Plagiarism Scan Report

Summary	
Report Genrated Date	24 Apr, 2018
Plagiarism Status	**100% Unique**
Total Words	915
Total Characters	8550
Any Ignore Url Used	

Content Checked For Plagiarism:

{
"Friend_Requests" : {
"sender_id" : {
"receiver_id" : {
"request_type" : "sent"
}
},
},
"Friends" : {
"user_id" : {
"friend_id" : {
"date" : "22-March-2018"
},
},
"Messages" : {
"user_id" : {
"receiver_id" : {
"-L8InfN6tb4hM_JZeFYa" : {
"message" : "jgnnqakacoaucplc0 ",
"seen" : false,
"time" : 1521824479260,
"type" : "text"
},
"-LAnjEsNNf5DjdzaaSWn" : {
"message" : "jk\n",
"seen" : false,
"time" : 1524507673810,
"type" : "text"
}
},
"Users" : {
"user_id" : {
"online" : true,
"user_image" : "";
"user_name" : "Arbind Saraf",

```
"user_status" : "Hey, I am using powerful Message App",
"user_thumb_image" :
}
}
```

3.4. User Interface Model

ConstraintLayout permits you to generate large and complex layouts with a flat view order (no nested view groups). It's alike to RelativeLayout in that all opinions are laid out according to relations between sibling views in addition the parent layout, but it's more elastic than RelativeLayout and cooler to use with Android Studio's Layout Editor.

All the control of ConstraintLayout is available straight from the Layout Editor's graphic tools, as the layout API and the Layout Editor were specifically built for each other. So you can figure your layout with ConstraintLayout totally by drag-and-dropping in its place of editing the XML.

ConstraintLayout is accessible in an API public library that's well-suited with Android 2.3 (API level 9) and greater. This sheet delivers a monitor to construction a layout with ConstraintLayout in Android Studio 3.0 or higher. If you'd like more info about the Layout Editor himself, see the Android Studio monitor to form a UI with Layout Editor.

To describe a view's location in ConstraintLayout, you necessity add at minimum one horizontal and one vertical constraint on behalf of the view. Each constraint represents a joining or alignment to additional view, the parent layout, or an imperceptible guideline. Each constraint describes the view's position along also the vertical or horizontal axis; so each view must have a minimum of one constraint for each axis, but often more are necessary.

When you drip a view into the Layout Editor, it stays somewhere you leave it even if it has nope constraints. Though, this is simply to create editing cooler; if a view has no restrictions when you track your layout on a device.

In figure 3, the layout looks decent in the editor, however there's not any vertical restriction on view C. When this layout draws on a android device, view C flat aligns with the left and right ends of view A, but seems at the top of the screen since it has no vertical constraint.

Figure 3. The editor shows view C below A, but it has no vertical constraint

4. Programing concept
4.1. List of APIs used in App

• Firebase APIs --
```
implementation 'com.google.firebase:firebase-auth:11.0.4'
implementation 'com.google.firebase:firebase-database:11.0.4'
implementation 'com.google.firebase:firebase-storage:11.0.4'
implementation 'com.firebaseui:firebase-ui-database:2.3.0'
```

• Cirricular Image View --'de.hdodenhof:circleimageview:2.1.0'
• Cropping image --'com.theartofdev.edmodo:android-image-cropper:2.4.+'
• Picasso image utilities --'com.squareup.picasso:picasso:2.5.2'
• Image Compressor --'id.zelory:compressor:2.1.0'
• Offline Features 'com.squareup.okhttp:okhttp:2.5.0'

4.2. Important Code Snippets:

Firebase
Initialize firebase variables
```
private DatabaseReference mFirebaseDatabaseReference;
private FirebaseRecyclerAdapter
mFirebaseAdapter;
```

