ROYAL UNIVERSITY OF BHUTAN
COLLEGE OF SCIENCE AND TECHNOLOGY
PHUENTSHOLING: BHUTAN

PLAGIARISM DECLARATION FORM

Student Name: Divyash Chhetri

Student No: 02200174

Module No and Title of the module: CTE306 - Mobile Application Development
Assignment no and Title of the Assignment: Lab Wok 08

Section H2 of the Royal University of Bhutan's *Wheel of Academic Law* provides the following definition of academic dishonesty:

"Academic dishonesty may be defined as any attempt by a student to gain an unfair advantage in any assessment. It may be demonstrated by one of the following:

Collusion: the representation of a piece of unauthorized group work as the work of a single candidate.

Commissioning: submitting an assignment done by another person as the student's own work.

Duplication: the inclusion in coursework of material identical or substantially similar to material which has already been submitted for any other assessment within the University.

False declaration: making a false declaration in order to receive special consideration by an Examination Board or to obtain extensions to deadlines or exemption from work.

Falsification of data: presentation of data in laboratory reports, projects, etc., based on work purported to have been carried out by the student, which have been invented, altered or copied by the student.

Plagiarism: the unacknowledged use of another's work as if it were one's own.

Examples are:

- verbatim copying of another's work without acknowledgement
- paraphrasing of another's work by simply changing a few words or altering the order of presentation, without acknowledgement
- ideas or intellectual data in any form presented as one's own without acknowledging the source(s)
- making significant use of unattributed digital images such as graphs, tables, photographs, etc. taken from test books, articles, films, plays, handouts, internet, or any other source, whether published or unpublished
- submission of a piece of work which has previously been assessed for a different award or module or at a different institution as if it were new work
- use of any material without prior permission of copyright from appropriate authority or owner of the materials used"

Student Declaration

I confirm that I have read and understood the above definitions of academic dishonesty. I declare that I have not committed any academic dishonesty when completing the attached piece of work.

Signature of Student: Divyash Chhetri

Date: 20 Sept 2022

Lab 08

CTE306 – Mobile Application Development

Date – 20th September 2022

Divyash Chhetri
02200174
BE 3 IT

Module Tutor: Mr. Pema Galey

Department of Information Technology
College of Science and Technology

Aim

To execute the data model referring to the codelabs.

Task

Execute the Data Model by referring to the codelabs related practicals in the given link below:

10.1 Part A: Room, LiveData, and ViewModel
10.1 Part B: Room, LiveData, and ViewModel

For Concept reference, visit following link:
https://google-developer-training.github.io/android-developer-fundamentals-course-concepts-v2/unit-4-saving-user-data/lesson-10-storing-data-with-room/10-1-c-room-livedata-viewmodel/10-1-c-room-livedata-viewmodel.html#relatedpractical

Theory

The Android Architecture Components provide libraries for common activities like as lifecycle management and data persistence to make it easier to implement the recommended architecture. With minimal boilerplate code, strong, tested, and maintainable app architecture is made possible with the aid of architecture components. A UI controller, a ViewModel that serves LiveData, a Repository, and a Room database make up the architecture. A data access object provides access to the SQLite database that supports the Room database (DAO).

The UI controller, which is in responsible for preparing data for the UI, is provided with a ViewModel helper class by Android Architecture Components. In order to ensure that the data held by ViewModel objects is instantly accessible to the activity or fragment instance, they are automatically kept throughout configuration changes.

LiveData is a class that holds observable data. LiveData is lifecycle-aware, which means it respects the life cycle of other app components like activities, fragments, or services. Because of this awareness, LiveData only updates app component observers that are currently in the active lifecycle state.

Program Code

AndroidManifest.xml

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

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity
            android:name=".MainActivity"
            android:exported="true"
            android:theme="@style/AppTheme.NoActionBar">
            <intent-filter>
               <action android:name="android.intent.action.MAIN" />
               <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name=".NewWordActivity"></activity>
    </application>

</manifest>
```

MainActivity.java

```
package com.example.datamodel;

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

import androidx.annotation.Nullable;
import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.Toolbar;
import androidx.lifecycle.Observer;
import androidx.lifecycle.ViewModelProviders;
import androidx.recyclerview.widget.ItemTouchHelper;
import androidx.recyclerview.widget.LinearLayoutManager;
```

```
import androidx.recyclerview.widget.RecyclerView;

import
com.google.android.material.floatingactionbutton.FloatingActionButton;

import java.util.List;

public class MainActivity extends AppCompatActivity {

    public static final int NEW_WORD_ACTIVITY_REQUEST_CODE = 1;
    private WordViewModel mWordViewModel;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Toolbar toolbar = findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);

