COMP 1786 Logbook

Basic Information

1.1 Student name	**Nguyen Duc Hieu**
1.2 Who did you work with? Note that for logbook exercises you are allowed to work with one other person as long as you give their name and login id and both contribute to the work.	**Name:** **Login id:**
1.3 Which Exercise is this? Tick as appropriate.	• Exercise 1 ☐ • Exercise 2 ☐ • Exercise 3 ☒
1.4 How well did you complete the exercise? Tick as appropriate.	• I tried but couldn't complete it ☐ • I did it but I feel I should have done better ☐ • I did everything that was asked ☐ • I did more than was asked for ☒
1.5 Briefly explain your answer to question 1.4. Without any explanation/justification, your scores will be deducted.	**Exercise 1:** **A, Frontend** 1. Bottom Navigation with 3 tabs (Contacts, Favorites, Settings) 2. Color system with 60+ definitions 3. Real-time search by name/email 4. 4 sorting modes (A-Z, Z-A, Newest, Oldest) 5. Complete Favorites system 6. Image upload from device (external resources) 7. Appealing empty states 8. 27+ custom icons 9. Material Design 3 components 10. Gradient backgrounds **B, Backend / Architecture** 1. Extended Database schema (isFavorite, createdAt, updatedAt) 2. 15 complex DAO queries (COLLATE NOCASE, substring, partial update) 3. Broadcast Receiver for real-time sync 4. Async operations with ExecutorService 5. Internal storage management 6. Permission handling based on API level 7. Comprehensive data validation

	8. SecurityException handling

2. Exercise answer

2.1 Screen shots demonstrating what you achieved.

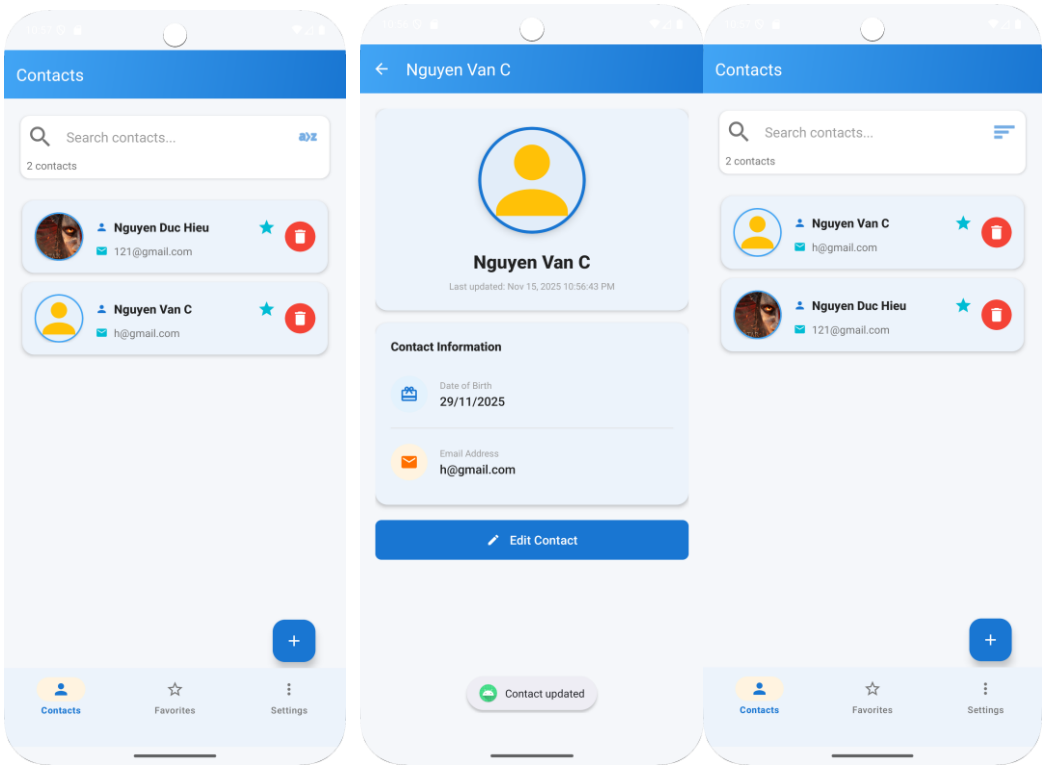

Figure 1 - Contact list, Update Contact, Sort Contact A-Z, Z-A

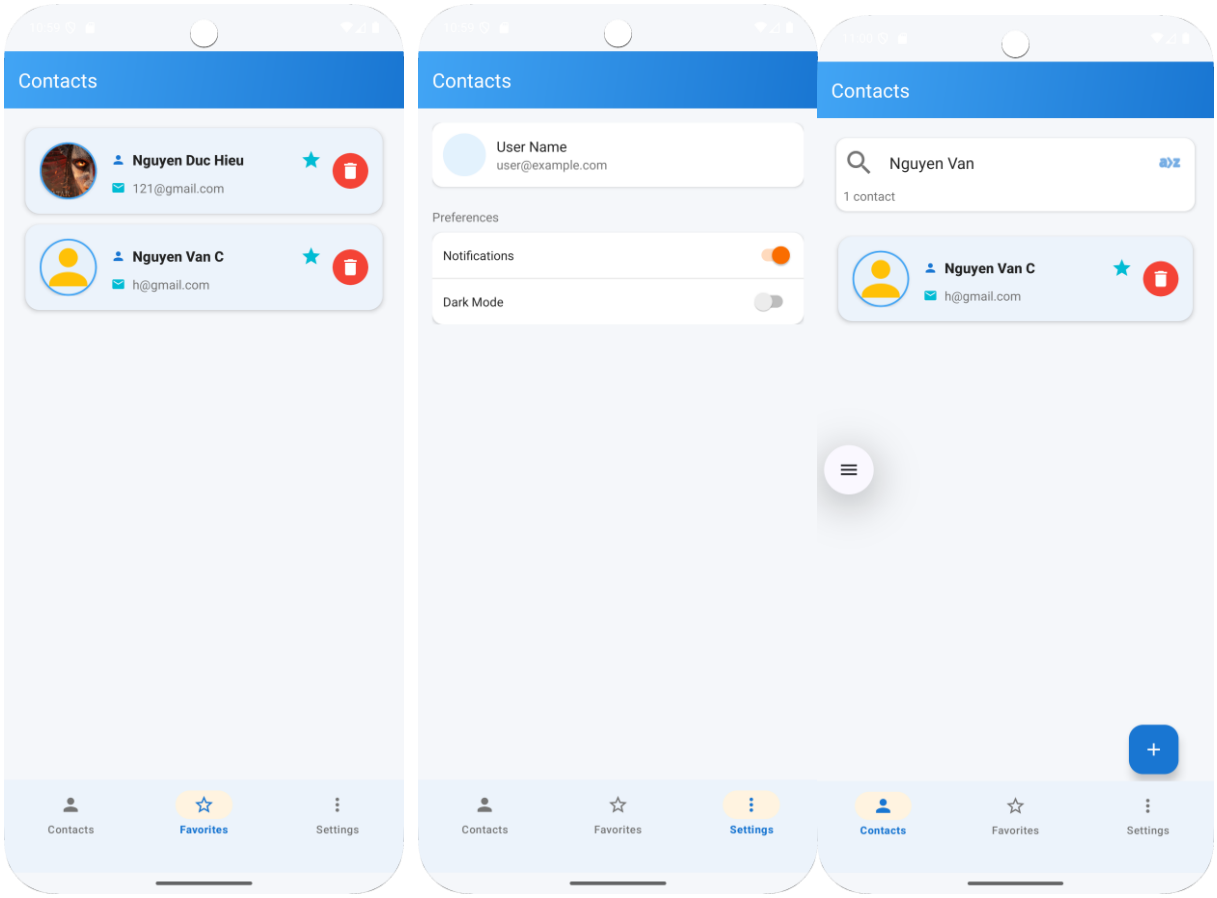

Figure 2 - Favourites, Setting, Search Contact

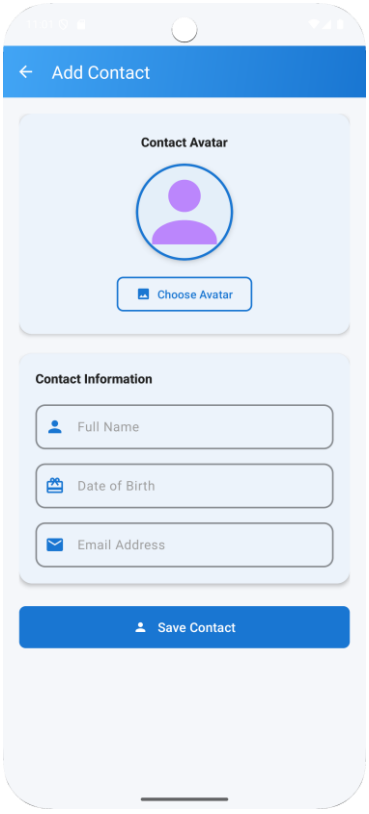

Figure 3 - Add new contact

2.2 Code that you wrote

AvatarGridAdapter.java

```java
package com.example.comp1786_logbook_ex3_connectdatabaseapplication.activity;

import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ImageView;

import androidx.annotation.NonNull;
import androidx.recyclerview.widget.RecyclerView;

import com.example.comp1786_logbook_ex3_connectdatabaseapplication.R;
import com.example.comp1786_logbook_ex3_connectdatabaseapplication.helper.ResUtils;

import java.util.List;

public class AvatarGridAdapter extends RecyclerView.Adapter<AvatarGridAdapter.VH> {

    private final List<String> names;
    private final OnPick onPick;
    private String selectedName;

    public interface OnPick { void onPick(String name); }

    public AvatarGridAdapter(List<String> names, OnPick onPick) {
        this.names = names;
        this.onPick = onPick;
        this.selectedName = null;
    }

    public void setSelectedName(String selectedName) {
        this.selectedName = selectedName;
        notifyDataSetChanged();
    }

    @NonNull
    @Override
    public VH onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
        View v = LayoutInflater.from(parent.getContext()).inflate(R.layout.item_avatar, parent, false);
        return new VH(v, onPick);
    }

    @Override
    public void onBindViewHolder(@NonNull VH holder, int position) {
        String name;
        if (position < names.size()) {
            name = names.get(position);
        } else {
            name = "upload";
        }
        boolean isSelected = selectedName != null && selectedName.equals(name);
        holder.bind(name, isSelected, () -> {
            this.selectedName = name;
            notifyDataSetChanged();
        });
    }

    @Override
    public int getItemCount() { return names.size() + 1; } // Add 1 for the upload button

    public static class VH extends RecyclerView.ViewHolder {
        ImageView img; OnPick onPick;
        View root;
        public VH(View v, OnPick onPick) {
            super(v);
            img = v.findViewById(R.id.imgAvatar);
            this.onPick = onPick;
            this.root = v;
        }
        public void bind(String name, boolean isSelected, Runnable onSelectChanged) {
            if (name.equals("upload")) {
                img.setImageResource(R.drawable.ic_add_a_photo); // A new drawable for upload
                root.setSelected(false);
                root.setOnClickListener(v -> onPick.onPick("upload"));
            } else {
```

```
            int res = ResUtils.nameToDrawable(img.getContext(), name);
            if (res != 0) img.setImageResource(res);
            root.setSelected(isSelected);
            root.setOnClickListener(v -> {
                onSelectChanged.run();
                onPick.onPick(name);
            });
        }
    }
}
```

