mobile programming project

모바일프로그래밍 프로젝트

mobile programming project__

```
<p style="text-align:left;">Left</p>
<p style="text-align:center;">Center</p>
<p style="text-align:right;">Right</p>
-
-
-
```

학　　과 : 컴퓨터공학과

이　　름 : 오진영

mobile programming project

프로젝트 개요 ; Project outline ___

프로젝트 기능 ; Project function ___

앱 화면 구성 ; Screen Configuration___

앱 구조 및 구현 방식 ; Structure & Implementation Method___

Q&A ; Q&A ___

mobile programming project

Project outline ___

프로젝트 명

MyTimetable – 나만의 시간표 관리 앱

특징

직관적인 시간표 입력 기능

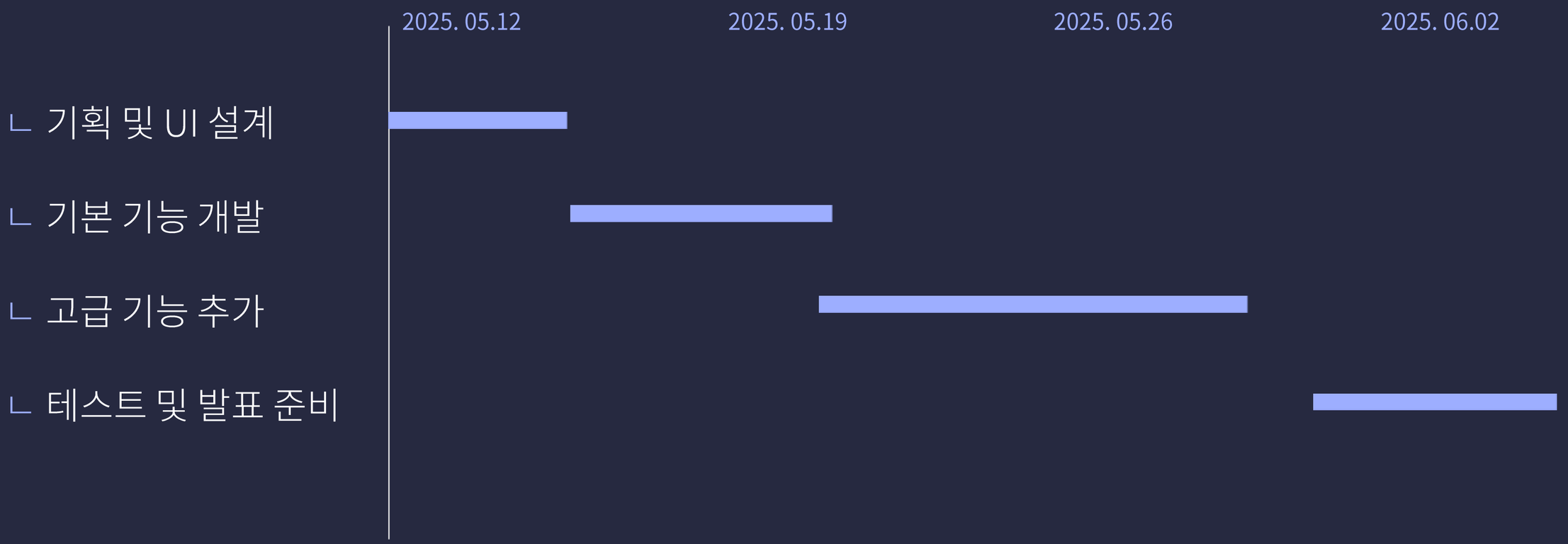
시간표 목록 조회 기능

시간표 삭제 가능

로컬 DB 저장

mobile programming project

Development Schedule



