Tên: Trần Đại Chí

MSSV: 18127070

1. **Các kỹ thuật áp dụng**:
   - Ở trong ứng dụng ghi chú, thay vì sử dụng SharedPreference cho phần lưu trữ các ghi chú thì chúng ta sẽ sử dụng Room database. Đây là một cơ sở dữ liệu ORM dựa trên SQLite database được google phát triển và cho phép dễ dàng thao tác các truy vấn SQLite trong Android chẳng hạn insert, delete, update
   - Trước tiên chúng ta sẽ cần thêm thư viện này vào phần build.gradle

```
implementation 'androidx.room:room-runtime:2.2.6'
annotationProcessor 'androidx.room:room-compiler:2.2.6'
```

   - Tiếp đến chúng ta sẽ tạo một Room Entities

```
//init database
@Entity(tableName = "notes")
public class Note {
    @PrimaryKey(autoGenerate = true) private int id; //key of each note
    @ColumnInfo(name = "text") private String noteText; //content of note
    @ColumnInfo(name = "date") private long noteDate; //latest date of note
    @ColumnInfo(name = "title") private String titleText; //title of note
    @ColumnInfo(name = "tag") private String tagText; //tag of note
    @Ignore private boolean checked = false; //checkbox to delete multiple notes, no need to store its state in DB

    public Note() {
        //default constructor
    }

    public Note(String noteText, long noteDate, String titleText, String tagText) {
        this.noteText = noteText;
        this.noteDate = noteDate;
        this.titleText = titleText;
        this.tagText = tagText;
    }

    public int getId() { return id; }

    public void setId(int id) { this.id = id; }

    public String getNoteText() { return noteText; }

    public void setNoteText(String noteText) { this.noteText = noteText; }

    public long getNoteDate() { return noteDate; }

    public void setNoteDate(long noteDate) { this.noteDate = noteDate; }

    public String getTitleText() { return titleText; }

    public void setTitleText(String titleText) { this.titleText = titleText; }
```

   - Trong đó tableName = “notes” là tên bảng cơ sở dữ liệu của chúng ta và chúng ta có thể đặt tên theo ý mình. Chúng ta cũng sẽ có ID là một khóa chính và các cột của nó bao gồm: nội dung ghi chú, ngày tạo/chỉnh sửa, tiêu đề, tag. Chúng ta cũng sẽ tạo constructor và getter/setter cho nó tương tự như một “Model”
   - Tiếp theo, chúng ta sẽ tạo một interface

```java
//create DAO interface
@Dao
public interface NotesDao {
    //insert note
    @Insert(onConflict = OnConflictStrategy.REPLACE)
    void insertNote(Note note);

    //delete multiple notes
    @Delete
    void deleteNote(Note... note);

    //edit exist note
    @Update
    void updateNote(Note note);

    //get all notes
    @Query("SELECT * FROM notes")
    List<Note> getNotes();

    //get note by its specific id
    @Query("SELECT * FROM notes WHERE id = :noteId")
    Note getNoteById(int noteId);

    //get multiple notes by their title
    @Query("SELECT * FROM notes WHERE title LIKE :titleSearch")
    List<Note> getNoteByTitle(String titleSearch);

    //get multiple notes by their tag
    @Query("SELECT * FROM notes WHERE tag LIKE :tagSearch")
    List<Note> getNoteByTag(String tagSearch);

    //get multiple notes by their string
    @Query("SELECT * FROM notes WHERE text = :stringSearch")
    List<Note> getNoteByString(String stringSearch);
```

- Chúng ta sẽ có các hàm thêm, xóa, chỉnh sửa note và các hàm phục vụ cho việc tìm kiếm các ghi chú dựa trên các câu truy vấn trong SQL
- Cuối cùng chúng ta sẽ khởi tạo database

```java
//create room database
@Database(entities = Note.class, version = 1, exportSchema = false)
public abstract class NotesDB extends RoomDatabase {
    public abstract NotesDao notesDao();

    public static NotesDB getInstance(Context context){
        return Room.databaseBuilder(context, NotesDB.class,  name: "mydb").allowMainThreadQueries().build();
    }
}
```