AvatarPickerActivity

```
package com.example.comp1786_logbook_ex3_connectdatabaseapplication.activity;

import android.content.Intent;
import android.content.pm.PackageManager;
import android.net.Uri;
import android.os.Build;
import android.os.Bundle;
import android.widget.Toast;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.app.ActivityCompat;
import androidx.core.content.ContextCompat;
import androidx.recyclerview.widget.GridLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import com.example.comp1786_logbook_ex3_connectdatabaseapplication.R;
import com.example.comp1786_logbook_ex3_connectdatabaseapplication.helper.ResUtils;
import com.google.android.material.appbar.MaterialToolbar;

import java.io.File;
import java.io.FileOutputStream;
import java.io.InputStream;
import java.util.List;

public class AvatarPickerActivity extends AppCompatActivity {

    public static final String EXTRA_CHOSEN = "chosen";
    public static final String EXTRA_CURRENT_AVATAR = "current_avatar";
    private static final int REQ_PICK_IMAGE = 2002;
    private static final int REQ_PERMISSION = 2003;

    public static void startForResult(AppCompatActivity from, int req) {
        startForResult(from, req, null);
    }

    public static void startForResult(AppCompatActivity from, int req, @Nullable String currentAvatar) {
        Intent intent = new Intent(from, AvatarPickerActivity.class);
        if (currentAvatar != null) {
            intent.putExtra(EXTRA_CURRENT_AVATAR, currentAvatar);
        }
        from.startActivityForResult(intent, req);
    }

    @Override
    protected void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_avatar_picker);

        MaterialToolbar toolbar = findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);
        toolbar.setNavigationOnClickListener(v -> finish());

        RecyclerView rv = findViewById(R.id.rvAvatars);
        rv.setLayoutManager(new GridLayoutManager(this, 4));
        List<String> names = ResUtils.avatarNamesFromArray(this, R.array.contact_avatars);
        AvatarGridAdapter adapter = new AvatarGridAdapter(names, this::onPicked);
```

```
        // Highlight current avatar if provided
        String current = getIntent().getStringExtra(EXTRA_CURRENT_AVATAR);
        if (current != null) {
            adapter.setSelectedName(current);
        }

        rv.setAdapter(adapter);
    }

    private void onPicked(String name) {
        if (name.equals("upload")) {
            // Check if we need to request permission (Android 13+)
            if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.TIRAMISU) {
                if (ContextCompat.checkSelfPermission(this, android.Manifest.permission.READ_MEDIA_IMAGES)
                        != PackageManager.PERMISSION_GRANTED) {
                    // Request permission
                    ActivityCompat.requestPermissions(this,
                            new String[]{android.Manifest.permission.READ_MEDIA_IMAGES},
                            REQ_PERMISSION);
                    return;
                }
            } else if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
                if (ContextCompat.checkSelfPermission(this, android.Manifest.permission.READ_EXTERNAL_STORAGE)
                        != PackageManager.PERMISSION_GRANTED) {
                    // Request permission
                    ActivityCompat.requestPermissions(this,
                            new String[]{android.Manifest.permission.READ_EXTERNAL_STORAGE},
                            REQ_PERMISSION);
                    return;
                }
            }

            // Permission granted or not needed, launch picker
            launchImagePicker();
        } else {
            Intent data = new Intent();
            data.putExtra(EXTRA_CHOSEN, name);
            setResult(RESULT_OK, data);
            finish();
        }
    }

    private void launchImagePicker() {
        // Use ACTION_OPEN_DOCUMENT for better permission handling
        // This grants persistent read access to the URI
        Intent intent = new Intent(Intent.ACTION_OPEN_DOCUMENT);
        intent.setType("image/*");
        intent.addCategory(Intent.CATEGORY_OPENABLE);
        // This flag allows us to get persistent read permission
        intent.addFlags(Intent.FLAG_GRANT_READ_URI_PERMISSION | Intent.FLAG_GRANT_PERSISTABLE_URI_PERMISSION);

        android.util.Log.d("AvatarPicker", "Launching image picker with ACTION_OPEN_DOCUMENT");
        startActivityForResult(intent, REQ_PICK_IMAGE);
    }

    @Override
    public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions, @NonNull int[] grantResults) {
        super.onRequestPermissionsResult(requestCode, permissions, grantResults);
        if (requestCode == REQ_PERMISSION) {
            if (grantResults.length > 0 && grantResults[0] == PackageManager.PERMISSION_GRANTED) {
                // Permission granted, launch image picker
                launchImagePicker();
            } else {
                // Permission denied
                Toast.makeText(this, "Permission denied. Cannot access images.", Toast.LENGTH_SHORT).show();
            }
        }
    }

    @Override
    protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {
        super.onActivityResult(requestCode, resultCode, data);
        if (requestCode == REQ_PICK_IMAGE && resultCode == RESULT_OK && data != null) {
            Uri uri = data.getData();
```

```
        if (uri != null) {
            android.util.Log.d("AvatarPicker", "Received URI: " + uri.toString());

            // Copy the image to internal storage to ensure persistent access
            // This is necessary because Photo Picker URIs are temporary
            try {
                String savedUri = copyImageToInternalStorage(uri);
                if (savedUri != null) {
                    android.util.Log.d("AvatarPicker", "Saved URI: " + savedUri);
                    Intent resultIntent = new Intent();
                    resultIntent.putExtra(EXTRA_CHOSEN, savedUri);
                    setResult(RESULT_OK, resultIntent);
                    finish();
                } else {
                    Toast.makeText(this, "Failed to save image", Toast.LENGTH_SHORT).show();
                }
            } catch (SecurityException e) {
                android.util.Log.e("AvatarPicker", "SecurityException: " + e.getMessage(), e);
                Toast.makeText(this, "Permission error: " + e.getMessage(), Toast.LENGTH_LONG).show();
            } catch (Exception e) {
                android.util.Log.e("AvatarPicker", "Error saving image: " + e.getMessage(), e);
                Toast.makeText(this, "Error: " + e.getMessage(), Toast.LENGTH_SHORT).show();
            }
        }
    }

    /**
     * Copy the selected image to app's internal storage for persistent access
     * @param sourceUri The URI from the image picker
     * @return The file:// URI string of the saved image, or null if failed
     */
    private String copyImageToInternalStorage(Uri sourceUri) {
        android.util.Log.d("AvatarPicker", "Starting image copy from: " + sourceUri);
        InputStream inputStream = null;
        FileOutputStream outputStream = null;

        try {
            // Create avatars directory in internal storage
            File avatarsDir = new File(getFilesDir(), "avatars");
            if (!avatarsDir.exists()) {
                boolean created = avatarsDir.mkdirs();
                android.util.Log.d("AvatarPicker", "Created avatars dir: " + created + " at " + avatarsDir.getAbsolutePath());
            }

            // Generate unique filename
            String fileName = "avatar_" + System.currentTimeMillis() + ".jpg";
            File destFile = new File(avatarsDir, fileName);
            android.util.Log.d("AvatarPicker", "Destination file: " + destFile.getAbsolutePath());

            // Copy the image - THIS is where SecurityException might occur
            try {
                inputStream = getContentResolver().openInputStream(sourceUri);
            } catch (SecurityException e) {
                android.util.Log.e("AvatarPicker", "SecurityException opening input stream! This means we don't have permission to read the URI.",
e);
                // Re-throw with more context
                throw new SecurityException("Cannot access selected image. The Photo Picker URI may have expired or permission was denied.", e);
            }

            if (inputStream == null) {
                android.util.Log.e("AvatarPicker", "Failed to open input stream for URI: " + sourceUri);
                return null;
            }

            android.util.Log.d("AvatarPicker", "Input stream opened successfully");

            outputStream = new FileOutputStream(destFile);
            byte[] buffer = new byte[4096];
            int bytesRead;
            long totalBytes = 0;

            while ((bytesRead = inputStream.read(buffer)) != -1) {
                outputStream.write(buffer, 0, bytesRead);
```

```
            totalBytes += bytesRead;
        }

        outputStream.flush();
        outputStream.close();
        inputStream.close();

        android.util.Log.d("AvatarPicker", "Image copied successfully. Total bytes: " + totalBytes);

        // Verify file exists and has content
        if (destFile.exists() && destFile.length() > 0) {
            String fileUri = Uri.fromFile(destFile).toString();
            android.util.Log.d("AvatarPicker", "Returning file URI: " + fileUri);
            return fileUri;
        } else {
            android.util.Log.e("AvatarPicker", "Destination file is empty or doesn't exist");
            return null;
        }

    } catch (SecurityException e) {
        // Re-throw to be handled in onActivityResult
        android.util.Log.e("AvatarPicker", "SecurityException during image copy - rethrowing", e);
        throw e;
    } catch (Exception e) {
        android.util.Log.e("AvatarPicker", "Error copying image: " + e.getMessage(), e);
    } finally {
        // Clean up streams
        try {
            if (outputStream != null) outputStream.close();
            if (inputStream != null) inputStream.close();
        } catch (Exception e) {
            android.util.Log.e("AvatarPicker", "Error closing streams: " + e.getMessage());
        }
    }
    return null;
  }
}
```

ContactsAdapter.java

```
package com.example.comp1786_logbook_ex3_connectdatabaseapplication.activity;

import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.CheckBox;
import android.widget.ImageView;
import android.widget.TextView;

import androidx.annotation.NonNull;
import androidx.recyclerview.widget.RecyclerView;

import com.example.comp1786_logbook_ex3_connectdatabaseapplication.R;
import com.example.comp1786_logbook_ex3_connectdatabaseapplication.helper.ResUtils;
import com.example.comp1786_logbook_ex3_connectdatabaseapplication.model.Contact;

import java.util.ArrayList;
import java.util.HashSet;
import java.util.List;
import java.util.Set;

public class ContactsAdapter extends RecyclerView.Adapter<ContactsAdapter.VH> {

    private final List<Contact> items = new ArrayList<>();
    private final OnItemClick onItemClick;
    private final OnItemDelete onItemDelete;
    private final OnFavoriteToggle onFavoriteToggle;

    private boolean selectionMode = false;
    private final Set<Long> selectedIds = new HashSet<>();

    public interface OnItemClick { void onClick(Contact c); }
    public interface OnItemDelete { void onDelete(Contact c); }
    public interface OnFavoriteToggle { void onToggle(Contact c); }
```

```
public ContactsAdapter(OnItemClick cb, OnItemDelete delCb, OnFavoriteToggle favCb) {
    this.onItemClick = cb;
    this.onItemDelete = delCb;
    this.onFavoriteToggle = favCb;
}

public void setItems(List<Contact> list) {
    items.clear();
    if (list != null) items.addAll(list);
    // Clear selection when data set changes
    selectedIds.clear();
    selectionMode = false;
    notifyDataSetChanged();
}

public void setSelectionMode(boolean enabled) {
    if (selectionMode != enabled) {
        selectionMode = enabled;
        if (!enabled) {
            selectedIds.clear();
        }
        notifyDataSetChanged();
    }
}

public boolean isSelectionMode() {
    return selectionMode;
}

public Set<Long> getSelectedIds() {
    return new HashSet<>(selectedIds);
}

public void toggleSelection(long id) {
    if (selectedIds.contains(id)) {
        selectedIds.remove(id);
    } else {
        selectedIds.add(id);
    }
    notifyDataSetChanged();
}

@NonNull
@Override
public VH onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
    View v = LayoutInflater.from(parent.getContext()).inflate(R.layout.item_contact, parent, false);
    VH vh = new VH(v, onItemClick, onFavoriteToggle, selectedIds, this);
    // attach delete callback as tag so the static VH can access it
    v.setTag(onItemDelete);
    return vh;
}

@Override
public void onBindViewHolder(@NonNull VH holder, int position) {
    holder.bind(items.get(position), selectionMode, selectedIds.contains(items.get(position).id));
}

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

static class VH extends RecyclerView.ViewHolder {
    ImageView img;
    TextView txtName, txtEmail;
    View btnDelete;
    ImageView imgFavorite;
    CheckBox cbSelect;
    OnItemClick cb;
    OnFavoriteToggle favCb;
    Set<Long> selectedIds;
    ContactsAdapter adapter;

    public VH(@NonNull View itemView, OnItemClick cb, OnFavoriteToggle favCb, Set<Long> selectedIds, ContactsAdapter adapter) {
        super(itemView);
```

```
            img = itemView.findViewById(R.id.imgAvatar);
            txtName = itemView.findViewById(R.id.txtName);
            txtEmail = itemView.findViewById(R.id.txtEmail);
            btnDelete = itemView.findViewById(R.id.btnDelete);
            imgFavorite = itemView.findViewById(R.id.imgFavorite);
            cbSelect = itemView.findViewById(R.id.cbSelect);
            this.cb = cb;
            this.favCb = favCb;
            this.selectedIds = selectedIds;
            this.adapter = adapter;
        }
        public void bind(Contact c, boolean selectionMode, boolean isSelected) {
            // Handle uploaded images (file:// or content://) and built-in avatars
            if (c.avatarName != null &&
                (c.avatarName.startsWith("content://") || c.avatarName.startsWith("file://"))) {
                try {
                    img.setImageURI(android.net.Uri.parse(c.avatarName));
                } catch (SecurityException e) {
                    android.util.Log.e("ContactsAdapter", "SecurityException loading image: " + c.avatarName, e);
                    // Fall back to default avatar
                    img.setImageResource(R.drawable.avatar_1);
                } catch (Exception e) {
                    android.util.Log.e("ContactsAdapter", "Error loading image: " + c.avatarName, e);
                    img.setImageResource(R.drawable.avatar_1);
                }
            } else {
                int res = ResUtils.nameToDrawable(itemView.getContext(), c.avatarName);
                if (res != 0) img.setImageResource(res);
            }

            txtName.setText(c.name);
            txtEmail.setText(c.email);

            // Favorite icon - set click listener BEFORE card click to prevent propagation
            if (imgFavorite != null && favCb != null) {
                imgFavorite.setImageResource(c.isFavorite ? R.drawable.ic_star_filled : R.drawable.ic_star_border);
                imgFavorite.setOnClickListener(v -> {
                    try {
                        android.util.Log.d("ContactsAdapter", "imgFavorite clicked for id=" + c.id + ", name=" + c.name + ", isFavorite=" +
c.isFavorite);
                        // Disable rapid double taps
                        v.setEnabled(false);
                        v.postDelayed(() -> v.setEnabled(true), 300);

                        // Delegate the actual toggle to the activity/fragment
                        favCb.onToggle(c);
                    } catch (Exception ex) {
                        android.util.Log.e("ContactsAdapter", "Error toggling favorite", ex);
                    }
                });
            }

            // Selection checkbox visibility/state
            if (cbSelect != null) {
                cbSelect.setVisibility(selectionMode ? View.VISIBLE : View.GONE);
                cbSelect.setChecked(isSelected);
                cbSelect.setOnClickListener(v -> {
                    // Toggle selection state
                    if (selectedIds.contains(c.id)) {
                        selectedIds.remove(c.id);
                    } else {
                        selectedIds.add(c.id);
                    }
                    adapter.notifyDataSetChanged();
                });
            }

            // Card click - set AFTER other click listeners
            View cardContainer = itemView.findViewById(R.id.cardContainer);
            if (cardContainer != null) {
                cardContainer.setOnClickListener(v -> cb.onClick(c));
            } else {
                itemView.setOnClickListener(v -> cb.onClick(c));
            }
```

```
        itemView.setOnLongClickListener(v -> {
            // Long press can be handled by activity via adapter state; no-op here
            return false;
        });

        btnDelete.setOnClickListener(v -> {
            Object tag = itemView.getTag();
            if (tag instanceof OnItemDelete) {
                ((OnItemDelete) tag).onDelete(c);
            }
        });
    }
  }
}
```