mobile programming project

project function ___

시간표 입력

요일, 시간, 과목명을 직접 입력 및 저장

화면에서 Spinner와 입력창을 통해
시간표를 추가

시간표 조회

RecyclerView로 정리된 시간표 목록 확인

메인화면에서 입력된 시간표들이
리스트로 정리되어 요일, 시간, 과목을
한눈에 볼 수 있게 구성

mobile programming project

project function ___

삭제 기능

리스트 항목을 길게 눌러 삭제

삭제 전에는 확인 다이얼로그가 떠서
실수 삭제 방지

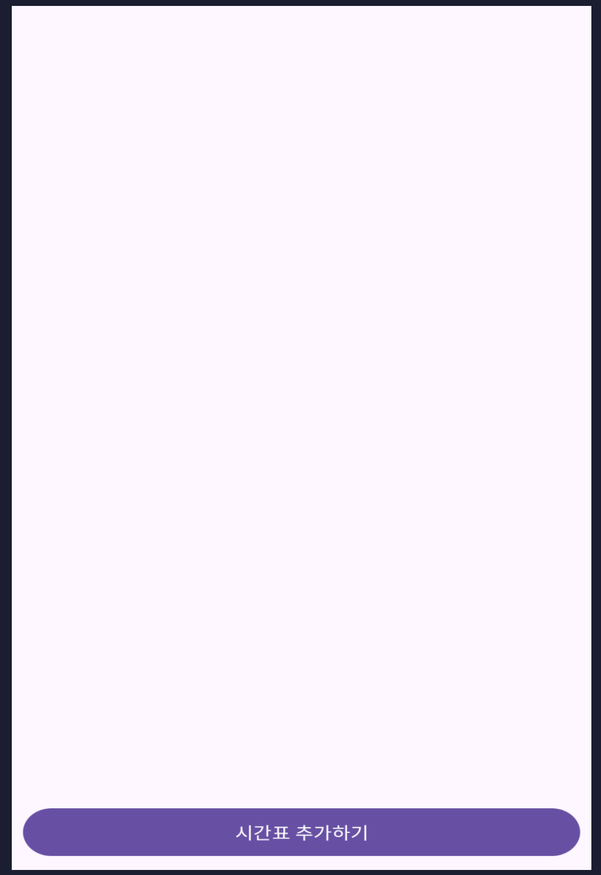
영구 저장

SQLite를 사용하여 앱 종료 후에도 유지

앱 종료해도 데이터 유지
별도의 로그인 없이도 개인 시간표 관리

mobile programming project

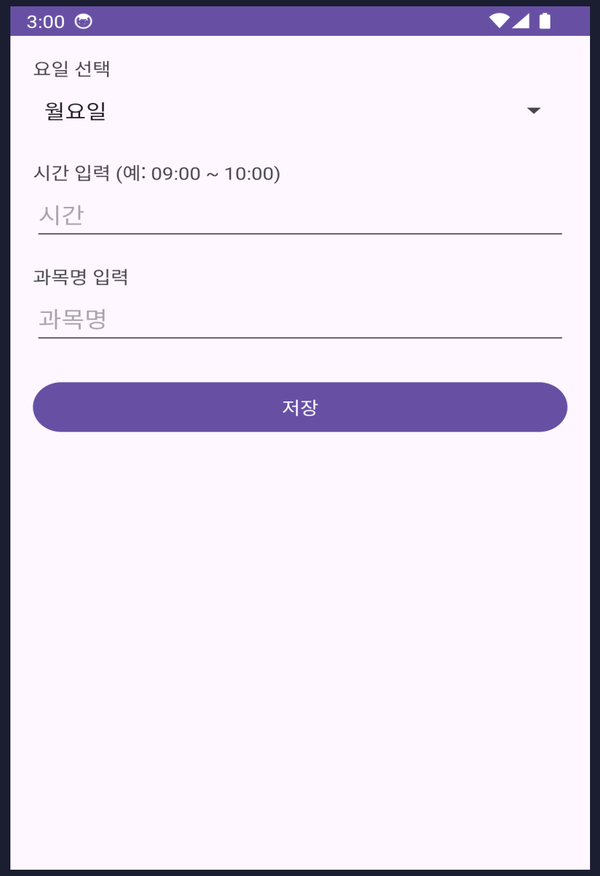
앱 메인 화면 – MainActivity : 시간표 목록 조회

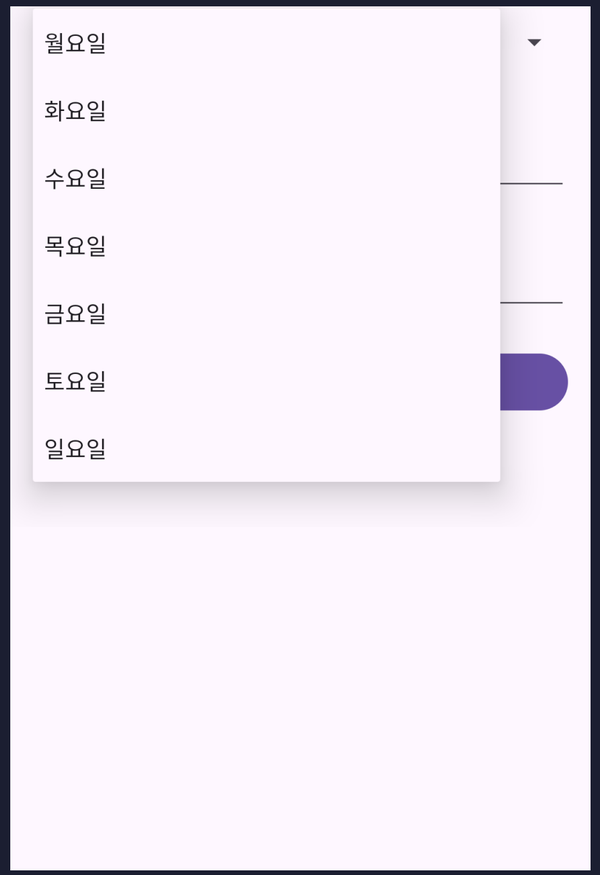


- 앱의 메인 화면
- 사용자가 입력한 시간표 목록이 요일, 시간 과목 순으로 출력
- RecyclerView를 이용해 리스트 형태로 정리
- 하단의 시간표 추가하기 버튼을 누르면 입력 화면으로 이동

