**Section 20: CardView - The Sports App**

**Key Concepts Taught**

1. **CardView**: Container widget with rounded corners, shadows, and elevation
2. **RecyclerView Integration**: Displaying CardViews in scrollable lists
3. **Custom Adapter/ViewHolder**: Binding data to CardView items
4. **Material Design Principles**: Using elevation, corner radius, and shadows
5. **Layout Optimization**: Efficient view inflation and recycling

**Implementation Steps with Detailed Comments**

**1. Project Setup**

gradle

*// Add dependencies in build.gradle (Module)*

implementation 'androidx.cardview:cardview:1.0.0'

implementation 'androidx.recyclerview:recyclerview:1.3.2'

* **Comment**: CardView and RecyclerView dependencies enable modern UI components

**2. Card Item Layout (**card\_item\_layout.xml**)**

xml

<androidx.cardview.widget.CardView

xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:app="http://schemas.android.com/apk/res-auto"

android:id="@+id/cardView"

android:layout\_width="match\_parent"

android:layout\_height="200dp"

android:layout\_margin="8dp"

app:cardCornerRadius="20dp" <!-- Rounded corners -->

app:cardElevation="8dp"> *<!-- Shadow depth -->*

<androidx.constraintlayout.widget.ConstraintLayout

android:layout\_width="match\_parent"

android:layout\_height="match\_parent">

*<!-- Sport image filling card -->*

<ImageView

android:id="@+id/imageView"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:scaleType="centerCrop"

app:layout\_constraintBottom\_toBottomOf="parent"

app:layout\_constraintEnd\_toEndOf="parent"

app:layout\_constraintStart\_toStartOf="parent"

app:layout\_constraintTop\_toTopOf="parent"/>

*<!-- Sport name centered in card -->*

<TextView

android:id="@+id/textView"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:textSize="24sp"

android:textStyle="bold"

android:textColor="@android:color/white"

app:layout\_constraintBottom\_toBottomOf="parent"

app:layout\_constraintEnd\_toEndOf="parent"

app:layout\_constraintStart\_toStartOf="parent"

app:layout\_constraintTop\_toTopOf="parent"/>

</androidx.constraintlayout.widget.ConstraintLayout>

</androidx.cardview.widget.CardView>

* **Best Practice**:
  + Use centerCrop scaleType for consistent image aspect ratios
  + Elevation values: 2-8dp for subtle shadows (Material Design guidelines)

**3. Model Class (**Sport.java**)**

java

public class Sport {

private String sportName; *// Sport title (e.g., "Football")*

private int sportImage; *// Drawable resource ID (R.drawable.football)*

*// Constructor initializes data fields*

public Sport(String sportName, int sportImage) {

this.sportName = sportName;

this.sportImage = sportImage;

}

*// Getters allow adapter to access private data*

public String getSportName() { return sportName; }

public int getSportImage() { return sportImage; }

}

* **Comment**: Model classes encapsulate data for cleaner architecture

**4. Adapter Class (**CustomAdapter.java**)**

java

public class CustomAdapter extends RecyclerView.Adapter<CustomAdapter.SportViewHolder> {

private List<Sport> sportList; *// Data source*

*// Constructor receives sports data*

public CustomAdapter(List<Sport> sportList) {

this.sportList = sportList;

}

*// ViewHolder: Caches view references*

public static class SportViewHolder extends RecyclerView.ViewHolder {

ImageView sportImageView;

TextView sportTextView;

CardView cardView;

public SportViewHolder(@NonNull View itemView) {

super(itemView);

*// Initialize views ONCE (efficient)*

sportImageView = itemView.findViewById(R.id.imageView);

sportTextView = itemView.findViewById(R.id.textView);

cardView = itemView.findViewById(R.id.cardView);

}

}

@NonNull

@Override

public SportViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {

*// Inflate card layout: Convert XML to View object*

View view = LayoutInflater.from(parent.getContext())

.inflate(R.layout.card\_item\_layout, parent, false); *// false = don't attach to parent yet*

return new SportViewHolder(view); *// Pass to ViewHolder*

}

@Override

public void onBindViewHolder(@NonNull SportViewHolder holder, int position) {

Sport currentSport = sportList.get(position); *// Get data for position*

*// Bind data to views*

holder.sportImageView.setImageResource(currentSport.getSportImage());

holder.sportTextView.setText(currentSport.getSportName());

}

@Override

public int getItemCount() {

return sportList.size(); *// Inform RecyclerView of item count*

}

}

**5. MainActivity Setup**

java

public class MainActivity extends AppCompatActivity {

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

*// 1. Initialize RecyclerView*

RecyclerView recyclerView = findViewById(R.id.recyclerView);

