รายงานการเชื่อมต่อฐานข้อมูล (Database Connection)

SSID Application (Suturing Skills Identification Device)

ชื่อ-นามสกุล: [กรอกชื่อ-นามสกุล] **รหัสนักศึกษา:** [กรอกรหัสนักศึกษา] **รายวิชา:** Mobile Application Development **วันที่ส่ง:** 1 กุมภาพันธ์ 2569

📸 ** หลักฐาน UPDATE:**

[แทรกรูป: Database Inspector แสดงข้อมูลที่ถูกแก้ไข (is_read เปลี่ยนจาก 0 เป็น 1)]

สารบัญ

---

1. SQLite Database (2.5 คะแนน)

1.1 โครงสร้างฐานข้อมูล

ฐานข้อมูล SQLite ชื่อ `ssid_app.db` ประกอบด้วย 4 ตาราง:

ตาราง	คำอธิบาย
users	ข้อมูลผู้ใช้งาน
sessions	ผลการวิเคราะห์วิดีโอ
notifications	การแจ้งเตือน
assignments	งานที่มอบหมาย

1.2 Code การเชื่อมต่อ SQLite

ไฟล์: `lib/services/database_helper.dart`

```dart
import 'package:sqflite/sqflite.dart';
import 'package:path/path.dart';

class DatabaseHelper {
  static final DatabaseHelper instance = DatabaseHelper._init();
  static Database? _database;

  DatabaseHelper._init();

  Future<Database> get database async {
    if (_database != null) return _database!;
    _database = await _initDB('ssid_app.db');
    return _database!;
  }

  Future<Database> _initDB(String filePath) async {
    final dbPath = await getDatabasesPath();
    final path = join(dbPath, filePath);

    return await openDatabase(
      path,
      version: 1,
      onCreate: _createDB,
    );
  }
}
```

1.3 CRUD Operations - SQLite

CREATE (เพิ่มข้อมูล)

```dart
// CREATE - Insert new user
Future<int> createUser(Map<String, dynamic> user) async {
  final db = await database;
  return await db.insert('users', user);
}

// CREATE - Insert new session
Future<int> createSession(Map<String, dynamic> session) async {
  final db = await database;
  return await db.insert('sessions', session);
}
```

📸 **หลักฐาน CREATE:**

[แทรกรูป: หน้าจอ Database Inspector แสดงข้อมูลที่ถูกเพิ่มในตาราง users]

READ (อ่านข้อมูล)

```
// READ - Get user by email
Future<Map<String, dynamic>?> getUserByEmail(String email) async {
  final db = await database;
  final result = await db.query(
    'users',
    where: 'email = ?',
    whereArgs: [email],
  );
  return result.isNotEmpty ? result.first : null;
}

// READ - Get all sessions
Future<List<Map<String, dynamic>>> getAllSessions() async {
  final db = await database;
  return await db.query('sessions', orderBy: 'analyzed_at DESC');
}
```

📸 **หลักฐาน READ:**

[แทรกรูป: Database Inspector แสดงผล Query ข้อมูลจากตาราง sessions]

UPDATE (แก้ไขข้อมูล)

```
// UPDATE - Update user
Future<int> updateUser(int id, Map<String, dynamic> user) async {
  final db = await database;
  return await db.update(
    'users',
    user,
    where: 'id = ?',
    whereArgs: [id],
  );
}

// UPDATE - Mark notification as read
Future<int> markNotificationAsRead(int id) async {
  final db = await database;
  return await db.update(
    'notifications',
    {'is_read': 1},
```

```
      where: 'id = ?',
      whereArgs: [id],
    );
  }
```

📸 **หลักฐาน UPDATE:**

[แทรกรูป: Database Inspector แสดงข้อมูลที่ถูกแก้ไข (is_read เปลี่ยนจาก 0 เป็น 1)]

DELETE (ลบข้อมูล)

```
// DELETE - Delete session
Future<int> deleteSession(int id) async {
  final db = await database;
  return await db.delete(
    'sessions',
    where: 'id = ?',
    whereArgs: [id],
  );
}

// DELETE - Delete notification
Future<int> deleteNotification(int id) async {
  final db = await database;
  return await db.delete(
    'notifications',
    where: 'id = ?',
    whereArgs: [id],
  );
}
```

📸 **หลักฐาน DELETE:**

[แทรกรูป: Database Inspector ก่อนและหลังลบข้อมูล]

1.4 ภาพหน้าจอ SQLite Database

ตาราง users

[แทรกรูป: Database Inspector แสดงตาราง users]

ตาราง sessions

[แทรกรูป: Database Inspector แสดงตาราง sessions]

ตาราง notifications

[แทรกรูป: Database Inspector แสดงตาราง notifications]

ตาราง assignments

[แทรกรูป: Database Inspector แสดงตาราง assignments]

---

2. Firebase Firestore (2.5 คะแนน)

2.1 การตั้งค่า Firebase

ไฟล์ Configuration: `android/app/google-services.json`

Dependencies ใน pubspec.yaml:

```
dependencies:
  firebase_core: ^3.15.2
  cloud_firestore: ^5.6.12
  firebase_auth: ^5.7.0
```