mobile programming project

시간표 추가 화면 – AddScheduleActivity

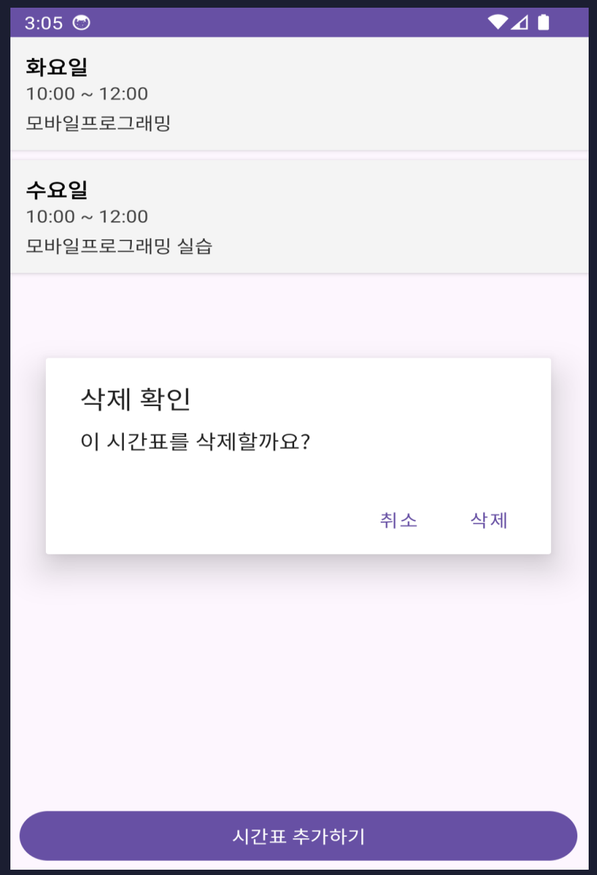


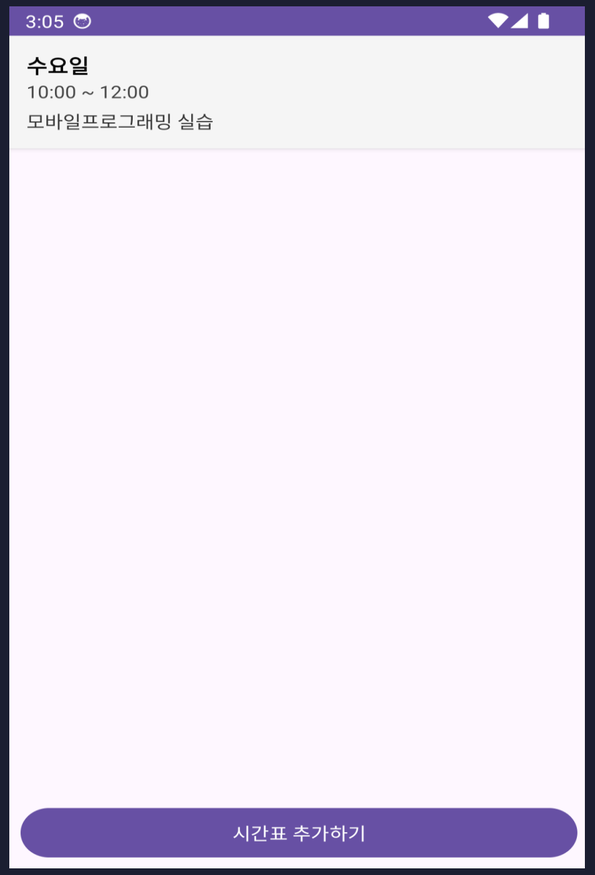


- 사용자가 새로운 시간표 항목 입력
- 요일은 Spinner로 선택
- 시간과 과목은 직접 입력
- 저장 버튼 누르면 메인화면으로 돌아감

mobile programming project

삭제 기능 화면 – 롱클릭 시 다이얼로그

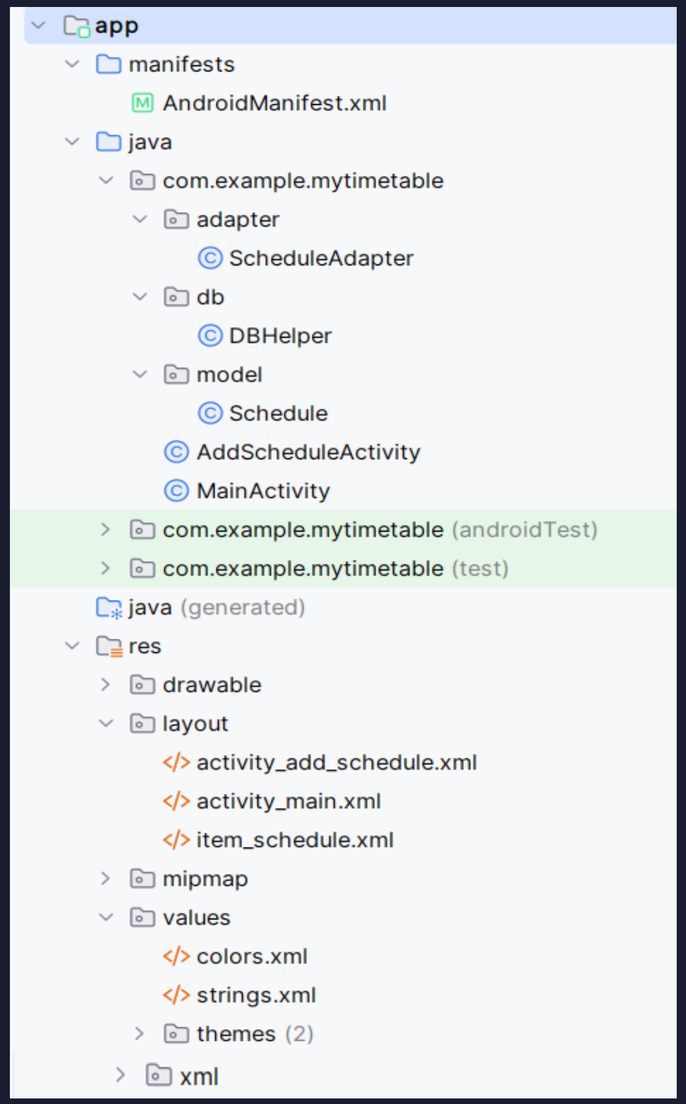




- 메인화면에서 시간표 항목 길게 누르면 삭제 여부를 묻는 다이얼로그 생성
- 삭제를 누르면 해당 항목이 삭제되고 리스트 자동 갱신

mobile programming project

앱 구조



1. AndroidManifest.xml
2. strings.xml
3. activity_add_schedule.xml
4. activity_main.xml
5. item_schedule.xml
6. Schedule.java (데이터 클래스)
7. DBHelper.java (DB 저장/삭제)
8. ScheduleAdapter.java (리스트 출력/삭제)
9. AddScheduleActivity.java (입력 로직)
10. MainActivity.java (전체 제어)

앱 전반구조 -> 데이터 흐름 -> 로직 구조

mobile programming project

AndroidManifest.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.mytimetable">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.MyTimetable">

        <activity
            android:name=".MainActivity"
            android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

        <!-- 시간표 추가 화면 -->
