Flutter To-Do App

프로젝트 개요

본 애플리케이션은 Flutter 프레임워크를 기반으로 개발된 To-Do(할 일) 관리 앱입니다. GetX를 활용한 상태 관리와 모듈화된 서비스 구조를 통해 다양한 기능을 효과적으로 구현하였습니다. Firebase 인증, 로컬 데이터베이스(SQLite), SharedPreferences, GetStorage 등 다양한 기술을 통합하여 사용자에게 편리하고 풍부한 기능을 제공하며, 동적 테마 적용으로 사용자 경험을 향상시킵니다.

기술 스택

- **프레임워크**: Flutter
- **상태 관리**: GetX
- **로컬 저장소**:
  - GetStorage (테마 설정)
  - SharedPreferences (스크립트)
  - SQLite (DBHelper, 할 일 목록)
- **인증**: Firebase Authentication (Google OAuth)
- **음성 인식**: `speech_to_text` 패키지
- **UI**: Material Design, Google Fonts, `date_picker_timeline`, `flutter_staggered_animations`
- **기타**: `intl` (날짜 포맷팅), `uuid` (고유 ID 생성)

주요 기능

1. Google OAuth 로그인

- **목표**: Google 계정을 통한 간편하고 안전한 앱 로그인.
- **주요 파일**: `services/auth_service.dart`, `ui/pages/login_page.dart`, `main.dart`.
- **구현 요약**:
  - **`AuthService` (`services/auth_service.dart`)**:
    - `FirebaseAuth` 및 `GoogleSignIn` 패키지를 활용하여 Google OAuth 인증을 처리합니다.
    - `currentUser` (Rx<User?>): 현재 로그인된 사용자 정보를 반응형으로 관리합니다.
    - `signInWithGoogle()`: Google 로그인 흐름을 시작하고, 성공 시 `UserCredential`을 받아 `currentUser`를 업데이트합니다. 오류 발생 시 사용자에게 스낵바를 통해 알림을 제공합니다.
    - `signOut()`: Firebase에서 로그아웃 처리 후 `currentUser`를 `null`로 설정합니다.
    - GetX를 통해 싱글톤 서비스로 등록되어 앱 전역에서 접근 가능하도록 설계되었습니다.
  - **로그인 UI (`ui/pages/login_page.dart`)**:
    - `StatefulWidget`으로 구현되었으며, `AuthService` 인스턴스를 활용합니다.
    - `initState()`에서 현재 로그인 상태를 확인하여, 이미 로그인된 경우 `HomePage`로 자동 이동 처리합니다.
    - "Google 계정으로 로그인" 버튼을 제공하며, 클릭 시 `_signInWithGoogle()` 메서드를 호출합니다.
    - `_signInWithGoogle()`: `authService.signInWithGoogle()`을 호출하고, 로딩 상태(`_isLoading`)를 관리합니다. 로그인 결과에 따라 성공/실패 스낵바를 표시하고 `HomePage`로 이동합니다.
    - UI는 현재 앱 테마(라이트/다크)에 맞춰 동적으로 스타일링됩니다.

- **초기화 (`main.dart`)**:
  - 앱 시작 시 `Firebase.initializeApp()`을 호출하여 Firebase 서비스를 초기화합니다.
  - `Get.putAsync(() async => AuthService())`를 사용하여 `AuthService`를 비동기적으로 초기화하고 GetX 의존성 관리 시스템에 등록합니다.
  - 앱의 시작 화면은 `LoginPage`로 설정됩니다.