ContactsListActivity.java

```
package com.example.comp1786_logbook_ex3_connectdatabaseapplication.activity;

import android.content.DialogInterface;
import android.content.Intent;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.TextView;
import android.widget.Toast;

import androidx.annotation.Nullable;
import androidx.appcompat.app.AlertDialog;
import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.SearchView;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import com.example.comp1786_logbook_ex3_connectdatabaseapplication.R;
import com.example.comp1786_logbook_ex3_connectdatabaseapplication.model.AppDatabase;
import com.example.comp1786_logbook_ex3_connectdatabaseapplication.model.Contact;
import com.example.comp1786_logbook_ex3_connectdatabaseapplication.model.ContactDao;
import com.example.comp1786_logbook_ex3_connectdatabaseapplication.helper.Constants;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.IntentFilter;

import java.util.ArrayList;
import java.util.Calendar;
import java.util.List;
import java.util.Set;
import java.util.concurrent.Executors;

public class ContactsListActivity extends AppCompatActivity implements View.OnClickListener {

    private RecyclerView rv;
    private ContactsAdapter adapter;
    private ContactDao dao;
    private TextView tvContactCount;
    private static final int REQ_ADD = 1;

    private enum SortMode { NAME_ASC, NAME_DESC, CREATED_DESC, CREATED_ASC }
    private SortMode currentSort = SortMode.NAME_ASC;
    private boolean showFavoritesOnly = false;
    private boolean filterBirthMonth = false;
    private String currentSearchQuery = "";

    private final BroadcastReceiver contactsChangedReceiver = new BroadcastReceiver() {
        @Override
        public void onReceive(Context context, Intent intent) {
            if (Constants.ACTION_CONTACTS_CHANGED.equals(intent.getAction())) {
                loadContacts();
            }
        }
    };
```

```java
@Override
protected void onCreate(@Nullable Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_contacts_list);
    dao = AppDatabase.getInstance(this).contactDao();

    // Setup toolbar
    com.google.android.material.appbar.MaterialToolbar toolbar = findViewById(R.id.toolbar);
    setSupportActionBar(toolbar);

    rv = findViewById(R.id.rvContacts);
    rv.setLayoutManager(new LinearLayoutManager(this));
    adapter = new ContactsAdapter(this::onItemClick, this::onItemDelete, this::onFavoriteToggle);
    rv.setAdapter(adapter);

    tvContactCount = findViewById(R.id.tvContactCount);
    findViewById(R.id.fabAdd).setOnClickListener(this);

    loadContacts();


}

@Override
protected void onDestroy() {
    super.onDestroy();
    unregisterReceiver(contactsChangedReceiver);
}

private void loadContacts() {
    Executors.newSingleThreadExecutor().execute(() -> {
        List<Contact> all;

        // First, get the base list based on sorting
        switch (currentSort) {
            case NAME_DESC:
                all = dao.getAllSortedByNameDesc();
                break;
            case CREATED_DESC:
                all = dao.getAllSortedByCreatedAtDesc();
                break;
            case CREATED_ASC:
                all = dao.getAllSortedByCreatedAtAsc();
                break;
            case NAME_ASC:
            default:
                all = dao.getAllSortedByNameAsc();
                break;
        }

        // Apply filters in memory to allow combinations
        if (all != null) {
            List<Contact> filtered = new ArrayList<>(all);

            // Apply search filter
            if (currentSearchQuery != null && !currentSearchQuery.trim().isEmpty()) {
                String query = currentSearchQuery.trim().toLowerCase();
                filtered = new ArrayList<>();
                for (Contact c : all) {
                    if ((c.name != null && c.name.toLowerCase().contains(query)) ||
                        (c.email != null && c.email.toLowerCase().contains(query))) {
                        filtered.add(c);
                    }
                }
            }

            // Apply favorites filter
            if (showFavoritesOnly) {
                List<Contact> favFiltered = new ArrayList<>();
                for (Contact c : filtered) {
                    if (c.isFavorite) {
                        favFiltered.add(c);
                    }
                }
```

```
                filtered = favFiltered;
            }

            // Apply birth month filter
            if (filterBirthMonth) {
                int month = Calendar.getInstance().get(Calendar.MONTH) + 1;
                String monthStr = month < 10 ? "0" + month : String.valueOf(month);
                List<Contact> birthFiltered = new ArrayList<>();
                for (Contact c : filtered) {
                    if (c.dob != null && c.dob.length() >= 5) {
                        String contactMonth = c.dob.substring(3, 5);
                        if (monthStr.equals(contactMonth)) {
                            birthFiltered.add(c);
                        }
                    }
                }
                filtered = birthFiltered;
            }

            all = filtered;
        }

        List<Contact> finalAll = all;
        runOnUiThread(() -> {
            adapter.setItems(finalAll);
            // Update contact count
            int count = finalAll != null ? finalAll.size() : 0;
            String countText = count + (count == 1 ? " contact" : " contacts");
            if (tvContactCount != null) {
                tvContactCount.setText(countText);
            }
        });
    });
}

private void onItemClick(Contact c) {
    // Open detail activity for the selected contact
    Intent i = new Intent(this, DetailContactActivity.class);
    i.putExtra(DetailContactActivity.EXTRA_ID, c.id);
    startActivityForResult(i, REQ_ADD);
}

private void onItemDelete(Contact c) {
    // Confirm deletion
    new AlertDialog.Builder(this)
            .setTitle(R.string.delete)
            .setMessage(getString(R.string.confirm_delete_single, c.name))
            .setPositiveButton(android.R.string.ok, (dialog, which) -> {
                Executors.newSingleThreadExecutor().execute(() -> {
                    dao.delete(c);
                    runOnUiThread(() -> {
                        Intent intent = new Intent(Constants.ACTION_CONTACTS_CHANGED);
                        sendBroadcast(intent);
                        Toast.makeText(this, "Deleted: " + c.name, Toast.LENGTH_SHORT).show();
                    });
                });
            })
            .setNegativeButton(android.R.string.cancel, null)
            .show();
}

private void onFavoriteToggle(Contact c) {
    android.util.Log.d("ContactsListActivity", "onFavoriteToggle called for: " + c.name + ", current: " + c.isFavorite + ", id=" + c.id);
    Executors.newSingleThreadExecutor().execute(() -> {
        boolean newFav = !c.isFavorite;
        long ts = System.currentTimeMillis();
        if (c.id > 0) {
            // Use dedicated update to avoid replacing the row
            dao.updateFavorite(c.id, newFav ? 1 : 0, ts);
        } else {
            // fallback: set on object and upsert
            c.isFavorite = newFav;
            c.updatedAt = ts;
            long newId = dao.upsert(c);
```

```
            if (newId > 0) c.id = newId;
        }

        // Update in-memory model for immediate feedback (adapter reload will use DB data)
        c.isFavorite = newFav;
        c.updatedAt = ts;

        android.util.Log.d("ContactsListActivity", "Favorite updated in DB for: " + c.name + ", new value: " + c.isFavorite);
        runOnUiThread(() -> {
            Intent intent = new Intent(Constants.ACTION_CONTACTS_CHANGED);
            sendBroadcast(intent);
            Toast.makeText(this, c.name + (c.isFavorite ? " added to favorites" : " removed from favorites"), Toast.LENGTH_SHORT).show();
        });
    });
}

@Override
public void onClick(View v) {
    if (v.getId() == R.id.fabAdd) {
        startActivityForResult(new Intent(this, EditContactActivity.class), REQ_ADD);
    }
}

@Override
protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    if (requestCode == REQ_ADD && resultCode == RESULT_OK) {
        // Reload contacts immediately to show updated data
        loadContacts();
    }
}

@Override
public boolean onPrepareOptionsMenu(Menu menu) {
    // Update checkable menu items
    MenuItem favItem = menu.findItem(R.id.action_filter_favorites);
    if (favItem != null) {
        favItem.setChecked(showFavoritesOnly);
    }

    MenuItem birthItem = menu.findItem(R.id.action_filter_birth_month);
    if (birthItem != null) {
        birthItem.setChecked(filterBirthMonth);
    }

    // Show/hide delete selected based on selection mode
    MenuItem deleteSelectedItem = menu.findItem(R.id.action_delete_selected);
    if (deleteSelectedItem != null) {
        deleteSelectedItem.setVisible(adapter != null && adapter.isSelectionMode());
    }

    return super.onPrepareOptionsMenu(menu);
}

@Override
public boolean onCreateOptionsMenu(Menu menu) {
    getMenuInflater().inflate(R.menu.menu_contacts_list, menu);

    MenuItem searchItem = menu.findItem(R.id.action_search);
    SearchView searchView = (SearchView) searchItem.getActionView();
    searchView.setQueryHint(getString(R.string.search_hint));
    searchView.setOnQueryTextListener(new SearchView.OnQueryTextListener() {
        @Override
        public boolean onQueryTextSubmit(String query) {
            currentSearchQuery = query;
            loadContacts();
            return true;
        }

        @Override
        public boolean onQueryTextChange(String newText) {
            currentSearchQuery = newText;
            loadContacts();
            return true;
```

```
        }
    });
    return true;
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
    int id = item.getItemId();
    if (id == R.id.action_sort_name_asc) {
        currentSort = SortMode.NAME_ASC;
        loadContacts();
        return true;
    } else if (id == R.id.action_sort_name_desc) {
        currentSort = SortMode.NAME_DESC;
        loadContacts();
        return true;
    } else if (id == R.id.action_sort_created_desc) {
        currentSort = SortMode.CREATED_DESC;
        loadContacts();
        return true;
    } else if (id == R.id.action_sort_created_asc) {
        currentSort = SortMode.CREATED_ASC;
        loadContacts();
        return true;
    } else if (id == R.id.action_filter_favorites) {
        showFavoritesOnly = !showFavoritesOnly;
        item.setChecked(showFavoritesOnly);
        loadContacts();
        return true;
    } else if (id == R.id.action_filter_birth_month) {
        filterBirthMonth = !filterBirthMonth;
        item.setChecked(filterBirthMonth);
        loadContacts();
        return true;
    } else if (id == R.id.action_clear_all) {
        confirmClearAll();
        return true;
    } else if (id == R.id.action_multi_select) {
        toggleSelectionMode();
        return true;
    } else if (id == R.id.action_delete_selected) {
        deleteSelectedContacts();
        return true;
    }
    return super.onOptionsItemSelected(item);
}

private void toggleSelectionMode() {
    boolean enable = !adapter.isSelectionMode();
    adapter.setSelectionMode(enable);
    invalidateOptionsMenu(); // Refresh menu to show/hide delete selected
    if (!enable) {
        Toast.makeText(this, R.string.selection_mode_off, Toast.LENGTH_SHORT).show();
    } else {
        Toast.makeText(this, R.string.selection_mode_on, Toast.LENGTH_SHORT).show();
    }
}

private void confirmClearAll() {
    new AlertDialog.Builder(this)
            .setTitle(R.string.clear_all_title)
            .setMessage(R.string.clear_all_message)
            .setPositiveButton(android.R.string.ok, (dialog, which) ->
                    Executors.newSingleThreadExecutor().execute(() -> {
                        dao.clearAll();
                        runOnUiThread(() -> {
                            Intent intent = new Intent(Constants.ACTION_CONTACTS_CHANGED);
                            sendBroadcast(intent);
                            Toast.makeText(this, R.string.cleared_all, Toast.LENGTH_SHORT).show();
                        });
                    }))
            .setNegativeButton(android.R.string.cancel, null)
            .show();
}
```

```
    private void deleteSelectedContacts() {
        Set<Long> ids = adapter.getSelectedIds();
        if (ids.isEmpty()) {
            Toast.makeText(this, R.string.no_items_selected, Toast.LENGTH_SHORT).show();
            return;
        }
        new AlertDialog.Builder(this)
                .setTitle(R.string.delete)
                .setMessage(getString(R.string.confirm_delete_multiple, ids.size()))
                .setPositiveButton(android.R.string.ok, (dialog, which) ->
                        Executors.newSingleThreadExecutor().execute(() -> {
                            dao.deleteByIds(new ArrayList<>(ids));
                            runOnUiThread(() -> {
                                Intent intent = new Intent(Constants.ACTION_CONTACTS_CHANGED);
                                sendBroadcast(intent);
                                Toast.makeText(this, R.string.deleted_selected, Toast.LENGTH_SHORT).show();
                                adapter.setSelectionMode(false);
                            });
                        }))
                .setNegativeButton(android.R.string.cancel, null)
                .show();
    }
}
```