        <activity android:name=".AddScheduleActivity" />
    </application>

</manifest>
```

- <activity> 등록 (앱 화면 정의)
  - 앱 실행시 처음 열리는 MainActivity를 등록하는 코드
  - Intent-fliter를 통해 앱의 진입점이라는 것을 알림
  - Android 12 이상은 Android:exported=“true”를 명시해야 앱이 실행
- AddScheduleActivity 등록
  - 시간표를 입력하는 화면인 AddSchedule이 등록돼 있어야, MainActivity에서 화면 전환 가능

mobile programming project

strings.xml

```xml
<resources>
    <string name="app_name">MyTimetable</string>
    <string name="btn_save">저장</string>
    <string name="btn_add_schedule">시간표 추가하기</string>

    <string-array name="days_of_week">
        <item>월요일</item>
        <item>화요일</item>
        <item>수요일</item>
        <item>목요일</item>
        <item>금요일</item>
        <item>토요일</item>
        <item>일요일</item>
    </string-array>
</resources>
```

1. 앱에서 사용하는 모든 문자열을 한 곳에서 관리
   - 앱 전반에서 사용하는 텍스트를 모아놓은 리소스 파일
   - UI 요소의 텍스트를 직접 코드에 쓰는 대신, 여기서 참조하여 유지보수 용이하게 개발
2. 버튼 텍스트 재사용 가능하게 정의
   - 버튼에 표시되는 텍스트를 여기서 정의하여 화면마다 일관성을 유지
3. 요일 선택용 배열 정의
   - Spinner에 요일을 선택하게 만들기 위해 string-array를 정의하여 UI에서 바로 사용할 수 있게 개발

mobile programming project

activity_add_schedule.xml – (1)