- Tiếp đến chúng ta sẽ tạo note adapter là nơi xử lý và gán dữ liệu cho recyclerview

```java
public class NotesAdapter extends RecyclerView.Adapter<NotesAdapter.ViewHolder> {
    private Context context;
    private ArrayList<Note> notes;
    private ItemClickListener itemClickListener;
    private boolean multiCheckMode = false;

    @SuppressLint("SimpleDateFormat")
    public String displayTime(long time) {
        DateFormat format = new SimpleDateFormat( pattern: "EEE, dd MMM yyyy 'at' hh:mm aaa");
        return format.format(new Date(time));
    }

    //Constructor
    public NotesAdapter(Context context, ArrayList<Note> notes) {
        this.context = context;
        this.notes = notes;
    }

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

    @Override
    public void onBindViewHolder(@NonNull NotesAdapter.ViewHolder holder, int position) {
        if(!notes.isEmpty()){
            holder.noteText.setText("Your note: " + notes.get(position).getNoteText());
            holder.noteDate.setText("Last modified date: " + displayTime(notes.get(position).getNoteDate()));
            holder.titleText.setText("Title: " + notes.get(position).getTitleText());
            holder.tagText.setText("Tag: " + notes.get(position).getTagText());
            holder.itemView.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View v) {
                    itemClickListener.StartNoteClick(notes.get(position));
                }
            });
            holder.itemView.setOnLongClickListener(new View.OnLongClickListener() {
```

- Ở đây chúng ta sẽ sử dụng listener cho việc xử lý các thao các khi click vào 1 một ghi chú đã có sẵn sẽ chuyển sang phần chỉnh sửa ghi chú đó và khi click 1 hồi lâu sẽ hiện ra chế độ multi check mode cho phép click vào nhiều ghi chú để xóa cùng một lúc
- Dưới đây là phần chỉnh sửa ghi chú của chúng ta

```java
public class EditNoteActivity extends AppCompatActivity {
    private EditText inputNote, inputTitle, inputTag;
    private NotesDao dao;
    private Note tmpNote;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_edit_note);
        Toolbar toolbar = findViewById(R.id.edit_note_activity_toolbar);
        setSupportActionBar(toolbar);
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);
        getSupportActionBar().setTitle("Create/Edit note");
        inputNote = findViewById(R.id.input_note);
        inputTitle = findViewById(R.id.input_title);
        inputTag = findViewById(R.id.input_tag);
        dao = NotesDB.getInstance(this).notesDao();
        //show title, tag and content of exist note
        if(getIntent().getExtras() != null) {
            tmpNote = dao.getNoteById(getIntent().getExtras().getInt( key: "noteID",  defaultValue: 0));
            inputNote.setText(tmpNote.getNoteText());
            inputTitle.setText(tmpNote.getTitleText());
            inputTag.setText(tmpNote.getTagText());
        }
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.edit_note_menu, menu);
        return super.onCreateOptionsMenu(menu);
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        int id = item.getItemId();
        if (id == R.id.save_note) {
```

- Chúng ta sẽ lấy ID của ghi chú đó và kiểm tra nếu ID đó đã từng tồn tại trong cơ sở dữ liệu khi chúng ta đã thêm ghi chú đó trước đó thì toàn bộ ghi chú của chúng ta sẽ giữ như cũ và chúng ta chỉ cần thêm vào nội dung mới vào trong ghi chú đó và ngược lại
- Dưới đây là phần chúng ta sẽ tìm kiếm các ghi chú dựa trên tiêu đề và chuỗi của nó

```
private void filterByTitle(String title){
    ArrayList<Note> filteredListTitle = new ArrayList<>();
    for(Note note: notes){
        if(note.getTitleText().toLowerCase().contains(title.toLowerCase())){
            filteredListTitle.add(note);
        }
    }
    adapter = new NotesAdapter( context: this, filteredListTitle);
    adapter.setItemClickListener(this);
    recyclerView.setAdapter(adapter);
}

private void filterByString(String text){
    ArrayList<Note> filteredListText = new ArrayList<>();
    for(Note note: notes){
        if(note.getNoteText().toLowerCase().contains(text.toLowerCase())){
            filteredListText.add(note);
        }
    }
    adapter = new NotesAdapter( context: this, filteredListText);
    adapter.setItemClickListener(this);
    recyclerView.setAdapter(adapter);
}
```

- Chúng ta sẽ lấy ra tiêu đề của toàn bộ các ghi chú của chúng ta đem so sánh với tiêu đề chúng ta đã nhập vào thanh tìm kiếm và nếu giống nhau chúng ta sẽ thêm vào một arraylist chứa các ghi chú đó và hiện lên màn hình các ghi chú tìm kiếm được gần giống nhất với tiêu đề đã nhập của chúng ta và nó cũng tương tự như khi chúng ta tìm kiếm với chuỗi của các ghi chú
- Dưới đây là phần chúng ta hiển thị toàn bộ các ghi chú lên màn hình

```
private void loadNotes() {
    //get all notes from room database and add data to array list
    notes = new ArrayList<>();
    List<Note> list = dao.getNotes();
    notes.addAll(list);
    adapter = new NotesAdapter( context: this, notes);
    adapter.setItemClickListener(this);
    recyclerView.setAdapter(adapter);
    displayEmptyNote();
}
```