DetailContactActivity.java

```
package com.example.comp1786_logbook_ex3_connectdatabaseapplication.activity;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.TextView;
import android.widget.Toast;

import androidx.annotation.Nullable;
import androidx.appcompat.app.AppCompatActivity;

import com.example.comp1786_logbook_ex3_connectdatabaseapplication.R;
import com.example.comp1786_logbook_ex3_connectdatabaseapplication.helper.ResUtils;
import com.example.comp1786_logbook_ex3_connectdatabaseapplication.model.AppDatabase;
import com.example.comp1786_logbook_ex3_connectdatabaseapplication.model.Contact;
import com.example.comp1786_logbook_ex3_connectdatabaseapplication.model.ContactDao;
import com.google.android.material.appbar.MaterialToolbar;
import com.google.android.material.imageview.ShapeableImageView;
import com.example.comp1786_logbook_ex3_connectdatabaseapplication.helper.Constants;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.IntentFilter;

import java.text.DateFormat;
import java.util.Date;
import java.util.concurrent.Executors;

public class DetailContactActivity extends AppCompatActivity implements View.OnClickListener {

    public static final String EXTRA_ID = "contact_id";
    private long contactId = -1;
    private ContactDao dao;
    private Contact contact;
    private ShapeableImageView img;
    private TextView tvName, tvDob, tvEmail, tvLastUpdated;
    private static final int REQ_EDIT = 11;

    private final BroadcastReceiver contactsChangedReceiver = new BroadcastReceiver() {
        @Override
        public void onReceive(Context context, Intent intent) {
            if (Constants.ACTION_CONTACTS_CHANGED.equals(intent.getAction())) {
                loadContact();
            }
        }
    };

    @Override
```

```
    protected void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_contact_detail);

        MaterialToolbar toolbar = findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);
        toolbar.setNavigationOnClickListener(v -> finish());

        img = findViewById(R.id.imgAvatarDetail);
        tvName = findViewById(R.id.tvName);
        tvDob = findViewById(R.id.tvDob);
        tvEmail = findViewById(R.id.tvEmail);
        tvLastUpdated = findViewById(R.id.tvLastUpdated);
        findViewById(R.id.btnEdit).setOnClickListener(this);

        dao = AppDatabase.getInstance(this).contactDao();
        contactId = getIntent().getLongExtra(EXTRA_ID, -1);
        loadContact();

        // Register broadcast receiver with proper API level check
        if (android.os.Build.VERSION.SDK_INT >= android.os.Build.VERSION_CODES.TIRAMISU) {
            registerReceiver(contactsChangedReceiver, new IntentFilter(Constants.ACTION_CONTACTS_CHANGED),
Context.RECEIVER_NOT_EXPORTED);
        } else {
            registerReceiver(contactsChangedReceiver, new IntentFilter(Constants.ACTION_CONTACTS_CHANGED));
        }
    }

    @Override
    protected void onDestroy() {
        super.onDestroy();
        unregisterReceiver(contactsChangedReceiver);
    }

    private void loadContact() {
        Executors.newSingleThreadExecutor().execute(() -> {
            contact = dao.getById(contactId);
            if (contact != null) {
                runOnUiThread(this::showContact);
            } else {
                runOnUiThread(() -> {
                    Toast.makeText(this, "Contact not found", Toast.LENGTH_SHORT).show();
                    finish();
                });
            }
        });
    }

    private void showContact() {
        getSupportActionBar().setTitle(contact.name);

        // Handle uploaded images (file:// or content://) and built-in avatars
        if (contact.avatarName != null &&
            (contact.avatarName.startsWith("content://") || contact.avatarName.startsWith("file://"))) {
            try {
                img.setImageURI(android.net.Uri.parse(contact.avatarName));
            } catch (SecurityException e) {
                android.util.Log.e("DetailContact", "SecurityException loading image: " + contact.avatarName, e);
                // Fall back to default avatar
                img.setImageResource(R.drawable.avatar_1);
            } catch (Exception e) {
                android.util.Log.e("DetailContact", "Error loading image: " + contact.avatarName, e);
                img.setImageResource(R.drawable.avatar_1);
            }
        } else {
            int res = ResUtils.nameToDrawable(this, contact.avatarName);
            if (res != 0) img.setImageResource(res);
        }

        tvName.setText(contact.name);
        tvDob.setText(contact.dob);
        tvEmail.setText(contact.email);
```

```
        if (tvLastUpdated != null) {
            long ts = contact.updatedAt != 0L ? contact.updatedAt : contact.createdAt;
            if (ts != 0L) {
                String formatted = DateFormat.getDateTimeInstance().format(new Date(ts));
                tvLastUpdated.setText(getString(R.string.last_updated_format, formatted));
            }
        }
    }

    @Override
    public void onClick(View v) {
        Intent i = new Intent(this, EditContactActivity.class);
        i.putExtra("contact_id", contactId);
        startActivityForResult(i, REQ_EDIT);
    }

    @Override
    protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {
        super.onActivityResult(requestCode, resultCode, data);
        if (requestCode == REQ_EDIT && resultCode == RESULT_OK) {
            // Reload contact immediately to show updated data
            loadContact();
            // Set result OK so ContactsListActivity knows to refresh
            setResult(RESULT_OK);
        }
    }
}
```

EditContactActivity

```
package com.example.comp1786_logbook_ex3_connectdatabaseapplication.activity;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.EditText;
import android.widget.ImageView;
import android.widget.Toast;

import androidx.annotation.Nullable;
import androidx.appcompat.app.AppCompatActivity;

import com.example.comp1786_logbook_ex3_connectdatabaseapplication.R;
import com.example.comp1786_logbook_ex3_connectdatabaseapplication.helper.ResUtils;
import com.example.comp1786_logbook_ex3_connectdatabaseapplication.model.AppDatabase;
import com.example.comp1786_logbook_ex3_connectdatabaseapplication.model.Contact;
import com.example.comp1786_logbook_ex3_connectdatabaseapplication.model.ContactDao;
import com.google.android.material.appbar.MaterialToolbar;
import com.example.comp1786_logbook_ex3_connectdatabaseapplication.helper.Constants;
import android.app.DatePickerDialog;
import java.util.Calendar;

import java.util.concurrent.Executors;

public class EditContactActivity extends AppCompatActivity implements View.OnClickListener {

    private EditText etName, etDob, etEmail;
    private ImageView imgAvatar;
    private String avatarName = "avatar_1";
    private ContactDao dao;
    private Contact editingContact;
    private static final int REQ_PICK_AVATAR = 2001;

    @Override
    protected void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_edit_contact);

        MaterialToolbar toolbar = findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);
        toolbar.setNavigationOnClickListener(v -> finish());

        etName = findViewById(R.id.etName);
```

```
    etDob = findViewById(R.id.etDob);
    etEmail = findViewById(R.id.etEmail);
    imgAvatar = findViewById(R.id.imgPickAvatar);

    etDob.setOnClickListener(this);
    findViewById(R.id.btnChooseAvatar).setOnClickListener(this);
    findViewById(R.id.btnSave).setOnClickListener(this);

    dao = AppDatabase.getInstance(this).contactDao();

    updateAvatarImage();
    // If editing existing contact, load it
    long id = getIntent().getLongExtra("contact_id", -1);
    if (id != -1) {
        getSupportActionBar().setTitle("Edit Contact");
        Executors.newSingleThreadExecutor().execute(() -> {
            Contact found = dao.getById(id);
            if (found != null) {
                editingContact = found;
                runOnUiThread(() -> {
                    etName.setText(editingContact.name);
                    etDob.setText(editingContact.dob);
                    etEmail.setText(editingContact.email);
                    avatarName = editingContact.avatarName != null ? editingContact.avatarName : avatarName;
                    updateAvatarImage();
                });
            }
        });
    } else {
        getSupportActionBar().setTitle("Add Contact");
    }
}

private void updateAvatarImage() {
    if (avatarName.startsWith("content://") || avatarName.startsWith("file://")) {
        try {
            imgAvatar.setImageURI(android.net.Uri.parse(avatarName));
        } catch (SecurityException e) {
            android.util.Log.e("EditContact", "SecurityException loading image: " + avatarName, e);
            // Fall back to default avatar
            imgAvatar.setImageResource(R.drawable.avatar_1);
            Toast.makeText(this, "Unable to load image. Please select a new one.", Toast.LENGTH_SHORT).show();
        } catch (Exception e) {
            android.util.Log.e("EditContact", "Error loading image: " + avatarName, e);
            imgAvatar.setImageResource(R.drawable.avatar_1);
        }
    } else {
        int res = ResUtils.nameToDrawable(this, avatarName);
        if (res != 0) imgAvatar.setImageResource(res);
    }
}

@Override
public void onClick(View v) {
    int id = v.getId();
    if (id == R.id.btnChooseAvatar) {
        AvatarPickerActivity.startForResult(this, REQ_PICK_AVATAR, avatarName);
    } else if (id == R.id.etDob) {
        showDatePickerDialog();
    } else if (id == R.id.btnSave) {
        final String name = etName.getText().toString().trim();
        final String dob = etDob.getText().toString().trim();
        final String email = etEmail.getText().toString().trim();
        if (name.isEmpty()) {
            Toast.makeText(this, "Name required", Toast.LENGTH_SHORT).show();
            return;
        }
        if (dob.isEmpty()) {
            Toast.makeText(this, "Date of birth required", Toast.LENGTH_SHORT).show();
            return;
        }
        if (!email.isEmpty() && !android.util.Patterns.EMAIL_ADDRESS.matcher(email).matches()) {
            Toast.makeText(this, "Invalid email", Toast.LENGTH_SHORT).show();
            return;
```

```
        }

        final Contact c;
        if (editingContact != null) {
            // Update existing contact
            editingContact.name = name;
            editingContact.dob = dob;
            editingContact.email = email;
            editingContact.avatarName = avatarName;
            editingContact.updatedAt = System.currentTimeMillis();
            c = editingContact;
        } else {
            // Create new contact
            c = new Contact(name, dob, email, avatarName);
        }

        final Contact toSave = c;
        final boolean isNewContact = (editingContact == null);

        Executors.newSingleThreadExecutor().execute(() -> {
            dao.upsert(toSave);
            runOnUiThread(() -> {
                // Send broadcast to update all screens
                Intent intent = new Intent(Constants.ACTION_CONTACTS_CHANGED);
                sendBroadcast(intent);

                // Show appropriate message
                String message = isNewContact ? "Contact added" : "Contact updated";
                Toast.makeText(this, message, Toast.LENGTH_SHORT).show();

                setResult(RESULT_OK);
                finish();
            });
        });
    }
}

private void showDatePickerDialog() {
    Calendar calendar = Calendar.getInstance();
    int year = calendar.get(Calendar.YEAR);
    int month = calendar.get(Calendar.MONTH);
    int day = calendar.get(Calendar.DAY_OF_MONTH);

    DatePickerDialog datePickerDialog = new DatePickerDialog(this,
            (view, year1, monthOfYear, dayOfMonth) -> {
                // Format: dd/MM/yyyy
                String dayStr = dayOfMonth < 10 ? "0" + dayOfMonth : String.valueOf(dayOfMonth);
                String monthStr = (monthOfYear + 1) < 10 ? "0" + (monthOfYear + 1) : String.valueOf(monthOfYear + 1);
                String selectedDate = dayStr + "/" + monthStr + "/" + year1;
                etDob.setText(selectedDate);
            }, year, month, day);
    datePickerDialog.show();
}

@Override
protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    if (requestCode == REQ_PICK_AVATAR && resultCode == RESULT_OK && data != null) {
        String chosen = data.getStringExtra(AvatarPickerActivity.EXTRA_CHOSEN);
        if (chosen != null) {
            avatarName = chosen;
            updateAvatarImage();

            // If editing existing contact, update avatar immediately in database
            // This allows other screens to see the new avatar before saving
            if (editingContact != null) {
                editingContact.avatarName = avatarName;
                Executors.newSingleThreadExecutor().execute(() -> {
                    dao.upsert(editingContact);
                    runOnUiThread(() -> {
                        // Broadcast the change so DetailActivity and ListActivity update immediately
                        Intent intent = new Intent(Constants.ACTION_CONTACTS_CHANGED);
                        sendBroadcast(intent);
                        Toast.makeText(this, "Avatar updated", Toast.LENGTH_SHORT).show();
```

```
                    });
                });
            }
        }
    }
}
```

ResUtils.java

```
package com.example.comp1786_logbook_ex3_connectdatabaseapplication.helper;

import android.content.Context;
import androidx.annotation.ArrayRes;

import java.util.ArrayList;
import java.util.List;

public class ResUtils {
    public static int nameToDrawable(Context ctx, String name) {
        if (name == null) return 0;
        return ctx.getResources().getIdentifier(name, "drawable", ctx.getPackageName());
    }

    public static List<String> avatarNamesFromArray(Context ctx, @ArrayRes int arrRes) {
        List<String> out = new ArrayList<>();
        android.content.res.TypedArray ta = ctx.getResources().obtainTypedArray(arrRes);
        for (int i = 0; i < ta.length(); i++) {
            int resId = ta.getResourceId(i, 0);
            if (resId != 0) out.add(ctx.getResources().getResourceEntryName(resId));
        }
        ta.recycle();
        return out;
    }
}
```

AppDatabase.java

```
package com.example.comp1786_logbook_ex3_connectdatabaseapplication.model;

import androidx.room.Entity;
import androidx.room.PrimaryKey;

@Entity(tableName = "contacts")
public class Contact {
    @PrimaryKey(autoGenerate = true)
    public long id;

    public String name;
    public String dob;
    public String email;
    public String avatarName;

    // New fields: favorite flag and timestamps
    public boolean isFavorite;
    public long createdAt;
    public long updatedAt;

    public Contact(String name, String dob, String email, String avatarName) {
        this.name = name;
        this.dob = dob;
        this.email = email;
        this.avatarName = avatarName;
        long now = System.currentTimeMillis();
        this.createdAt = now;
        this.updatedAt = now;
        this.isFavorite = false;
    }

    // Convenience constructor to explicitly set timestamps and favorite
    public Contact(String name, String dob, String email, String avatarName, boolean isFavorite, long createdAt, long updatedAt) {
        this.name = name;
        this.dob = dob;
        this.email = email;
```

```
        this.avatarName = avatarName;
        this.isFavorite = isFavorite;
        this.createdAt = createdAt;
        this.updatedAt = updatedAt;
    }

    // Room requires a no-arg constructor for some uses
    public Contact() { }
}
```