```
<TextView android:text="요일 선택" android:layout_width="wrap_content" android:layout_height="wrap_content"/>
<Spinner
    android:id="@+id/spinner_day"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:entries="@array/days_of_week"
    android:minHeight="48dp"/>

<TextView android:text="시간 입력 (예: 09:00 ~ 10:00)" android:layout_width="wrap_content" android:layout_height
<EditText
    android:id="@+id/edit_time"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:hint="시간"
    android:minHeight="48dp"/>
```

시간표 항목을 입력하는 화면

1. 요일 선택 Spinner
   - 드롭다운 형태로 요일 선택
   - Strings.xml에 정의한 days_od_week 배열 사용
2. 시간 입력 EditText
   - 사용자가 원하는 시간대를 직접 입력
   - hint 속성으로 입력 예시 제공
   - 추후 TimePicker와 연동 가능

mobile programming project

activity_add_schedule.xml – (2)

```
<TextView android:text="과목명 입력" android:lay
<EditText
    android:id="@+id/edit_subject"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:hint="과목명"
    android:minHeight="48dp"/>

<Button
    android:id="@+id/btn_save"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="저장"
    android:layout_marginTop="24dp"/>
```

3. 과목 입력 EditText
   - 과목명을 직접 입력
   - 필수 입력 요소로 사용됨 (AddScheduleActivity.java에서 빈 값 체크 있음)
4. 저장 버튼
   - 사용자가 입력을 마친 후 데이터 저장
   - AddScheduleActivity.java에서 버튼 클릭시 SQLite에 저장되고 앱 메인화면으로 돌아감

mobile programming project

activity_main.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent" android:layout_height="match_parent">

    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/recycler_schedule"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_above="@+id/btn_add"/>

    <Button
        android:id="@+id/btn_add"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="시간표 추가하기"
        android:layout_alignParentBottom="true"
        android:layout_margin="8dp"/>
</RelativeLayout>
```

- 앱의 메인화면으로 사용자가 입력한 시간표 목록을 확인 가능
- RecyclerView를 통해 리스트 형태로 출력
- RelactiveLayout을 사용해 하단 버튼과 리스트가 충돌 없이 배치

1. RecyclerView 사용
   - 입력된 시간표를 리스트 형태로 출력
   - Adapter와 연결되어 실시간 데이터 반영
2. 추가 버튼(하단 고정)
   - 사용자가 새 시간표를 추가할 수 있도록 하단에 배치
   - 버튼 클릭시 AddScheduleActivity로 이동

mobile programming project

Item_schedule.xml

```xml
<TextView
    android:id="@+id/text_day"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="월요일"
    android:textSize="16sp"
    android:textStyle="bold"
    android:textColor="#000000" />

<TextView
    android:id="@+id/text_time"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="09:00 ~ 10:00"
    android:textSize="14sp"
    android:textColor="#444444" />

<TextView
    android:id="@+id/text_subject"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="자료구조"
    android:textSize="14sp"
    android:textColor="#333333"
    android:layout_marginTop="4dp" />
```

- 시간표 목록에 표시되는 각 한 줄(하나의 수업 정보)의 모양 정의

1. 요일 표시 TextView

첫번째 줄은 수업이 있는 요일을 굵은 글씨로 표시하여 어떤 요일인지 빠르게 구분할 수 있도록 구성

2. 시간 표시 TextView

두번째 줄에는 수업 시간을 출력하며, 가독성을 고려해 글자 크기 설정

3. 과목명 표시 TextView

마지막 줄에는 수업 과목명을 표시하며, 위쪽과의 간격을 두어 구분감

mobile programming project

Schedule.java – (1)

```java
package com.example.mytimetable.model;

8 usages
public class Schedule {
    2 usages
    private int id;
    2 usages
    private String day;
    2 usages
    private String time;
    2 usages
    private String subject;

    1 usage
    public Schedule(int id, String day, String time, String subject) {
        this.id = id;
        this.day = day;
        this.time = time;
        this.subject = subject;
    }

    1 usage
    public int getId() { return id; }
    1 usage
    public String getDay() { return day; }
    1 usage
    public String getTime() { return time; }
    1 usage
    public String getSubject() { return subject; }
}
```