reading from database
```
mFirebaseDatabaseReference = FirebaseDatabase.getInstance().getReference();
SnapshotParser parser = new SnapshotParser() {
@Override
public FriendlyMessage parseSnapshot(DataSnapshot dataSnapshot) {
FriendlyMessage friendlyMessage = dataSnapshot.getValue(FriendlyMessage.class);
if (friendlyMessage != null) {
friendlyMessage.setId(dataSnapshot.getKey());
}
return friendlyMessage;
}
};

DatabaseReference messagesRef = mFirebaseDatabaseReference.child(MESSAGES_CHILD);
FirebaseRecyclerOptions options =
new FirebaseRecyclerOptions.Builder()
.setQuery(messagesRef, parser)
.build();
mFirebaseAdapter = new FirebaseRecyclerAdapter(options) {
@Override
public MessageViewHolder onCreateViewHolder(ViewGroup viewGroup, int i) {
LayoutInflater inflater = LayoutInflater.from(viewGroup.getContext());
return new MessageViewHolder(inflater.inflate(R.layout.item_message, viewGroup, false));
}

@Override
protected void onBindViewHolder(final MessageViewHolder viewHolder,
int position,
FriendlyMessage friendlyMessage) {
mProgressBar.setVisibility(ProgressBar.INVISIBLE);
if (friendlyMessage.getText() != null) {
viewHolder.messageTextView.setText(friendlyMessage.getText());
viewHolder.messageTextView.setVisibility(TextView.VISIBLE);
viewHolder.messageImageView.setVisibility(ImageView.GONE);
} else {
String imageUrl = friendlyMessage.getImageUrl();
if (imageUrl.startsWith("gs://")) {
StorageReference storageReference = FirebaseStorage.getInstance()
.getReferenceFromUrl(imageUrl);
storageReference.getDownloadUrl().addOnCompleteListener(
new OnCompleteListener() {
@Override
public void onComplete(@NonNull Task task) {
if (task.isSuccessful()) {
String downloadUrl = task.getResult().toString();
Glide.with(viewHolder.messageImageView.getContext())
.load(downloadUrl)
.into(viewHolder.messageImageView);
} else {
Log.w(TAG, "Getting download url was not successful.",
task.getException());
}
}
});
} else {
Glide.with(viewHolder.messageImageView.getContext())
.load(friendlyMessage.getImageUrl())
.into(viewHolder.messageImageView);
}
viewHolder.messageImageView.setVisibility(ImageView.VISIBLE);
viewHolder.messageTextView.setVisibility(TextView.GONE);
}


viewHolder.messengerTextView.setText(friendlyMessage.getName());
if (friendlyMessage.getPhotoUrl() == null) {
viewHolder.messengerImageView.setImageDrawable(ContextCompat.getDrawable(MainActi
vity.this,
R.drawable.ic_account_circle_black_36dp));
} else {
Glide.with(MainActivity.this)
.load(friendlyMessage.getPhotoUrl())
.into(viewHolder.messengerImageView);
}

}
};
```

```
mFirebaseAdapter.registerAdapterDataObserver(new RecyclerView.AdapterDataObserver()
{
@Override
public void onItemRangeInserted(int positi mFirebaseAdapter.getItemCount();
int lastVisiblePosition =
mLinearLayoutManager.findLastCompletelyVisibleItemPosition();
// If the recycler view is initially being loaded or the
// user is at the bottom of the list, scroll to the bottom
// of the list to show the newly added message.
if (lastVisiblePosition == -1 ||
(positi (friendlyMessageCount - 1) &&
lastVisiblePosition == (positi" + requestCode + ", resultCode=" + resultCode);

if (requestCode == REQUEST_IMAGE) {
if (resultCode == RESULT_OK) {
if (data != null) {
final Uri uri = data.getData();
Log.d(TAG, "Uri: " + uri.toString());

FriendlyMessage tempMessage = new FriendlyMessage(null, mUsername, mPhotoUrl,
LOADING_IMAGE_URL);
mFirebaseDatabaseReference.child(MESSAGES_CHILD).push()
.setValue(tempMessage, new DatabaseReference.CompletionListener() {
@Override
public void onComplete(DatabaseError databaseError,
DatabaseReference databaseReference) {
if (databaseError == null) {
String key = databaseReference.getKey();
StorageReference storageReference =
FirebaseStorage.getInstance()
.getReference(mFirebaseUser.getUid())
.child(key)
.child(uri.getLastPathSegment());

putImageInStorage(storageReference, uri, key);
} else {
Log.w(TAG, "Unable to write message to database.",
databaseError.toException());
}
}
});
}
}
}
}
```

Cropping image

```
public void onActivityResult(int requestCode, int resultCode, Intent data) {
```

```
if (requestCode == CropImage.CROP_IMAGE_ACTIVITY_REQUEST_CODE) {
CropImage.ActivityResult result = CropImage.getActivityResult(data);
if (resultCode == RESULT_OK) {
Uri resultUri = result.getUri();
} else if (resultCode == CropImage.CROP_IMAGE_ACTIVITY_RESULT_ERROR_CODE) {
Exception error = result.getError();
}
}
}
```

Compressing image

Compressor is a lightweight and dominant android photo compression package. Compressing tools or API helps in compressing big-sized photos into smaller sized photos with very less or small damage in the meaning of the photo.

```
compressedImage = new Compressor(this)
.setMaxWidth(640)
.setMaxHeight(480)
.setQuality(75)
.setCompressFormat(Bitmap.CompressFormat.WEBP)
.setDestinationDirectoryPath(Environment.getExternalStoragePublicDirectory(
Environment.DIRECTORY_PICTURES).getAbsolutePath())
.compressToFile(actualImage);
```