*// 2. Set LayoutManager (MANDATORY)*

recyclerView.setLayoutManager(new LinearLayoutManager(this)); *// Vertical list*

*// 3. Prepare data*

List<Sport> sportList = new ArrayList<>();

sportList.add(new Sport("Football", R.drawable.football));

sportList.add(new Sport("Basketball", R.drawable.basketball));

*// Add more sports...*

*// 4. Create adapter with data*

CustomAdapter adapter = new CustomAdapter(sportList);

*// 5. Connect adapter to RecyclerView*

recyclerView.setAdapter(adapter);

}

}

* **Critical Note**: Forgetting setLayoutManager() causes blank RecyclerView

**6. Handle CardView Clicks (Assignment Solution)**

java

*// Step 1: Add interface to adapter*

public interface OnItemClickListener {

void onItemClick(int position); }

*// Step 2: Modify ViewHolder*

public class SportViewHolder ... implements View.OnClickListener {

*// ... existing code*

private OnItemClickListener listener;

public SportViewHolder(...) {

*// ... existing code*

cardView.setOnClickListener(this); *// Set click listener*

}

public void setListener(OnItemClickListener listener) {

this.listener = listener;

}

@Override

public void onClick(View v) {

if (listener != null) {

listener.onItemClick(getAdapterPosition());

}

}

}

*// Step 3: Implement in Activity*

public class MainActivity implements CustomAdapter.OnItemClickListener {

@Override

protected void onCreate(Bundle savedInstanceState) {

*// ... existing code*

adapter.setListener(this); *// Pass click listener*

}

@Override

public void onItemClick(int position) {

Sport clickedSport = sportList.get(position);

Toast.makeText(this, "Selected: " + clickedSport.getSportName(),

Toast.LENGTH\_SHORT).show();

}

}

**Tools & APIs Used**

* **AndroidX Libraries**: cardview, recyclerview
* **Key Classes**: CardView, RecyclerView, LinearLayoutManager
* **Design Tools**: Android Studio Layout Editor

**Best Practices & Modern Approaches**

1. **CardView Customization**:
   * **Corner Radius**: Use 4-12dp for subtle rounding (Material Design)
   * **Elevation**: 1-6dp for normal cards, 8-24dp for raised interactions
   * **Ripple Effect**: Add android:foreground="?selectableItemBackground" for touch feedback
2. **Performance Optimization**:
   * Use setHasFixedSize(true) when card dimensions are uniform
   * Implement DiffUtil for efficient data updates:

java

DiffUtil.DiffResult result = DiffUtil.calculateDiff(new SportDiffCallback(oldList, newList));

result.dispatchUpdatesTo(adapter);

1. **Image Loading**:
   * Replace setImageResource() with Glide/Picasso for better memory management:

java

Glide.with(context).load(resourceId).into(holder.sportImageView);

1. **Material 3 Cards**:
   * Use com.google.android.material.card.MaterialCardView for latest features:

xml

<com.google.android.material.card.MaterialCardView

app:strokeColor="@color/..."

app:strokeWidth="2dp">

**Part B: Important Topics Not Covered**

1. **CardView States**:
   * Checkable cards: app:checkable="true" + setChecked()
   * Draggable cards with ItemTouchHelper
2. **Hierarchical Cards**:
   * Card headers with app:headerLayout
   * Action buttons with app:actionLayout
3. **Advanced Styling**:
   * Custom shape theming with ShapeAppearanceOverlay
   * Dynamic elevation on press:

xml

<selector xmlns:android="...">

<item android:state\_pressed="true" android:eleveation="8dp"/>

<item android:elevation="4dp"/>

</selector>

1. **CardStack Layouts**:
   * Vertical stacking with com.yuyakaido.android:card-stack-view
   * Swipe gestures for Tinder-like interfaces
2. **Shared Element Transitions**:
   * Animate card expansion to detail view:

java

ActivityOptions options = ActivityOptions.makeSceneTransitionAnimation(

activity, cardView, "cardTransition");

1. **ConstraintLayout in Cards**:
   * Complex responsive layouts within cards:

xml

<androidx.constraintlayout.widget.ConstraintLayout

tools:ignore="MissingConstraints">

*<!-- Constrained elements -->*

</androidx.constraintlayout.widget.ConstraintLayout>

1. **Dark Mode Support**:
   * Theme-based card colors:

xml

app:cardBackgroundColor="?attr/colorSurface"

**Summary**

This section covered creating sports cards using CardView within RecyclerView. Key takeaways:

1. CardView provides Material Design containers with shadows and rounded corners
2. Essential attributes: cardCornerRadius, cardElevation
3. Adapter/ViewHolder pattern efficiently binds data to cards
4. Click handling requires custom interface implementation

For production apps:

* Use **MaterialCardView** for latest features
* Implement **image loading libraries** for memory efficiency
* Add **motion animations** for polished transitions
* Support **dark mode** with theme-aware colors
* Consider **card states** for interactive experiences