Contact.java

```
package com.example.comp1786_logbook_ex3_connectdatabaseapplication.model;

import androidx.room.Entity;
import androidx.room.PrimaryKey;

@Entity(tableName = "contacts")
public class Contact {
    @PrimaryKey(autoGenerate = true)
    public long id;

    public String name;
    public String dob;
    public String email;
    public String avatarName;

    // New fields: favorite flag and timestamps
    public boolean isFavorite;
    public long createdAt;
    public long updatedAt;

    public Contact(String name, String dob, String email, String avatarName) {
        this.name = name;
        this.dob = dob;
        this.email = email;
        this.avatarName = avatarName;
        long now = System.currentTimeMillis();
        this.createdAt = now;
        this.updatedAt = now;
        this.isFavorite = false;
    }

    // Convenience constructor to explicitly set timestamps and favorite
    public Contact(String name, String dob, String email, String avatarName, boolean isFavorite, long createdAt, long updatedAt) {
        this.name = name;
        this.dob = dob;
        this.email = email;
        this.avatarName = avatarName;
        this.isFavorite = isFavorite;
        this.createdAt = createdAt;
        this.updatedAt = updatedAt;
    }

    // Room requires a no-arg constructor for some uses
    public Contact() { }
}
```

ContactFragment.java

```
package com.example.comp1786_logbook_ex3_connectdatabaseapplication.ui;

import android.content.Context;
import android.content.Intent;
import android.os.Bundle;
import android.view.inputmethod.InputMethodManager;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.EditText;
import android.widget.ImageButton;
import android.widget.ImageView;
```

```java
import android.widget.TextView;
import android.widget.Toast;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.core.widget.NestedScrollView;
import androidx.fragment.app.Fragment;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import com.example.comp1786_logbook_ex3_connectdatabaseapplication.R;
import com.example.comp1786_logbook_ex3_connectdatabaseapplication.activity.ContactsAdapter;
import com.example.comp1786_logbook_ex3_connectdatabaseapplication.activity.DetailContactActivity;
import com.example.comp1786_logbook_ex3_connectdatabaseapplication.activity.EditContactActivity;
import com.example.comp1786_logbook_ex3_connectdatabaseapplication.helper.Constants;
import com.example.comp1786_logbook_ex3_connectdatabaseapplication.model.AppDatabase;
import com.example.comp1786_logbook_ex3_connectdatabaseapplication.model.Contact;
import com.example.comp1786_logbook_ex3_connectdatabaseapplication.model.ContactDao;
import com.google.android.material.floatingactionbutton.FloatingActionButton;

import java.util.ArrayList;
import java.util.Calendar;
import java.util.List;
import java.util.concurrent.Executors;

public class ContactsFragment extends Fragment implements View.OnClickListener {

    private RecyclerView rv;
    private ContactsAdapter adapter;
    private ContactDao dao;
    private TextView tvContactCount;
    private static final int REQ_ADD = 1;

    private enum SortMode { NAME_ASC, NAME_DESC, CREATED_DESC, CREATED_ASC }
    private SortMode currentSort = SortMode.NAME_ASC;
    private boolean showFavoritesOnly = false;
    private boolean filterBirthMonth = false;
    private String currentSearchQuery = "";

    private EditText etSearch;
    private ImageView imgSearchIcon;
    private ImageButton btnSort;

    @Nullable
    @Override
    public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup container, @Nullable Bundle savedInstanceState) {
        View v = inflater.inflate(R.layout.fragment_contacts, container, false);
        dao = AppDatabase.getInstance(requireContext()).contactDao();

        rv = v.findViewById(R.id.rvContactsFrag);
        rv.setLayoutManager(new LinearLayoutManager(requireContext()));
        adapter = new ContactsAdapter(this::onItemClick, this::onItemDelete, this::onFavoriteToggle);
        rv.setAdapter(adapter);

        tvContactCount = v.findViewById(R.id.tvContactCountFrag);
        FloatingActionButton fab = v.findViewById(R.id.fabAddFrag);
        if (fab != null) fab.setOnClickListener(this);

        // New search widgets
        etSearch = v.findViewById(R.id.etSearchFrag);
        imgSearchIcon = v.findViewById(R.id.imgSearchIcon);
        btnSort = v.findViewById(R.id.btnSortFrag);

        if (imgSearchIcon != null) imgSearchIcon.setOnClickListener(view -> {
            if (etSearch != null) {
                etSearch.requestFocus();
                openKeyboard(requireContext(), etSearch);
            }
        });

        if (etSearch != null) {
            etSearch.setOnFocusChangeListener((view, hasFocus) -> {
                if (hasFocus) openKeyboard(requireContext(), etSearch);
            });
```

```
        etSearch.addTextChangedListener(new android.text.TextWatcher() {
            @Override
            public void beforeTextChanged(CharSequence s, int start, int count, int after) {}

            @Override
            public void onTextChanged(CharSequence s, int start, int before, int count) {
                currentSearchQuery = s != null ? s.toString() : "";
                loadContacts();
            }

            @Override
            public void afterTextChanged(android.text.Editable s) {}
        });
    }

    if (btnSort != null) {
        btnSort.setOnClickListener(view -> {
            // Toggle between A-Z and Z-A for now
            if (currentSort == SortMode.NAME_ASC) {
                currentSort = SortMode.NAME_DESC;
                btnSort.setImageResource(android.R.drawable.ic_menu_sort_by_size); // visual change
            } else {
                currentSort = SortMode.NAME_ASC;
                btnSort.setImageResource(android.R.drawable.ic_menu_sort_alphabetically);
            }
            loadContacts();
        });
    }

    // empty layout's add button should open editor
    View empty = v.findViewById(R.id.empty_container_contacts);
    if (empty != null) {
        View btn = empty.findViewById(R.id.btnAddContactEmpty);
        if (btn != null) btn.setOnClickListener(view -> startActivityForResult(new Intent(requireContext(), EditContactActivity.class),
REQ_ADD));
    }

    loadContacts();
    return v;
  }

  private static void openKeyboard(Context ctx, View target) {
    target.post(() -> {
        target.requestFocus();
        InputMethodManager imm = (InputMethodManager) ctx.getSystemService(Context.INPUT_METHOD_SERVICE);
        if (imm != null) imm.showSoftInput(target, InputMethodManager.SHOW_IMPLICIT);
    });
  }

  private void loadContacts() {
    Executors.newSingleThreadExecutor().execute(() -> {
        List<Contact> all;

        switch (currentSort) {
            case NAME_DESC:
                all = dao.getAllSortedByNameDesc();
                break;
            case CREATED_DESC:
                all = dao.getAllSortedByCreatedAtDesc();
                break;
            case CREATED_ASC:
                all = dao.getAllSortedByCreatedAtAsc();
                break;
            case NAME_ASC:
            default:
                all = dao.getAllSortedByNameAsc();
                break;
        }

        if (all != null) {
            List<Contact> filtered = new ArrayList<>(all);

            if (currentSearchQuery != null && !currentSearchQuery.trim().isEmpty()) {
                String query = currentSearchQuery.trim().toLowerCase();
```

```
                filtered = new ArrayList<>();
                for (Contact c : all) {
                    if ((c.name != null && c.name.toLowerCase().contains(query)) ||
                        (c.email != null && c.email.toLowerCase().contains(query))) {
                        filtered.add(c);
                    }
                }
            }

            if (showFavoritesOnly) {
                List<Contact> favFiltered = new ArrayList<>();
                for (Contact c : filtered) {
                    if (c.isFavorite) {
                        favFiltered.add(c);
                    }
                }
                filtered = favFiltered;
            }

            if (filterBirthMonth) {
                int month = Calendar.getInstance().get(Calendar.MONTH) + 1;
                String monthStr = month < 10 ? "0" + month : String.valueOf(month);
                List<Contact> birthFiltered = new ArrayList<>();
                for (Contact c : filtered) {
                    if (c.dob != null && c.dob.length() >= 5) {
                        String contactMonth = c.dob.substring(3, 5);
                        if (monthStr.equals(contactMonth)) {
                            birthFiltered.add(c);
                        }
                    }
                }
                filtered = birthFiltered;
            }

            all = filtered;
        }

        List<Contact> finalAll = all;
        if (getActivity() == null) return;
        getActivity().runOnUiThread(() -> {
            adapter.setItems(finalAll);
            int count = finalAll != null ? finalAll.size() : 0;
            String countText = count + (count == 1 ? " contact" : " contacts");
            if (tvContactCount != null) {
                tvContactCount.setText(countText);
            }

            View empty = getView() != null ? getView().findViewById(R.id.empty_container_contacts) : null;
            if (empty != null) {
                empty.setVisibility((finalAll == null || finalAll.isEmpty()) ? View.VISIBLE : View.GONE);
            }
        });
    });
}

private void onItemClick(Contact c) {
    Intent i = new Intent(requireContext(), DetailContactActivity.class);
    i.putExtra(DetailContactActivity.EXTRA_ID, c.id);
    startActivityForResult(i, REQ_ADD);
}

private void onItemDelete(Contact c) {
    Executors.newSingleThreadExecutor().execute(() -> {
        dao.delete(c);
        if (getActivity() == null) return;
        getActivity().runOnUiThread(() -> {
            Intent intent = new Intent(Constants.ACTION_CONTACTS_CHANGED);
            requireActivity().sendBroadcast(intent);
            Toast.makeText(requireContext(), "Deleted: " + c.name, Toast.LENGTH_SHORT).show();
            loadContacts();
        });
    });
}
```

```
    private void onFavoriteToggle(Contact c) {
        Executors.newSingleThreadExecutor().execute(() -> {
            boolean newFav = !c.isFavorite;
            long ts = System.currentTimeMillis();
            if (c.id > 0) {
                dao.updateFavorite(c.id, newFav ? 1 : 0, ts);
            } else {
                c.isFavorite = newFav;
                c.updatedAt = ts;
                dao.upsert(c);
            }
            if (getActivity() == null) return;
            getActivity().runOnUiThread(() -> {
                Intent intent = new Intent(Constants.ACTION_CONTACTS_CHANGED);
                requireActivity().sendBroadcast(intent);
                Toast.makeText(requireContext(), c.name + (newFav ? " added to favorites" : " removed from favorites"),
Toast.LENGTH_SHORT).show();
                loadContacts();
            });
        });
    }

    @Override
    public void onClick(View v) {
        if (v.getId() == R.id.fabAddFrag) {
            startActivityForResult(new Intent(requireContext(), EditContactActivity.class), REQ_ADD);
        }
    }

    @Override
    public void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {
        super.onActivityResult(requestCode, resultCode, data);
        if (requestCode == REQ_ADD && getActivity() != null && resultCode == getActivity().RESULT_OK) {
            loadContacts();
        }
    }
}
```

FavoritesFragment.java

```
package com.example.comp1786_logbook_ex3_connectdatabaseapplication.ui;

import android.content.Intent;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.Toast;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.fragment.app.Fragment;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import com.example.comp1786_logbook_ex3_connectdatabaseapplication.R;
import com.example.comp1786_logbook_ex3_connectdatabaseapplication.activity.ContactsAdapter;
import com.example.comp1786_logbook_ex3_connectdatabaseapplication.activity.DetailContactActivity;
import com.example.comp1786_logbook_ex3_connectdatabaseapplication.helper.Constants;
import com.example.comp1786_logbook_ex3_connectdatabaseapplication.model.AppDatabase;
import com.example.comp1786_logbook_ex3_connectdatabaseapplication.model.Contact;
import com.example.comp1786_logbook_ex3_connectdatabaseapplication.model.ContactDao;

import java.util.List;
import java.util.concurrent.Executors;

public class FavoritesFragment extends Fragment {

    private RecyclerView rv;
    private ContactsAdapter adapter;
    private ContactDao dao;

    @Nullable
```

```
    @Override
    public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup container, @Nullable Bundle savedInstanceState) {
        View v = inflater.inflate(R.layout.fragment_favorites, container, false);
        dao = AppDatabase.getInstance(requireContext()).contactDao();

        rv = v.findViewById(R.id.rvFavorites);
        rv.setLayoutManager(new LinearLayoutManager(requireContext()));
        adapter = new ContactsAdapter(this::onItemClick, this::onItemDelete, this::onFavoriteToggle);
        rv.setAdapter(adapter);

        Button btnBrowse = v.findViewById(R.id.btnBrowseContactsFav);
        if (btnBrowse != null) btnBrowse.setOnClickListener(item -> {
            // switch to contacts tab in host activity
            if (getActivity() instanceof com.example.comp1786_logbook_ex3_connectdatabaseapplication.ui.MainNavActivity) {
                com.google.android.material.bottomnavigation.BottomNavigationView bn =
                        getActivity().findViewById(R.id.bottom_navigation);
                if (bn != null) bn.setSelectedItemId(R.id.nav_contacts);
            }
        });

        loadFavorites();
        return v;
    }

    private void loadFavorites() {
        Executors.newSingleThreadExecutor().execute(() -> {
            List<Contact> favs = dao.getFavorites();
            if (getActivity() == null) return;
            getActivity().runOnUiThread(() -> {
                adapter.setItems(favs);
                View empty = getView() != null ? getView().findViewById(R.id.empty_container_favorites) : null;
                if (empty != null) {
                    empty.setVisibility((favs == null || favs.isEmpty()) ? View.VISIBLE : View.GONE);
                }
            });
        });
    }

    private void onItemClick(Contact c) {
        Intent i = new Intent(requireContext(), DetailContactActivity.class);
        i.putExtra(DetailContactActivity.EXTRA_ID, c.id);
        startActivity(i);
    }

    private void onItemDelete(Contact c) {
        Executors.newSingleThreadExecutor().execute(() -> {
            dao.delete(c);
            if (getActivity() == null) return;
            getActivity().runOnUiThread(() -> {
                Intent intent = new Intent(Constants.ACTION_CONTACTS_CHANGED);
                requireActivity().sendBroadcast(intent);
                Toast.makeText(requireContext(), "Deleted: " + c.name, Toast.LENGTH_SHORT).show();
                loadFavorites();
            });
        });
    }

    private void onFavoriteToggle(Contact c) {
        Executors.newSingleThreadExecutor().execute(() -> {
            boolean newFav = !c.isFavorite;
            long ts = System.currentTimeMillis();
            if (c.id > 0) {
                dao.updateFavorite(c.id, newFav ? 1 : 0, ts);
            } else {
                c.isFavorite = newFav;
                c.updatedAt = ts;
                dao.upsert(c);
            }
            if (getActivity() == null) return;
            getActivity().runOnUiThread(() -> {
                Intent intent = new Intent(Constants.ACTION_CONTACTS_CHANGED);
                requireActivity().sendBroadcast(intent);
                Toast.makeText(requireContext(), c.name + (newFav ? " added to favorites" : " removed from favorites"),
Toast.LENGTH_SHORT).show();
```

```
            loadFavorites();
        });
    });
  }
}
```