- 앱에서 사용하는 시간표 데이터 구조(모델 클래스)
- 시간표의 각 항목을 하나의 객체로 표현
- 요일, 시간, 과목명, 고유한 ID 저장

1. 클래스가 정의된 위치
   - model 패키지에 위치
   - 앱에서 사용하는 데이터 구조만 따로 정리해 유지보수와 가독성 높임
2. 멤버 변수 정의
   - 시간표 항목 하나를 표현하기 위해 4개의 멤버 변수 사용
   - Id는 각 항목을 구분하는 고유 값
   - day, time, subject는 각각 요일, 시간, 과목 정보 저장

mobile programming project

Schedule.java – (2)

```java
package com.example.mytimetable.model;

8 usages
public class Schedule {
    2 usages
    private int id;
    2 usages
    private String day;
    2 usages
    private String time;
    2 usages
    private String subject;

    1 usage
    public Schedule(int id, String day, String time, String subject) {
        this.id = id;
        this.day = day;
        this.time = time;
        this.subject = subject;
    }

    1 usage
    public int getId() { return id; }
    1 usage
    public String getDay() { return day; }
    1 usage
    public String getTime() { return time; }
    1 usage
    public String getSubject() { return subject; }
}
```

3. 생성자 정의

생성자를 통해 객체를 만들 때 네 가지 값을 한번에 넣을 수 있도록 설계

4. Getter 메서드

외부 클래스에서는 이 객체의 값을 직접 접근하지 않고, getter 메서드를 통해 값을 읽을 수 있도록 캡슐화를 적용

mobile programming project

DBHelper.java – (1)

```
1  public class DBHelper extends SQLiteOpenHelper {

   1 usage
2  public static final String DB_NAME = "timetable.db";
   1 usage
   public static final int DB_VERSION = 1;

   2 usages
   public DBHelper(Context context) { super(context, DB_NAME, factory: null, DB_VERSION); }

3  @Override
   public void onCreate(SQLiteDatabase db) {
       db.execSQL(
               "CREATE TABLE schedule (" +
                       "id INTEGER PRIMARY KEY AUTOINCREMENT, " +
                       "day TEXT, " +
                       "time TEXT, " +
                       "subject TEXT)"
       );
   }

   1 usage
   public void deleteSchedule(int id) {
       SQLiteDatabase db = this.getWritableDatabase();
       db.delete( table: "schedule", whereClause: "id=?", new String[]{String.valueOf(id)});
       db.close();
   }
```

- SQLite 데이터베이스를 생성하고 관리하는 핵심 클래스
- 시간표 정보를 저장하는 schedule 테이블 생성
- 데이터를 삭제할 수 있는 기능 제공

1. SQLiteOpenHelper 상속
   - 안드로이드에서 제공하는 SQLiteOpenHelper 클래스를 상속받아 데이터베이스 생성과 버전 관리 담당
2. 데이터베이스 이름과 버전
   - 데이터베이스 파일의 이름은 timetable.db로 지정, 현재는 초기 버전인 1
3. 테이블 생성 – onCreate()
   - 앱이 처음 실행될 때 onCreate() 메서드가 호출되어 schedule 테이블 생성
   - 각 항목은 id, day, time, subject로 구성, id는 자동 증가하는 기본 키

mobile programming project

DBHelper.java – (2)

④
```
public void deleteSchedule(int id) {
    SQLiteDatabase db = this.getWritableDatabase();
    db.delete( table: "schedule",  whereClause: "id=?", new String[]{String.valueOf(id)});
    db.close();
}
```

no usages

⑤
```
@Override
public void onUpgrade(SQLiteDatabase db, int oldV, int newV) {
    db.execSQL("DROP TABLE IF EXISTS schedule");
    onCreate(db);
}
```

4. 삭제 기능 구현
   - 사용자가 항목을 삭제할 때 호출되는 메서드
   - 해당 id에 해당하는 데이터를 DB에서 삭제
5. 버전 변경시 onUpgrade() 처리
   - 데이터베이스 버전이 바뀔 경우 기존 테이블을 삭제하고 새로 생성해 앱이 최신 구조를 유지할 수 있도록 개발

mobile programming project

ScheduleAdapter.java – (1)

```
① public class ScheduleAdapter extends RecyclerView.Adapter<ScheduleAdapter.ViewHolder> {

    3 usages
    private List<Schedule> scheduleList;
    3 usages
    private OnItemDeleteListener onItemDeleteListener;

    1 usage
    public ScheduleAdapter(List<Schedule> list) { this.scheduleList = list; }

    4 usages
② public static class ViewHolder extends RecyclerView.ViewHolder {
        2 usages
        TextView day, time, subject;

        1 usage
        public ViewHolder(View view) {
            super(view);
            day = view.findViewById(R.id.text_day);
            time = view.findViewById(R.id.text_time);
            subject = view.findViewById(R.id.text_subject);
        }
    }
```

- 시간표 데이터를 리스트 형태로 출력해주는 RecyclerView의 어댑터 클래스
- 각 시간표 항목을 화면에 표시하고, 롱클릭시 삭제 다이얼로그를 띄워서 삭제 요청 처리

