trọng (xóa, trả sách)

PACKAGES ĐƯỢC PHÉP SỬ DỤNG

```
dependencies:
  flutter:
    sdk: flutter
  firebase_core: ^latest
  cloud_firestore: ^latest
  shared_preferences: ^latest
  intl: ^latest  # Cho date formatting (tùy chọn)
```

CHECKLIST NỘP BÀI

- [ ] Project Flutter hoàn chỉnh, có thể chạy được
- [ ] Firebase project đã tạo và kết nối thành công
- [ ] Có ít nhất 5 users mẫu trong Firestore
- [ ] Có ít nhất 10 books mẫu trong Firestore
- [ ] Có ít nhất 5 borrows mẫu trong Firestore (cả borrowed và returned)
- [ ] Tất cả các chức năng CRUD hoạt động đúng
- [ ] UI hiển thị dữ liệu từ Firestore
- [ ] Real-time updates hoạt động
- [ ] Error handling đầy đủ
- [ ] Code được tổ chức rõ ràng, có comment cần thiết
- [ ] File README.md mô tả cách chạy project

CÁCH NỘP BÀI

1. Nén toàn bộ project Flutter thành file `.zip`
2. Export Firestore data (hoặc screenshot) để chứng minh có dữ liệu
3. Gửi kèm file `README.md` hướng dẫn chạy project
4. Gửi link GitHub (nếu có) để code review

💡 LƯU Ý

- **Không copy code từ nhau** - Giám khảo sẽ kiểm tra similarity
- **Chú ý Security Rules** - Đảm bảo Firestore rules phù hợp với logic ứng dụng
- **Test kỹ các trường hợp edge cases** - Hết sách, user mượn quá nhiều, mất mạng, v.v.
- **Code sạch và có comment** - Dễ đọc, dễ maintain
- **Sử dụng ChatGPT hợp lý** - Hiểu code bạn viết, không chỉ copy-paste

---

Chúc các bạn làm bài tốt! 🚀