2. 홈페이지 구현

- **목표**: 로그인 후 메인 화면. 할 일 목록, 날짜 선택, 테마 변경, 로그아웃 기능 제공.
- **주요 파일**: `ui/pages/home_page.dart`, `controllers/task_controller.dart`, `services/auth_service.dart`, `services/theme_services.dart`.
- **구현 요약**:
  - **인증 확인**:
    - `initState()`: `AuthService.currentUser` 확인, 미로그인 시 `LoginPage`로 이동.
  - **UI 구성 (`ui/pages/home_page.dart`)**:
    - **앱바 (`_appBar`)**: 날짜 표시, 테마 변경 (`ThemeServices().switchTheme()`), 스크립트 페이지 이동, 로그아웃 (`authService.signOut()`).
    - **날짜 선택 (`_dateBar`)**: `date_picker_timeline` 사용. 날짜 선택 시 `_taskController.getTasksByDate(date)` 호출.
    - **할 일 목록 (`_showTasks`)**: `Obx`로 `_taskController.taskList` 실시간 업데이트. `FlutterStaggeredAnimations` 적용. `TaskTile` 탭 시 `TaskBottomSheet` (완료/삭제 옵션) 표시. 빈 목록 시 "No Tasks" 메시지.
    - **플로팅 액션 버튼**: "+ 할 일 추가" -> `AddSpeechToTextPage` 이동 (음성 입력 유도).
  - **데이터 연동**:
    - `TaskController` (`_taskController`): GetX로 주입.
    - `onReady()`: `_taskController.getTasks()` 호출 (초기 목록 로드).

3. To-do 추가 기능

- **목표**: 사용자가 새 할 일을 입력하고 데이터베이스에 저장.
- **주요 파일**: `ui/pages/add_task_page.dart`, `controllers/task_controller.dart`, `models/task.dart`, `db/db_helper.dart`.
- **구현 요약**:
  - **데이터 모델 (`models/task.dart`)**:
    - `Task` 클래스: `id`, `title`, `note`, `isCompleted`, `date`, `startTime`, `endTime`, `color`, `remind`, `repeat` 등 속성.
    - `toJson()`, `fromJson()`: Map 객체 변환 (DB 처리 용이).
  - **컨트롤러 (`controllers/task_controller.dart`)**:
    - `addTask({Task? task})`: `DBHelper.insert(task)`로 DB 저장 후 `taskList` 갱신.
  - **UI (`ui/pages/add_task_page.dart`)**:
    - `TaskController` 사용. `InputField`로 제목, 내용, 날짜, 시간, 반복, 색상 등 입력.
    - `initialTaskContent`: STT 결과로 내용 필드 초기화 가능.
    - 마이크 아이콘: `AddSpeechToTextPage` 이동 (음성 입력).
    - 날짜/시간 선택: `showDatePicker`, `showTimePicker` 사용.
    - "할 일 생성" 버튼: `_validateInputs()` (유효성 검사) -> `_addTaskToDb()` (`_taskController.addTask()`) -> `HomePage` 이동.
  - **데이터베이스 (`db/db_helper.dart`)**:

    - `DBHelper`: SQLite DB 생성, `tasks` 테이블 정의, CRUD 연산 담당.

4. To-do 완료 / 삭제 기능

- **목표**: 기존 할 일 완료 처리 또는 삭제.
- **주요 파일**: `controllers/task_controller.dart`, `ui/widgets/task_bottom_sheet.dart`, `ui/pages/home_page.dart`.
- **구현 요약**:
    - **컨트롤러 (`controllers/task_controller.dart`)**:
        - `markTaskAsCompleted(int id)`: `DBHelper.update(id)`로 `isCompleted` 상태 변경 후 `taskList` 갱신.
        - `deleteTasks(Task task)`: `DBHelper.delete(task)`로 DB 삭제 후 `taskList` 갱신.
    - **UI (`ui/widgets/task_bottom_sheet.dart`)**:
        - `TaskBottomSheet.showBottomSheet()`: `Get.bottomSheet()`로 바텀 시트 표시.
        - 포함 버튼: "할 일 완료" (미완료 시), "할 일 삭제", "취소".
        - 버튼 탭 시 해당 컨트롤러 메서드 호출. 동적 테마 스타일 적용.
    - **호출 (`ui/pages/home_page.dart`)**:
        - `HomePage`의 `TaskTile` 탭 시 `TaskBottomSheet.showBottomSheet()` 호출.