1. RecyclerView.Adapter 상속
   - RecyclerView.Adapter를 상속해서 리스트 화면에서 데이터를 어떻게 표시할지 정의
2. ViewHolder 패턴
   - ViewHolder는 시간표 한 줄에서 사용하는 UI 요소들을 저장하는 클래스
   - 매번 뷰를 새로 생성하지 않고 재사용함으로써 성능을 높임
3. onCreateViewHolder – 레이아웃 연결

- item_schedule.xml 레이아웃을 읽어와서 각 시간표 항목을 어떻게 보여줄지 정의

mobile programming project

ScheduleAdapter.java – (2)

```java
@Override
public ViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
    View view = LayoutInflater.from(parent.getContext())
            .inflate(R.layout.item_schedule, parent, attachToRoot: false);
    return new ViewHolder(view);
}

@Override
public void onBindViewHolder(ViewHolder holder, int position) {
    Schedule schedule = scheduleList.get(position);

    holder.day.setText(schedule.getDay());
    holder.time.setText(schedule.getTime());
    holder.subject.setText(schedule.getSubject());

    // ✅ 롱클릭 시 삭제 다이얼로그
    holder.itemView.setOnLongClickListener(v -> {
        AlertDialog.Builder builder = new AlertDialog.Builder(holder.itemView.getContext());
        builder.setTitle("삭제 확인")
                .setMessage("이 시간표를 삭제할까요?")
                .setPositiveButton( text: "삭제", (dialog, which) -> {
                    if (onItemDeleteListener != null) {
                        onItemDeleteListener.onDelete(schedule.getId());
                    }
                })
                .setNegativeButton( text: "취소", listener: null)
                .show();
        return true;
    });
}
```

3. onCreateViewHolder – 레이아웃 연결
   - item_schedule.xml 레이아웃을 읽어와서 각 시간표 항목을 어떻게 보여줄지 정의
4. onBindViewHolder – 데이터 바인딩
   - 데이터의 요일, 시간, 과목 정보를 각각의 TextView에 연결해서 화면에 표시
5. 롱클릭시 삭제 다이얼로그 구현
   - 사용자가 리스트 항목을 길게 누르면 삭제 여부를 묻는 다이얼로그가 표시되고, 확인을 누르면 외부에서 등록된 삭제 리스너를 통해 DB에서 데이터를 삭제

mobile programming project

ScheduleAdapter.java – (3)

```
    @Override
    public int getItemCount() { return scheduleList.size(); }

    // ✅ 인터페이스 정의
⑥ 2 usages
    public interface OnItemDeleteListener {
        1 usage
        void onDelete(int id);
    }

    // ✅ 리스너 등록 메서드
    1 usage
    public void setOnItemDeleteListener(OnItemDeleteListener listener) {
        this.onItemDeleteListener = listener;
    }
}
```

6. 삭제 리스너 인터페이스 정의
   - 삭제 처리는 Adapter 내부에서 직접하지 않고, 외부(MainActivity)에서 처리할 수 있도록 인터페이스를 정의해 구조를 분리

mobile programming project

AddScheduleActivity.java – (1)

```
spinnerDay = findViewById(R.id.spinner_day);
editTime = findViewById(R.id.edit_time);
editSubject = findViewById(R.id.edit_subject);
btnSave = findViewById(R.id.btn_save);

dbHelper = new DBHelper( context: this);

btnSave.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        String day = spinnerDay.getSelectedItem().toString();
        String time = editTime.getText().toString();
        String subject = editSubject.getText().toString();

        if (time.isEmpty() || subject.isEmpty()) {
            Toast.makeText( context: AddScheduleActivity.this, text: "모든 항목을 입력하세요", Toast.LENGTH_SHORT).show();
            return;
        }

        SQLiteDatabase db = dbHelper.getWritableDatabase();
        ContentValues values = new ContentValues();
        values.put("day", day);
        values.put("time", time);
        values.put("subject", subject);

        db.insert( table: "schedule",  nullColumnHack: null, values);
        db.close();

        Toast.makeText( context: AddScheduleActivity.this,  text: "저장되었습니다!", Toast.LENGTH_SHORT).show();
```

① ②

- 시간표 항목을 입력하는 화면
- 사용자가 요일, 시간, 과목명을 입력한 뒤 저장 버튼을 누르면, 입력한 정보가 SQLite에 저장되고 메인화면으로 돌아감