MainNavActivity.java

```
package com.example.comp1786_logbook_ex3_connectdatabaseapplication.ui;

import android.os.Bundle;
import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.Toolbar;
import androidx.fragment.app.Fragment;
import com.example.comp1786_logbook_ex3_connectdatabaseapplication.R;
import com.google.android.material.bottomnavigation.BottomNavigationView;

public class MainNavActivity extends AppCompatActivity {

    private Toolbar toolbar;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main_nav);

        toolbar = findViewById(R.id.toolbar_main_nav);
        setSupportActionBar(toolbar);

        BottomNavigationView bottomNav = findViewById(R.id.bottom_navigation);
        bottomNav.setOnItemSelectedListener(item -> {
            int id = item.getItemId();
            // Use if/else because R.id values may not be compile-time constants in some setups
            if (id == R.id.nav_contacts) {
                setTitle("Contacts");
                replaceFragment(new ContactsFragment());
                return true;
            } else if (id == R.id.nav_favorites) {
                setTitle("Favorites");
                replaceFragment(new FavoritesFragment());
                return true;
            } else if (id == R.id.nav_settings) {
                setTitle("Settings");
                replaceFragment(new SettingsFragment());
                return true;
            }
            return false;
        });

        // default
        if (savedInstanceState == null) {
            bottomNav.setSelectedItemId(R.id.nav_contacts);
        }
    }

    private void replaceFragment(Fragment f) {
        getSupportFragmentManager()
                .beginTransaction()
                .replace(R.id.fragment_container, f)
                .commitAllowingStateLoss();
    }
}
```

SettingsFragment.java

```
package com.example.comp1786_logbook_ex3_connectdatabaseapplication.ui;

import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
```

```java
import androidx.fragment.app.Fragment;

import com.example.comp1786_logbook_ex3_connectdatabaseapplication.R;

public class SettingsFragment extends Fragment {

    @Nullable
    @Override
    public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup container, @Nullable Bundle savedInstanceState) {
        return inflater.inflate(R.layout.fragment_settings, container, false);
    }
}
```

MainActivity.java

```java
package com.example.comp1786_logbook_ex3_connectdatabaseapplication;

import android.content.Intent;
import android.os.Bundle;

import androidx.appcompat.app.AppCompatActivity;

import com.example.comp1786_logbook_ex3_connectdatabaseapplication.activity.ContactsListActivity;
import com.example.comp1786_logbook_ex3_connectdatabaseapplication.ui.MainNavActivity;

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        // Launch the new navigation activity (contains bottom navigation and fragments)
        startActivity(new Intent(this, MainNavActivity.class));
        finish();
    }
}
```

activity_avatar_picker.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<androidx.coordinatorlayout.widget.CoordinatorLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:fitsSystemWindows="true"
    android:background="?android:attr/colorBackground">

    <com.google.android.material.appbar.AppBarLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:theme="@style/Theme.AppCompat.Light.DarkActionBar">

        <com.google.android.material.appbar.MaterialToolbar
            android:id="@+id/toolbar"
            android:layout_width="match_parent"
            android:layout_height="?attr/actionBarSize"
            app:title="@string/choose_avatar"
            app:titleTextColor="?android:attr/textColorPrimary" />

    </com.google.android.material.appbar.AppBarLayout>

    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/rvAvatars"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:padding="8dp"
        android:clipToPadding="false"
        app:layout_behavior="@string/appbar_scrolling_view_behavior" />

</androidx.coordinatorlayout.widget.CoordinatorLayout>
```

activity_contact_detail.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<androidx.coordinatorlayout.widget.CoordinatorLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:fitsSystemWindows="true"
    android:background="@color/background">

    <!-- App Bar with Gradient -->
    <com.google.android.material.appbar.AppBarLayout
        android:id="@+id/appBarLayout"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="@drawable/gradient_primary"
        android:elevation="4dp">

        <com.google.android.material.appbar.MaterialToolbar
            android:id="@+id/toolbar"
            android:layout_width="match_parent"
            android:layout_height="?attr/actionBarSize"
            android:background="@android:color/transparent"
            app:title="Contact Details"
            app:titleTextColor="@color/white"
            app:navigationIconTint="@color/white" />

    </com.google.android.material.appbar.AppBarLayout>

    <androidx.core.widget.NestedScrollView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        app:layout_behavior="@string/appbar_scrolling_view_behavior">

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="vertical"
            android:padding="20dp">

            <!-- Avatar Card -->
            <com.google.android.material.card.MaterialCardView
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_marginBottom="16dp"
                app:cardCornerRadius="16dp"
                app:cardElevation="3dp"
                app:strokeWidth="0dp">

                <LinearLayout
                    android:layout_width="match_parent"
                    android:layout_height="wrap_content"
                    android:orientation="vertical"
                    android:padding="24dp"
                    android:gravity="center">

                    <!-- Avatar with border -->
                    <com.google.android.material.card.MaterialCardView
                        android:layout_width="140dp"
                        android:layout_height="140dp"
                        app:cardCornerRadius="70dp"
                        app:cardElevation="6dp"
                        app:strokeWidth="4dp"
                        app:strokeColor="@color/primary">

                        <com.google.android.material.imageview.ShapeableImageView
                            android:id="@+id/imgAvatarDetail"
                            android:layout_width="match_parent"
                            android:layout_height="match_parent"
                            android:contentDescription="Contact avatar"
                            android:scaleType="centerCrop"
                            android:src="@drawable/ic_person_placeholder"
                            app:shapeAppearanceOverlay="@style/ShapeAppearance.Material3.Corner.Full" />

                    </com.google.android.material.card.MaterialCardView>
```

```
        <!-- Name -->
        <TextView
            android:id="@+id/tvName"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginTop="20dp"
            android:textSize="24sp"
            android:textColor="@color/text_primary"
            android:textStyle="bold"
            android:fontFamily="sans-serif-medium"
            android:gravity="center"
            tools:text="John Doe" />

        <!-- Last updated -->
        <TextView
            android:id="@+id/tvLastUpdated"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginTop="8dp"
            android:textSize="12sp"
            android:textColor="@color/text_hint"
            tools:text="Last updated: 01/01/2025 10:00" />

    </LinearLayout>

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

<!-- Contact Information Card -->
<com.google.android.material.card.MaterialCardView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginBottom="16dp"
    app:cardCornerRadius="16dp"
    app:cardElevation="3dp"
    app:strokeWidth="0dp">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        android:padding="20dp">

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Contact Information"
            android:textSize="16sp"
            android:textColor="@color/text_primary"
            android:textStyle="bold"
            android:fontFamily="sans-serif-medium"
            android:layout_marginBottom="16dp" />

        <!-- Date of Birth Row -->
        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="horizontal"
            android:gravity="center_vertical"
            android:paddingTop="12dp"
            android:paddingBottom="12dp"
            android:background="?attr/selectableItemBackground">

            <com.google.android.material.card.MaterialCardView
                android:layout_width="48dp"
                android:layout_height="48dp"
                app:cardCornerRadius="24dp"
                app:cardElevation="0dp"
                app:cardBackgroundColor="@color/primary_container">

                <ImageView
                    android:layout_width="24dp"
                    android:layout_height="24dp"
                    android:layout_gravity="center"
```

```
            android:src="@drawable/ic_cake"
            app:tint="@color/primary" />

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

        <LinearLayout
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:layout_marginStart="16dp"
            android:orientation="vertical">

            <TextView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="Date of Birth"
                android:textSize="12sp"
                android:textColor="@color/text_hint"
                android:fontFamily="sans-serif" />

            <TextView
                android:id="@+id/tvDob"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_marginTop="2dp"
                android:textSize="16sp"
                android:textColor="@color/text_primary"
                android:fontFamily="sans-serif-medium"
                tools:text="01/01/1990" />

        </LinearLayout>

    </LinearLayout>

    <!-- Divider -->
    <View
        android:layout_width="match_parent"
        android:layout_height="1dp"
        android:background="@color/divider"
        android:layout_marginTop="8dp"
        android:layout_marginBottom="8dp" />

    <!-- Email Row -->
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        android:gravity="center_vertical"
        android:paddingTop="12dp"
        android:paddingBottom="12dp"
        android:background="?attr/selectableItemBackground">

        <com.google.android.material.card.MaterialCardView
            android:layout_width="48dp"
            android:layout_height="48dp"
            app:cardCornerRadius="24dp"
            app:cardElevation="0dp"
            app:cardBackgroundColor="@color/secondary_container">

            <ImageView
                android:layout_width="24dp"
                android:layout_height="24dp"
                android:layout_gravity="center"
                android:src="@drawable/ic_email"
                app:tint="@color/secondary" />

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

        <LinearLayout
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:layout_marginStart="16dp"
            android:orientation="vertical">
```

```
                    <TextView
                        android:layout_width="wrap_content"
                        android:layout_height="wrap_content"
                        android:text="Email Address"
                        android:textSize="12sp"
                        android:textColor="@color/text_hint"
                        android:fontFamily="sans-serif" />

                    <TextView
                        android:id="@+id/tvEmail"
                        android:layout_width="wrap_content"
                        android:layout_height="wrap_content"
                        android:layout_marginTop="2dp"
                        android:textSize="16sp"
                        android:textColor="@color/text_primary"
                        android:fontFamily="sans-serif-medium"
                        tools:text="john.doe@example.com" />

                </LinearLayout>

            </LinearLayout>

        </LinearLayout>

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

    <!-- Edit Button -->
    <com.google.android.material.button.MaterialButton
        android:id="@+id/btnEdit"
        style="@style/PrimaryButton"
        android:layout_width="match_parent"
        android:layout_height="60dp"
        android:text="Edit Contact"
        android:textSize="16sp"
        app:icon="@drawable/ic_edit"
        app:iconGravity="textStart"
        app:iconPadding="12dp" />

  </LinearLayout>

 </androidx.core.widget.NestedScrollView>

</androidx.coordinatorlayout.widget.CoordinatorLayout>
```

activity_contacts_list.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.coordinatorlayout.widget.CoordinatorLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:fitsSystemWindows="true"
    android:background="@color/background">

    <!-- App Bar with Gradient -->
    <com.google.android.material.appbar.AppBarLayout
        android:id="@+id/appBarLayout"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="@drawable/gradient_primary"
        android:elevation="4dp"
        app:elevation="4dp">

        <com.google.android.material.appbar.MaterialToolbar
            android:id="@+id/toolbar"
            android:layout_width="match_parent"
            android:layout_height="?attr/actionBarSize"
            android:background="@android:color/transparent"
            app:title="My Contacts"
            app:titleTextColor="@color/white"
```

```
        app:navigationIconTint="@color/white" />

    </com.google.android.material.appbar.AppBarLayout>

    <!-- Main Content -->
    <androidx.core.widget.NestedScrollView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        app:layout_behavior="@string/appbar_scrolling_view_behavior">

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="vertical"
            android:padding="16dp">

            <!-- Header Card -->
            <com.google.android.material.card.MaterialCardView
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_marginBottom="16dp"
                app:cardCornerRadius="12dp"
                app:cardElevation="2dp"
                app:strokeWidth="0dp">

                <LinearLayout
                    android:layout_width="match_parent"
                    android:layout_height="wrap_content"
                    android:orientation="vertical"
                    android:padding="16dp">

                    <TextView
                        android:layout_width="wrap_content"
                        android:layout_height="wrap_content"
                        android:text="Contact List"
                        android:textColor="@color/text_primary"
                        android:textSize="20sp"
                        android:textStyle="bold"
                        android:fontFamily="sans-serif-medium" />