- Chúng ta sẽ lấy toàn bộ các ghi chú dựa trên truy vấn SQL như trong phần interface đã đề cập ở trên và hiển thị nó lên recyclerview và thiết lập listener cho nó để chúng ta có thể click và long click được trên các ghi chú đó
- Dưới đây là phần xử lý khi chúng ta click vào một ghi chú:

```
@Override
public void StartNoteClick(Note note) {
    Intent intent = new Intent( packageContext: MainActivity.this, EditNoteActivity.class);
    intent.putExtra( name: "noteID", note.getId()); //get id of note when click to edit
    startActivity(intent);
}
```

- Chúng ta sẽ lấy ID và vào phần chỉnh sửa ghi chú mà chúng ta đã click

```
@Override
public void StartNoteLongClick(Note note) {
    fab.setVisibility(View.GONE);
    note.setChecked(true); //set current note selected as checked
    adapter.setMultiCheckMode(true);
    adapter.setItemClickListener(new ItemClickListener() {
        @Override
        public void StartNoteClick(Note note) {
            note.setChecked(!note.isChecked());
            List<Note> selectedNote = adapter.getCheckedNotes();
            if(selectedNote.size() == 0) {
                act.finish();
            }
            adapter.notifyDataSetChanged();
        }

        @Override
        public void StartNoteLongClick(Note note) {

        }
    });
    mActionModeCallback = new ActionMode.Callback() {
        @Override
        public boolean onActionItemClicked(ActionMode mode, MenuItem item) {
            switch(item.getItemId()){
                case R.id.action_delete_notes:
                    DeleteMultipleNotes();
                    mode.finish();
                    return true;
                default: return false;
            }
        }

        @Override
        public boolean onCreateActionMode(ActionMode actionMode, Menu menu) {
            actionMode.getMenuInflater().inflate(R.menu.main_action_mode, menu);
            act = actionMode;
            return true;
        }
```

- Một cách tương tự, khi chúng ta thực hiện long click vào 1 ghi chú, 1 thanh action mode sẽ hiện ra phía trên với icon thùng rác để chúng ta xóa các ghi chú và có các checkbox để chúng ta tích vào để chọn các ghi chú mà chúng ta muốn xóa
- Cuối cùng là phần xóa các ghi chú của chúng ta:

```
private void DeleteMultipleNotes() {
    List<Note> selectedNote = adapter.getCheckedNotes();
    if(selectedNote.size() != 0) {
        AlertDialog.Builder builder = new AlertDialog.Builder( context: this);
        builder.setTitle("Delete note");
        if(selectedNote.size() == 1){
            builder.setMessage("Do you want to delete this note?");
        }
        else{
            builder.setMessage("Do you want to delete " + selectedNote.size() + " notes");
        }
        builder.setPositiveButton( text: "Yes", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                for(Note note: selectedNote){
                    dao.deleteNote(note);
                }
                loadNotes(); //update list note after deleting
                Toast.makeText( context: MainActivity.this, text: "Delete note successfully!!!", Toast.LENGTH_SHORT).show();
            }
        });
        builder.setNegativeButton( text: "No", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) { dialog.cancel(); }
        });
        AlertDialog alertDialog = builder.create();
        alertDialog.show();
    }
}
```

- Chúng ta sẽ dùng một AlertDialog để hiển thị khi xóa các ghi chú. Nếu chúng ta click vào “No” (negative button) thì sẽ hủy việc xóa ghi chú và ngược lại khi chúng ta click vào “Yes” (positive button) thì sẽ xóa các ghi chú đó ra khỏi cơ sở dữ liệu và update lại màn hình hiển thị danh sách các ghi chú sau khi đã xóa thành công

2. **Link youtube demo các tính năng**: https://www.youtube.com/watch?v=Jj2CcD0HVjU&ab_channel=chitran
3. **Điểm tự đánh giá**: 10/10

References:

[1]: https://viblo.asia/p/su-dung-room-database-trong-android-naQZRD0q5vx

[2]: https://developer.android.com/training/data-storage/room