1. UI 요소 초기화
   - 레이아웃 XML 파일(activity_add_schedule.xml)에 있는 Spinner, EditText, Button을 Java 코드에서 연결해서 사용할 수 있도록 초기화
2. DBHelper 초기화
   - DB에 데이터를 저장하기 위해 DBHelper 클래스를 객체로 생성해 사용

mobile programming project

AddScheduleActivity.java – (2)

```
spinnerDay = findViewById(R.id.spinner_day);
editTime = findViewById(R.id.edit_time);
editSubject = findViewById(R.id.edit_subject);
btnSave = findViewById(R.id.btn_save);

dbHelper = new DBHelper( context: this);

btnSave.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        String day = spinnerDay.getSelectedItem().toString();
        String time = editTime.getText().toString();
        String subject = editSubject.getText().toString();

        if (time.isEmpty() || subject.isEmpty()) {
            Toast.makeText( context: AddScheduleActivity.this, text: "모든 항목을 입력하세요", Toast.LENGTH_SHORT).show();
            return;
        }

        SQLiteDatabase db = dbHelper.getWritableDatabase();
        ContentValues values = new ContentValues();
        values.put("day", day);
        values.put("time", time);
        values.put("subject", subject);

        db.insert( table: "schedule",  nullColumnHack: null, values);
        db.close();

        Toast.makeText( context: AddScheduleActivity.this, text: "저장되었습니다!", Toast.LENGTH_SHORT).show();
        // 메인화면으로 돌아가기
        finish();
```

3. 저장 버튼 클릭 이벤트 처리
   - 사용자가 저장 버튼을 클릭하면 동작할 코드를 정의
4. 입력값 검증 및 저장 처리
   - 입력값이 비어 있을 경우 경고 메시지를 띄우고 저장하지 않도록 예외 처리
5. SQLite에 데이터 저장
   - ContentValues에 입력된 데이터를 담아서 schedule 테이블에 insert() 메서드로 저장
6. 저장 완료 후 종료 및 Toast 출력
   - 저장이 완료되면 사용자에게 메시지를 보여주고, finish()를 통해 메인 화면으로 자동 복귀

mobile programming project

MainActivity.java – (1)

```
recyclerView.setLayoutManager(new LinearLayoutManager( context: this));
recyclerView.setAdapter(adapter);
```

```
private void loadSchedules() {
    scheduleList.clear();

    SQLiteDatabase db = dbHelper.getReadableDatabase();
    Cursor cursor = db.rawQuery( sql: "SELECT * FROM schedule",  selectionArgs: null);

    while (cursor.moveToNext()) {
        int id = cursor.getInt(cursor.getColumnIndexOrThrow( s: "id"));
        String day = cursor.getString(cursor.getColumnIndexOrThrow( s: "day"));
        String time = cursor.getString(cursor.getColumnIndexOrThrow( s: "time"));
        String subject = cursor.getString(cursor.getColumnIndexOrThrow( s: "subject"));

        Schedule schedule = new Schedule(id, day, time, subject);
        scheduleList.add(schedule);
    }
```

전체 흐름을 제어하는 중심 액티비티, 앱의 메인 화면

저장된 시간표를 RecyclerView로 출력

새로운 시간표를 추가하거나 삭제하는 기능 전체를 제어

1. RecyclerView 설정 + Adapter 연결
   - 시간표 항목을 리스트 형태로 보여주기 위해 RecyclerView를 사용
   - 데이터를 화면에 출력하기 위해 ScheduleAdapter를 연결

2. SQLite DB에서 데이터 불러오기
   - DB에서 시간표 데이터를 가져오는 loadSchedules() 메서드를 통해 앱 실행 시마다 최신 데이터를 불러와 리스트에 표시

mobile programming project

MainActivity.java – (2)

```
adapter.setOnItemDeleteListener(id -> {
    dbHelper.deleteSchedule(id); // DB에서 삭제
    loadSchedules();             // 목록 새로고침
});
```

```
btnAdd.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        // 시간표 추가 액티비티로 이동
        Intent intent = new Intent( packageContext: MainActivity.this, AddScheduleActivity.class);
        startActivity(intent);
    }
});
```

```
@Override
protected void onResume() {
    super.onResume();
    loadSchedules(); // 다시 화면에 들어오면 최신 데이터 로드
}
```

3. 데이터 삭제 기능 구현
   - ScheduleAdapter에서 롱클릭 이벤트가 발생하면 여기서 deleteScheduld()을 호출해 해당 데이터를 DB에서 삭제하고 리스트를 다시 새로고침

4. 추가 버튼 클릭 시 입력 화면으로 이동
   - 하단의 버튼을 누르면 AddScheduleActivity로 이동해 새로운 시간표를 입력 가능

5. onResume()에서 자동 새로고침 처리
   - 사용자가 입력화면에서 돌아왔을 때 자동으로 시간표 목록이 갱신되도록 onResume()에서 loadSchedules()를 호출

mobile programming project

감사합니다!