2.2 Code การเชื่อมต่อ Firestore

ไฟล์: `lib/services/firestore_service.dart`

```
import 'package:cloud_firestore/cloud_firestore.dart';

class FirestoreService {
  final FirebaseFirestore _firestore = FirebaseFirestore.instance;

  // Reference to collections
  CollectionReference get usersCollection => _firestore.collection('users');
  CollectionReference get sessionsCollection => _firestore.collection('sessions');
  CollectionReference get notificationsCollection =>
_firestore.collection('notifications');
  CollectionReference get assignmentsCollection =>
```

```
_firestore.collection('assignments');
}
```

2.3 CRUD Operations - Firebase Firestore

CREATE (เพิ่มข้อมูล)

```
// CREATE - Add user to Firestore
Future<void> createUser(Map<String, dynamic> userData) async {
  await usersCollection.add({
    ...userData,
    'createdAt': FieldValue.serverTimestamp(),
  });
}

// CREATE - Add session to Firestore
Future<void> createSession(Map<String, dynamic> sessionData) async {
  await sessionsCollection.add({
    ...sessionData,
    'analyzedAt': FieldValue.serverTimestamp(),
  });
}
```

📸 หลักฐาน CREATE (Firebase Console):

[แทรกรูป: Firebase Console แสดง Document ที่ถูกเพิ่มใน Collection users]

READ (อ่านข้อมูล)

```
// READ - Get all users
Stream<QuerySnapshot> getUsers() {
  return usersCollection.snapshots();
}

// READ - Get sessions by user ID
Future<List<DocumentSnapshot>> getSessionsByUserId(String odby('analyzedAt',
descending: true)
    .get();
  return snapshot.docs;
}
```

📸 หลักฐาน READ (Firebase Console):

[แทรกรูป: Firebase Console แสดง Collection sessions พร้อมข้อมูล]

UPDATE (แก้ไขข้อมูล)

```
// UPDATE - Update user profile
Future<void> updateUser(String docId, Map<String, dynamic> userData) async {
  await usersCollection.doc(docId).update(userData);
}

// UPDATE - Mark notification as read
Future<void> markNotificationAsRead(String docId) async {
  await notificationsCollection.doc(docId).update({
    'isRead': true,
  });
}
```

📸 **หลักฐาน UPDATE (Firebase Console):**

[แทรกรูป: Firebase Console แสดง Field ที่ถูกแก้ไข]

DELETE (ลบข้อมูล)

```
// DELETE - Delete session
Future<void> deleteSession(String docId) async {
  await sessionsCollection.doc(docId).delete();
}

// DELETE - Delete notification
Future<void> deleteNotification(String docId) async {
  await notificationsCollection.doc(docId).delete();
}
```

📸 **หลักฐาน DELETE (Firebase Console):**

[แทรกรูป: Firebase Console ก่อนและหลังลบ Document]

2.4 ภาพหน้าจอ Firebase Firestore

Collection users

[แทรกรูป: Firebase Console แสดง Collection users]

Collection sessions

[แทรกรูป: Firebase Console แสดง Collection sessions]

Collection notifications

[แทรกรูป: Firebase Console แสดง Collection notifications]

Collection assignments

[แทรกรูป: Firebase Console แสดง Collection assignments]

---

3. สรุป

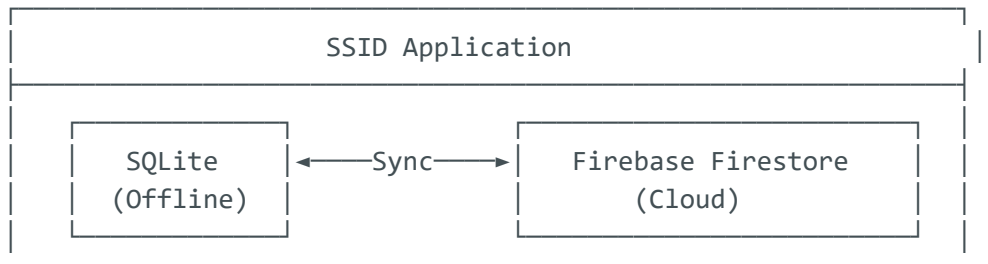
รายการที่ดำเนินการ

หัวข้อ	รายละเอียด	คะแนน
SQLite Database	เชื่อมต่อและทำ CRUD ได้ครบ 4 operations	2.5
Firebase Firestore	เชื่อมต่อและทำ CRUD ได้ครบ 4 operations	2.5
รวม		**5.0**

Hybrid Database Architecture

แอปพลิเคชันใช้วิธี Hybrid Database:

- **SQLite**: เก็บข้อมูลท้องถิ่นสำหรับใช้งาน Offline
- **Firebase Firestore**: Sync ข้อมูลขึ้น Cloud สำหรับ Backup และ Real-time updates



วิธีถ่ายรูปหลักฐาน

SQLite (Android Studio)

1. รันแอปบน Emulator
2. View → Tool Windows → App Inspection
3. เลือก Process: com.example.ssid_app_v2
4. Tab: Database Inspector
5. เปิด ssid_app.db → แต่ละ Table
6. ถ่ายรูปหน้าจอ

Firebase (Web Browser)

1. เปิด https://console.firebase.google.com/
2. เลือก Project SSID
3. Firestore Database
4. คลิกแต่ละ Collection
5. ถ่ายรูปหน้าจอ

หมายเหตุ: แทนที่ข้อความ [แทรกรูป: ...] ด้วยรูปภาพจริงก่อนส่งงาน