                    <TextView
                        android:id="@+id/tvContactCount"
                        android:layout_width="wrap_content"
                        android:layout_height="wrap_content"
                        android:layout_marginTop="4dp"
                        android:text="0 contacts"
                        android:textColor="@color/text_secondary"
                        android:textSize="14sp" />

                </LinearLayout>

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

            <!-- RecyclerView Container -->
            <androidx.recyclerview.widget.RecyclerView
                android:id="@+id/rvContacts"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:clipToPadding="false"
                android:nestedScrollingEnabled="false"
                android:overScrollMode="never"
                tools:listitem="@layout/item_contact" />

        </LinearLayout>

    </androidx.core.widget.NestedScrollView>

    <!-- Floating Action Button with Animation -->
    <com.google.android.material.floatingactionbutton.FloatingActionButton
        android:id="@+id/fabAdd"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="bottom|end"
        android:layout_margin="24dp"
```

```
        android:contentDescription="Add new contact"
        android:src="@drawable/ic_add"
        app:backgroundTint="@color/secondary"
        app:tint="@color/white"
        app:borderWidth="0dp"
        app:elevation="6dp"
        app:pressedTranslationZ="12dp"
        app:rippleColor="@color/overlay_light" />

</androidx.coordinatorlayout.widget.CoordinatorLayout>
```

activity_edit_contact.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:fitsSystemWindows="true"
    android:background="@color/background">

    <!-- Toolbar with Gradient -->
    <com.google.android.material.appbar.AppBarLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="@drawable/gradient_primary"
        android:elevation="4dp">

        <com.google.android.material.appbar.MaterialToolbar
            android:id="@+id/toolbar"
            android:layout_width="match_parent"
            android:layout_height="?attr/actionBarSize"
            android:background="@android:color/transparent"
            app:title="Edit Contact"
            app:titleTextColor="@color/white"
            app:navigationIconTint="@color/white" />

    </com.google.android.material.appbar.AppBarLayout>

    <ScrollView
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:fillViewport="true">

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="vertical"
            android:padding="20dp">

            <!-- Avatar Section Card -->
            <com.google.android.material.card.MaterialCardView
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_marginBottom="24dp"
                app:cardCornerRadius="16dp"
                app:cardElevation="3dp"
                app:strokeWidth="0dp">

                <LinearLayout
                    android:layout_width="match_parent"
                    android:layout_height="wrap_content"
                    android:orientation="vertical"
                    android:padding="24dp"
                    android:gravity="center">

                    <TextView
                        android:layout_width="wrap_content"
                        android:layout_height="wrap_content"
```

```
            android:text="Contact Avatar"
            android:textSize="16sp"
            android:textColor="@color/text_primary"
            android:textStyle="bold"
            android:fontFamily="sans-serif-medium"
            android:layout_marginBottom="16dp" />

        <!-- Avatar with border -->
        <com.google.android.material.card.MaterialCardView
            android:layout_width="120dp"
            android:layout_height="120dp"
            app:cardCornerRadius="60dp"
            app:cardElevation="4dp"
            app:strokeWidth="3dp"
            app:strokeColor="@color/primary">

            <com.google.android.material.imageview.ShapeableImageView
                android:id="@+id/imgPickAvatar"
                android:layout_width="match_parent"
                android:layout_height="match_parent"
                android:contentDescription="Contact avatar"
                android:scaleType="centerCrop"
                android:src="@drawable/ic_person_placeholder"
                app:shapeAppearanceOverlay="@style/ShapeAppearance.Material3.Corner.Full" />

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

        <!-- Choose Avatar Button -->
        <com.google.android.material.button.MaterialButton
            android:id="@+id/btnChooseAvatar"
            style="@style/SecondaryButton"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginTop="16dp"
            android:text="Choose Avatar"
            app:icon="@drawable/ic_add_a_photo"
            app:iconGravity="textStart"
            app:iconPadding="8dp" />

    </LinearLayout>

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

<!-- Contact Information Card -->
<com.google.android.material.card.MaterialCardView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    app:cardCornerRadius="16dp"
    app:cardElevation="3dp"
    app:strokeWidth="0dp">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        android:padding="20dp">

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Contact Information"
            android:textSize="16sp"
            android:textColor="@color/text_primary"
            android:textStyle="bold"
            android:fontFamily="sans-serif-medium"
            android:layout_marginBottom="16dp" />

        <!-- Name Field -->
        <com.google.android.material.textfield.TextInputLayout
            style="@style/CustomTextInputLayout"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginBottom="12dp"
            android:hint="Full Name"
```

```
            app:startIconDrawable="@drawable/ic_person"
            app:startIconTint="@color/primary"
            app:boxBackgroundMode="outline">

            <com.google.android.material.textfield.TextInputEditText
                android:id="@+id/etName"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:inputType="textPersonName"
                android:textSize="16sp" />

        </com.google.android.material.textfield.TextInputLayout>

        <!-- Date of Birth Field -->
        <com.google.android.material.textfield.TextInputLayout
            style="@style/CustomTextInputLayout"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginBottom="12dp"
            android:hint="Date of Birth"
            app:startIconDrawable="@drawable/ic_cake"
            app:startIconTint="@color/primary"
            app:boxBackgroundMode="outline">

            <com.google.android.material.textfield.TextInputEditText
                android:id="@+id/etDob"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:focusable="false"
                android:clickable="true"
                android:inputType="none"
                android:textSize="16sp" />

        </com.google.android.material.textfield.TextInputLayout>

        <!-- Email Field -->
        <com.google.android.material.textfield.TextInputLayout
            style="@style/CustomTextInputLayout"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:hint="Email Address"
            app:startIconDrawable="@drawable/ic_email"
            app:startIconTint="@color/primary"
            app:boxBackgroundMode="outline">

            <com.google.android.material.textfield.TextInputEditText
                android:id="@+id/etEmail"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:inputType="textEmailAddress"
                android:textSize="16sp" />

        </com.google.android.material.textfield.TextInputLayout>

    </LinearLayout>

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

<!-- Save Button -->
<com.google.android.material.button.MaterialButton
    android:id="@+id/btnSave"
    style="@style/PrimaryButton"
    android:layout_width="match_parent"
    android:layout_height="60dp"
    android:layout_marginTop="24dp"
    android:text="Save Contact"
    android:textSize="16sp"
    app:icon="@drawable/ic_person"
    app:iconGravity="textStart"
    app:iconPadding="12dp" />

</LinearLayout>

</ScrollView>
```

```
</LinearLayout>
```

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.coordinatorlayout.widget.CoordinatorLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity"
    android:background="?android:attr/windowBackground">

    <com.google.android.material.appbar.AppBarLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:theme="@style/ThemeOverlay.MaterialComponents.Dark.ActionBar">

        <com.google.android.material.appbar.MaterialToolbar
            android:id="@+id/toolbar"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:paddingStart="8dp"
            android:paddingEnd="8dp"
            app:title="Contacts"
            app:titleTextAppearance="@style/TextAppearance.MaterialComponents.Headline6" />

    </com.google.android.material.appbar.AppBarLayout>

    <androidx.core.widget.NestedScrollView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        app:layout_behavior="@string/appbar_scrolling_view_behavior">

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:orientation="vertical"
            android:gravity="center"
            android:padding="24dp">

            <com.google.android.material.card.MaterialCardView
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_marginTop="40dp"
                app:cardCornerRadius="12dp"
                app:cardElevation="6dp"
                android:padding="24dp">

                <LinearLayout
                    android:layout_width="match_parent"
                    android:layout_height="wrap_content"
                    android:orientation="vertical">

                    <TextView
                        android:id="@+id/tvAppTitle"
                        android:layout_width="wrap_content"
                        android:layout_height="wrap_content"
                        android:text="@string/app_title"
                        android:textAppearance="@style/TextAppearance.MaterialComponents.Headline5" />

                    <TextView
                        android:id="@+id/tvAppSubtitle"
                        android:layout_width="wrap_content"
                        android:layout_height="wrap_content"
                        android:layout_marginTop="8dp"
                        android:text="@string/app_subtitle"
                        android:textAppearance="@style/TextAppearance.MaterialComponents.Body1" />

                </LinearLayout>
```

```
        </com.google.android.material.card.MaterialCardView>

    </LinearLayout>

</androidx.core.widget.NestedScrollView>

</androidx.coordinatorlayout.widget.CoordinatorLayout>
```

activity_main_nav.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.coordinatorlayout.widget.CoordinatorLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:fitsSystemWindows="true"
    android:background="@color/background">

    <com.google.android.material.appbar.AppBarLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="@drawable/gradient_primary">

        <com.google.android.material.appbar.MaterialToolbar
            android:id="@+id/toolbar_main_nav"
            android:layout_width="match_parent"
            android:layout_height="?attr/actionBarSize"
            android:background="@android:color/transparent"
            app:title="Contacts"
            app:titleTextColor="@color/white" />

    </com.google.android.material.appbar.AppBarLayout>

    <FrameLayout
        android:id="@+id/fragment_container"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        app:layout_behavior="@string/appbar_scrolling_view_behavior"
        android:paddingBottom="72dp" />

    <com.google.android.material.bottomnavigation.BottomNavigationView
        android:id="@+id/bottom_navigation"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_gravity="bottom"
        app:menu="@menu/bottom_nav_menu"
        app:itemIconTint="@color/bottom_nav_tint"
        app:itemTextColor="@color/bottom_nav_tint"
        android:background="@color/white"
        app:labelVisibilityMode="labeled"
        android:elevation="8dp" />

</androidx.coordinatorlayout.widget.CoordinatorLayout>
```

fragment_contacts.xml

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="16dp">

    <androidx.core.widget.NestedScrollView
        android:id="@+id/nestedScroll"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        app:layout_behavior="@string/appbar_scrolling_view_behavior">
```

```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical">

    <androidx.cardview.widget.CardView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginBottom="16dp"
        app:cardUseCompatPadding="true"
        app:cardCornerRadius="12dp">

        <!-- Vertical container: search row + contact count -->
        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="vertical"
            android:padding="8dp">

            <!-- Custom search row: icon outside, EditText, sort button -->
            <LinearLayout
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:orientation="horizontal"
                android:gravity="center_vertical">

                <ImageView
                    android:id="@+id/imgSearchIcon"
                    android:layout_width="36dp"
                    android:layout_height="36dp"
                    android:src="@drawable/ic_search"
                    android:tint="@color/text_secondary"
                    android:contentDescription="Search icon"
                    android:layout_marginEnd="8dp" />

                <EditText
                    android:id="@+id/etSearchFrag"
                    android:layout_width="0dp"
                    android:layout_height="40dp"
                    android:layout_weight="1"
                    android:background="@android:color/transparent"
                    android:hint="Search contacts..."
                    android:padding="8dp"
                    android:imeOptions="actionSearch"
                    android:inputType="text"
                    android:singleLine="true" />

                <ImageButton
                    android:id="@+id/btnSortFrag"
                    android:layout_width="40dp"
                    android:layout_height="40dp"
                    android:background="?attr/selectableItemBackgroundBorderless"
                    android:src="@android:drawable/ic_menu_sort_alphabetically"
                    android:contentDescription="Sort"
                    android:tint="@color/primary"
                    android:layout_marginStart="8dp" />

            </LinearLayout>

            <!-- Contact count text (used by fragment to show number of contacts) -->
            <TextView
                android:id="@+id/tvContactCountFrag"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_marginTop="8dp"
                android:text="0 contacts"
                android:textColor="@color/text_secondary"
                android:textSize="14sp" />

        </LinearLayout>

    </androidx.cardview.widget.CardView>

    <androidx.recyclerview.widget.RecyclerView
```

```
            android:id="@+id/rvContactsFrag"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:nestedScrollingEnabled="false"
            tools:listitem="@layout/item_contact" />

        <include
            android:id="@+id/empty_container_contacts"
            layout="@layout/layout_empty_contacts"
            android:visibility="gone" />

    </LinearLayout>

</androidx.core.widget.NestedScrollView>

<com.google.android.material.floatingactionbutton.FloatingActionButton
    android:id="@+id/fabAddFrag"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="end|bottom"
    android:layout_margin="24dp"
    android:src="@drawable/ic_add"
    app:backgroundTint="@color/primary"
    app:tint="@color/white" />

</FrameLayout>
```

fragment_favourites.xml

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

    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/rvFavorites"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:padding="16dp"
        tools:listitem="@layout/item_contact" />

    <include
        android:id="@+id/empty_container_favorites"
        layout="@layout/layout_empty_favorites"
        android:visibility="gone" />

</FrameLayout>
```

fragment_favourites.xml

```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="16dp">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical">

        <androidx.cardview.widget.CardView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginBottom="16dp"
            app:cardCornerRadius="12dp">
```

```
        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:padding="12dp"
            android:orientation="horizontal"
            android:gravity="center_vertical">

            <View
                android:layout_width="48dp"
                android:layout_height="48dp"
                android:background="@drawable/bg_avatar_circle" />

            <LinearLayout
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:orientation="vertical"
                android:layout_marginStart="12dp">

                <TextView
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:text="User Name"
                    android:textSize="16sp"
                    android:textColor="@color/text_primary" />

                <TextView
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:text="user@example.com"
                    android:textColor="@color/text_secondary"
                    android:textSize="14sp" />

            </LinearLayout>

        </LinearLayout>

    </androidx.cardview.widget.CardView>

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Preferences"
        android:textColor="@color/text_secondary"
        android:paddingTop="8dp"
        android:paddingBottom="8dp" />

    <androidx.cardview.widget.CardView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        app:cardCornerRadius="12dp">

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="vertical">

            <LinearLayout
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:padding="12dp"
                android:gravity="center_vertical">

                <TextView
                    android:layout_width="0dp"
                    android:layout_height="wrap_content"
                    android:layout_weight="1"
                    android:text="Notifications"
                    android:textColor="@color/text_primary" />

                <Switch
                    android:id="@+id/switchNotifications"
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:checked="true" />
```

```
        </LinearLayout>

        <View android:layout_width="match_parent" android:layout_height="1dp" android:background="@color/divider" />

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:padding="12dp"
            android:gravity="center_vertical">

            <TextView
                android:layout_width="0dp"
                android:layout_height="wrap_content"
                android:layout_weight="1"
                android:text="Dark Mode"
                android:textColor="@color/text_primary" />

            <Switch
                android:id="@+id/switchDarkMode"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content" />

        </LinearLayout>

    </LinearLayout>

  </androidx.cardview.widget.CardView>

 </LinearLayout>

</ScrollView>
```

1. Overall architecture

The app is a contact management system with a Room-backed database, bottom navigation, and a multi-screen, Material Design UI.