5. Speech-to-text (STT) 스크립트 추출

- **목표**: 사용자 음성을 텍스트로 변환하여 할 일 내용이나 스크립트로 활용.
- **주요 파일**: `ui/pages/add_speech_to_text.dart`, `speech_to_text` 패키지.
- **구현 요약**:
    - **STT 엔진 (`ui/pages/add_speech_to_text.dart`)**:
        - `stt.SpeechToText` 사용. `_initSpeech()`로 초기화.
        - `_startListening()`: 음성 인식 시작 (한국어 `ko_KR`), 결과는 `_titleController.text`에 반영.
        - `_stopListening()`: 음성 인식 중지.
    - **UI (`ui/pages/add_speech_to_text.dart`)**:
        - 마이크 버튼: 음성 인식 시작/중지 (로딩 인디케이터 표시).
        - 인식된 텍스트: `TextField` (`_titleController`)에 표시.
        - 스크립트 제목: `InputField` (`_scriptTitleController`)로 입력.
        - "스크립트 저장하기" 버튼: 유효성 검사 후 `Get.back(result: {'title': scriptTitle, 'content': scriptContent})`로 결과 전달.
    - **데이터 흐름**:
        - `AddSpeechToTextPage`는 텍스트를 직접 저장하지 않음.
        - 결과는 호출한 페이지로 전달되어 후속 처리 (예: 할 일 추가, 스크립트 저장).

6. 스크립트 저장 기능

- **목표**: STT 생성 또는 직접 입력한 텍스트와 제목을 "스크립트"로 저장 및 관리.
- **주요 파일**: `controllers/script_controller.dart`, `models/script_model.dart`, (예상) `ui/pages/script_page.dart`.
- **구현 요약**:
    - **데이터 모델 (`models/script_model.dart`)**:
        - `Script` 클래스: `id` (String, UUID), `title`, `content`, `createdAt` 속성.
        - `toJson()`, `fromJson()`: JSON 변환 (SharedPreferences 저장용).

    - **컨트롤러 (`controllers/script_controller.dart`)**:
        - `ScriptController` (GetX): 스크립트 데이터 관리.
        - `scripts = <Script>[].obs`: 반응형 스크립트 목록.
        - `addScript(String title, String content)`: 새 `Script` 생성 (UUID), `scripts` 리스트 추가, `SharedPreferences` 저장 (`_saveScriptsToPrefs()`).
        - `loadScripts()`: `SharedPreferences` ('user_scripts' 키)에서 스크립트 로드.
        - `deleteScript(String id)`, `updateScript(Script script)`: 삭제/업데이트 후 `SharedPreferences` 반영.
    - **UI (예상: `ui/pages/script_page.dart`)**:
        - `HomePage` 앱바 아이콘 통해 접근.
        - 스크립트 목록 표시, 새 스크립트 추가/조회/삭제 UI 제공.

7. 다크 모드 구현

- **목표**: 라이트/다크 모드 전환 및 사용자 선택 유지.
- **주요 파일**: `services/theme_services.dart`, `ui/theme.dart`, `main.dart`, `ui/pages/home_page.dart`.
- **구현 요약**:
    - **테마 정의 (`ui/theme.dart`)**:
        - `Themes` 클래스: `light` (`ThemeData`), `dark` (`ThemeData`) 정적 필드 (색상, 밝기 등 설정).
        - 색상 상수 및 텍스트 스타일 getter 제공. 텍스트 색상은 `Get.isDarkMode`로 동적 변경.
        - 폰트: `GoogleFonts.lato()`.
    - **테마 서비스 (`services/theme_services.dart`)**:
        - `ThemeServices`: 테마 상태 관리.
        - `GetStorage`: 테마 모드 (`isDarkMode` 키) 로컬 저장/로드.
        - `theme` getter: 저장된 설정 기반 `ThemeMode` 반환.
        - `switchTheme()`: `Get.changeThemeMode()`로 테마 전환 및 `GetStorage`에 저장.
    - **앱 적용 (`main.dart`)**:
        - `GetMaterialApp`: `theme` (`Themes.light`), `darkTheme` (`Themes.dark`), `themeMode` (`ThemeServices().theme`) 설정.
    - **UI 트리거 (`ui/pages/home_page.dart`)**:
        - `HomePage` 앱바 아이콘 버튼: `ThemeServices().switchTheme()` 호출.
        - 위젯 스타일: `context.theme` 또는 `Get.isDarkMode`로 동적 적용.