- Entry points:
  - MainActivity – trivial launcher that immediately opens MainNavActivity and finishes
  - MainNavActivity – real shell Activity with BottomNavigationView and a fragment container
- Navigation shell: MainNavActivity hosts three tabs via fragments:
  - ContactsFragment – main contacts browser (search, sort, inline count, empty state)
  - FavoritesFragment – filtered list of favorites
  - SettingsFragment – placeholder for settings/preferences
- Data layer:
  - Room Contact entity (id, name, dob, email, avatarName, isFavorite, createdAt, updatedAt)
  - AppDatabase singleton exposing ContactDao
  - Multiple sorted queries + favorite queries + bulk operations (e.g. deleteByIds, updateFavorite)
- Detail / edit flows (outside fragments):
  - DetailContactActivity – full contact details + last updated timestamp
  - EditContactActivity – add/edit contact with validation and date picker
  - AvatarPickerActivity – avatar chooser grid + image upload from device
- Supporting components:
  - ContactsAdapter – main RecyclerView adapter (favorites, delete, multi-select, custom avatars)
  - AvatarGridAdapter – grid adapter for built-in avatars + upload tile
  - ResUtils – resource helper (drawable name lookup, avatar name list)

Overall, it’s a bottom-nav driven CRUD app with Room persistence, multiple views of the same data (all / favorites / detail), and a fairly rich avatar / image pipeline.

2. Navigation & screens

2.1 MainActivity – launcher

- Contains no UI logic; in onCreate it immediately starts MainNavActivity and calls finish().
- Keeps the “main” entry point simple while allowing a dedicated Activity for navigation.

2.2 MainNavActivity – bottom navigation shell

- Layout: activity_main_nav.xml with:
  - Gradient AppBarLayout + MaterialToolbar (toolbar_main_nav)
  - FrameLayout fragment_container for hosting fragments
  - BottomNavigationView (bottom_navigation) anchored at the bottom
- Bottom navigation items:
  - nav_contacts → ContactsFragment (title “Contacts”)
  - nav_favorites → FavoritesFragment (title “Favorites”)
  - nav_settings → SettingsFragment (title “Settings”)
- On first launch (savedInstanceState == null), selects nav_contacts by default.

Result: a single-Activity, multi-fragment navigation model with clear tab responsibilities.

2.3 Fragment-level screens

ContactsFragment

- Main list UI for all contacts with:
  - Search bar (etSearchFrag), search icon (imgSearchIcon), and sort button (btnSortFrag)
  - Contact count label (tvContactCountFrag)
  - Contact list RecyclerView (rvContactsFrag) using ContactsAdapter
  - Empty state include layout (layout_empty_contacts) shown when there are no contacts
- Handles inline search (text watcher), quick sort toggle (A–Z / Z–A), and add FAB (fabAddFrag) to open EditContactActivity.

FavoritesFragment

- Shows only contacts where isFavorite == true using dao.getFavorites().
- Uses the same ContactsAdapter to render list.
- Has empty layout (layout_empty_favorites) + “Browse contacts” button that programmatically switches bottom navigation back to nav_contacts.

SettingsFragment

- Currently a simple placeholder layout (fragment_settings.xml), ready for future preferences.

3. Data layer – Room model & DAO usage

3.1 Contact entity

Defined in Contact.java:

- Fields:
  - long id (auto-generated primary key)
  - String name
  - String dob (formatted as dd/MM/yyyy)
  - String email

    - String avatarName (either drawable name, or file:// / content:// URI for uploaded avatar)
    - boolean isFavorite
    - long createdAt
    - long updatedAt
- Constructors:
    - Main constructor sets createdAt and updatedAt to System.currentTimeMillis() and defaults isFavorite=false
    - Overload to explicitly set timestamps and favorite
    - No-arg constructor required by Room

This extended schema supports sorting by creation date, favorite filtering, and "last updated" display.

3.2 DAO operations (from call sites)

Although ContactDao isn't shown, its API is used across the codebase:

- Sorted reads:
    - getAllSortedByNameAsc() / getAllSortedByNameDesc()
    - getAllSortedByCreatedAtAsc() / getAllSortedByCreatedAtDesc()
- Filtered reads:
    - getFavorites() for favorites tab
    - getById(long id) for detail/edit
- Updates:
    - upsert(Contact c) – insert or update
    - updateFavorite(long id, int isFavorite, long updatedAt)
- Deletes:
    - delete(Contact c)
    - deleteByIds(List<Long> ids) (bulk delete)
    - clearAll() (wipe table)

All Room operations are executed on background threads using Executors.newSingleThreadExecutor(); UI is updated via runOnUiThread.

4. List adapters & UI helpers

4.1 ContactsAdapter – main list adapter

- Maintains:
    - List<Contact> items
    - boolean selectionMode + Set<Long> selectedIds for multi-select
- Constructor takes three callbacks:
    - OnItemClick – open detail
    - OnItemDelete – delete this contact
    - OnFavoriteToggle – toggle favorite state

Binding (VH.bind) responsibilities:

- Avatar loading:
    - If avatarName starts with content:// or file:// → attempts setImageURI, with SecurityException / generic exception handling and fallback to default avatar
    - Else: resolves drawable via ResUtils.nameToDrawable and sets resource
- Text:
    - Sets txtName and txtEmail
- Favorite star:
    - Shows filled or border star based on c.isFavorite

    - Click listener:
        - Debounce rapid taps (disables view for 300ms)
        - Calls favCb.onToggle(c); actual persistence handled in Activity/Fragment
- Selection Checkbox:
    - Only visible when selectionMode is true
    - Keeps cbSelect checked state in sync with selectedIds
    - Toggles membership in selectedIds and notifies adapter to refresh
- Click actions:
    - Card click triggers onItemClick
    - Delete button calls OnItemDelete from item tag

This adapter supports single item actions (open, favorite, delete) and bulk operations (multi-select + delete).

4.2 AvatarGridAdapter – avatar picker grid

- Data: List<String> names plus one synthetic “upload” tile.
- getItemCount() returns names.size() + 1.
- Binds:
    - For regular avatar name: resolves drawable via ResUtils.nameToDrawable and marks selected state
    - For "upload": uses ic_add_a_photo icon and uses callback with "upload" when clicked
- Exposes setSelectedName to visually highlight the current avatar when editing a contact.

4.3 ResUtils

- nameToDrawable(Context, String) – dynamic lookup of drawable by name.
- avatarNamesFromArray(Context, @ArrayRes int) – reads a typed array of avatar drawables and returns their resource entry names (used to populate the avatar grid).

5. UX flows: search, sort, filters, and counts

5.1 Search

ContactsFragment:

- Has EditText etSearchFrag with a TextWatcher.
- On every text change:
    - Updates currentSearchQuery
    - Calls loadContacts() to requery and filter in memory.
- Filter logic:
    - Case-insensitive substring match on name or email.

ContactsListActivity (older Activity-based version):

- Uses SearchView in toolbar (onCreateOptionsMenu).
- onQueryTextChange and onQueryTextSubmit both reload contacts with current query.

5.2 Sorting

Both ContactsFragment and ContactsListActivity use an internal SortMode enum:

- NAME_ASC, NAME_DESC, CREATED_DESC, CREATED_ASC.

ContactsListActivity exposes all four modes via a toolbar menu; ContactsFragment toggles between A–Z and Z–A via btnSortFrag (icon swap for visual feedback). Sorting is applied in DAO level (by calling the relevant getAllSorted… method) before in-memory filters.

5.3 Additional filters

- Favorites-only filter (showFavoritesOnly):
  - In ContactsListActivity: toggle via menu item action_filter_favorites
  - In FavoritesFragment: always “favorites only” by design (dao.getFavorites())
- Birth-month filter (filterBirthMonth, action_filter_birth_month):
  - Computes current month as MM and filters contacts whose dob.substring(3,5) matches.
  - Allows user to quickly see who has a birthday this month.

5.4 Counts and empty states

- Contacts count:
  - ContactsFragment: tvContactCountFrag shows "N contact(s)".
  - ContactsListActivity: tvContactCount similar logic.
- Empty layouts:
  - layout_empty_contacts and layout_empty_favorites included and toggled VISIBLE/GONE depending on whether lists are empty.
  - Empty contacts layout includes a button to add a new contact.

Together, this gives the app a rich, filterable address book experience beyond simple CRUD.

6. Detail, edit, and avatar flows

6.1 DetailContactActivity – view-only screen

- Receives EXTRA_ID and loads Contact from DB on a background thread.
- UI:
  - Gradient toolbar titled with contact.name
  - Circular avatar photo in a MaterialCardView
  - Name, DOB, email
  - “Last updated” label using DateFormat.getDateTimeInstance() on updatedAt or createdAt.
- Avatar loading handles both built-in and uploaded URIs with robust error handling.
- Listens for ACTION_CONTACTS_CHANGED via BroadcastReceiver:
  - On receive, reloads the contact from DB to stay in sync if edited elsewhere.

6.2 EditContactActivity – add/edit screen

- Mode:
  - If contact_id extra is present → edit existing contact
  - Otherwise → create new contact
- UI:
  - Gradient app bar, large avatar, “Choose Avatar” button
  - Three TextInputLayout fields: Name, Date of Birth, Email
  - “Save Contact” primary button
- Behavior:
  - etDob opens DatePickerDialog (DD/MM/YYYY format).
  - Save button:
    - Validates name and DOB non-empty
    - Validates email format if not empty
    - Fills or updates Contact object (including avatarName and updatedAt)
    - Calls dao.upsert on background thread

        - Sends ACTION_CONTACTS_CHANGED broadcast and finishes with RESULT_OK
- Avatar selection:
    - Opens AvatarPickerActivity (startActivityForResult).
    - On result:
        - Updates avatarName
        - Immediately updates avatar image in UI
        - If editing existing contact, can persist avatar change immediately and broadcast.

6.3 AvatarPickerActivity – avatar + upload

- Toolbar-screen with a RecyclerView grid (rvAvatars) of predefined avatars plus a final "upload" tile.
- Reads avatar names from R.array.contact_avatars via ResUtils.avatarNamesFromArray.
- When predefined avatar picked:
    - Returns its name in extra EXTRA_CHOSEN and finishes.
- When upload tile picked:
    - Handles runtime permissions:
        - Android 13+: READ_MEDIA_IMAGES
        - Marshmallow–12: READ_EXTERNAL_STORAGE
    - Uses ACTION_OPEN_DOCUMENT with CATEGORY_OPENABLE and FLAG_GRANT_READ_URI_PERMISSION | FLAG_GRANT_PERSISTABLE_URI_PERMISSION to select an image.
    - Copies selected image to internal storage under /files/avatars/ via copyImageToInternalStorage(Uri) and returns a file:// URI.
    - Handles SecurityException with extra logging and user-friendly error message.

This pipeline gives a robust avatar system that supports both themed built-in icons and user images in a permission-safe way.

7. Sync and multi-Activity coordination

The app uses a broadcast-based synchronization pattern:

- Constants.ACTION_CONTACTS_CHANGED is sent whenever contacts change:
    - In EditContactActivity after add/update
    - In delete operations (single delete, bulk delete, clear all)
    - After avatar changes
- Recipients:
    - ContactsListActivity (legacy list UI)
    - DetailContactActivity
    - Fragments (ContactsFragment, FavoritesFragment) indirectly via sendBroadcast in hosting Activity
- Each listener, on receiving the broadcast, reloads its data (loadContacts() / loadFavorites() / loadContact()).

This keeps all screens consistent without tightly coupling them, at the cost of a simple global broadcast approach.

8. Layouts & visual design

Key layout themes:

- Gradient headers and Material cards:

    - Many Activities (activity_contacts_list, activity_contact_detail, activity_edit_contact, activity_main_nav) use gradient app bars and MaterialCardView sections with rounded corners and elevation.
- Contact list items (item_contact, not shown but used by ContactsAdapter):
    - Avatar image, name, email, favorite icon, delete button, optional selection checkbox.
- Search card (fragment_contacts.xml):
    - Card containing search icon, inline EditText, sort button, and contact count.
- Empty states:
    - Dedicated layouts for empty contacts and empty favorites with explanatory text and CTA button.
- Settings layout (current placeholder) shows a user card and basic switches (notifications, dark mode) ready for future wiring.