        RecyclerView recyclerView = findViewById(R.id.recyclerview);
        final WordListAdapter adapter = new WordListAdapter(this);
        recyclerView.setAdapter(adapter);
        recyclerView.setLayoutManager(new LinearLayoutManager(this));

        mWordViewModel =
ViewModelProviders.of(this).get(WordViewModel.class);
        mWordViewModel.getAllWords().observe(this, new
Observer<List<Word>>() {
            @Override
            public void onChanged(@Nullable final List<Word> words) {
                adapter.setWords(words);
            }
        });

        FloatingActionButton fab = findViewById(R.id.fab);
        fab.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent intent = new Intent(MainActivity.this,
NewWordActivity.class);
                startActivityForResult(intent,
NEW_WORD_ACTIVITY_REQUEST_CODE);
            }
        });

        ItemTouchHelper helper = new ItemTouchHelper(
```

```
                new ItemTouchHelper.SimpleCallback(0,
                        ItemTouchHelper.LEFT | ItemTouchHelper.RIGHT) {
                    @Override
                    public boolean onMove(RecyclerView recyclerView,
                                          RecyclerView.ViewHolder
viewHolder,
                                          RecyclerView.ViewHolder target) {
                        return false;
                    }

                    @Override
                    public void onSwiped(RecyclerView.ViewHolder
viewHolder, int direction) {
                        int position = viewHolder.getAdapterPosition();
                        Word myWord = adapter.getWordAtPosition(position);
                        Toast.makeText(MainActivity.this,
                                getString(R.string.delete_word_preamble) +
" " +
                                        myWord.getWord(),
Toast.LENGTH_LONG).show();

                        mWordViewModel.deleteWord(myWord);
                    }
                });
        helper.attachToRecyclerView(recyclerView);
    }

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

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

        if (id == R.id.clear_data) {
            Toast.makeText(this, R.string.clear_data_toast_text,
Toast.LENGTH_LONG).show();

            mWordViewModel.deleteAll();
            return true;
        }
        return super.onOptionsItemSelected(item);
    }
```

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

        if (requestCode == NEW_WORD_ACTIVITY_REQUEST_CODE && resultCode ==
RESULT_OK) {
            Word word = new
Word(data.getStringExtra(NewWordActivity.EXTRA_REPLY));
            // Save the data
            mWordViewModel.insert(word);
        } else {
            Toast.makeText(
                    this, R.string.empty_not_saved,
Toast.LENGTH_LONG).show();
        }
    }
}
```

NewWordActivity.java

```
package com.example.datamodel;

import android.content.Intent;
import android.os.Bundle;
import android.text.TextUtils;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;

import androidx.appcompat.app.AppCompatActivity;

public class NewWordActivity extends AppCompatActivity {

    public static final String EXTRA_REPLY = "com.example.datamodel.REPLY";

    private EditText mEditWordView;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_new_word);

        mEditWordView = findViewById(R.id.edit_word);
        final Button button = findViewById(R.id.button_save);
        button.setOnClickListener(new View.OnClickListener() {
```

```
            public void onClick(View view) {
                Intent replyIntent = new Intent();
                if (TextUtils.isEmpty(mEditWordView.getText())) {
                    setResult(RESULT_CANCELED, replyIntent);
                } else {
                    String word = mEditWordView.getText().toString();
                    replyIntent.putExtra(EXTRA_REPLY, word);
                    setResult(RESULT_OK, replyIntent);
                }
                finish();
            }
        });
    }
}
```

Word.java

```
package com.example.datamodel;

import androidx.annotation.NonNull;
import androidx.room.ColumnInfo;
import androidx.room.Entity;
import androidx.room.PrimaryKey;

@Entity(tableName = "word_table")
public class Word {

    @PrimaryKey
    @NonNull
    @ColumnInfo(name = "word")
    private String mWord;

    public Word(@NonNull String word) {
        this.mWord = word;
    }

    public String getWord() {
        return this.mWord;
    }
}
```

WordDao.java

```java
package com.example.datamodel;

import androidx.lifecycle.LiveData;
import androidx.room.Dao;
import androidx.room.Delete;
import androidx.room.Insert;
import androidx.room.OnConflictStrategy;
import androidx.room.Query;

import java.util.List;

@Dao
public interface WordDao {

    @Insert(onConflict = OnConflictStrategy.IGNORE)
    void insert(Word word);

    @Query("DELETE FROM word_table")
    void deleteAll();

    @Delete
    void deleteWord(Word word);

    @Query("SELECT * from word_table LIMIT 1")
    Word[] getAnyWord();

    @Query("SELECT * from word_table ORDER BY word ASC")
    LiveData<List<Word>> getAllWords();

}
```

WordListAdapter.java

```java
package com.example.datamodel;

import android.content.Context;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;

import androidx.recyclerview.widget.RecyclerView;

import java.util.List;

public class WordListAdapter extends
RecyclerView.Adapter<WordListAdapter.WordViewHolder> {
```

```
    private final LayoutInflater mInflater;
    private List<Word> mWords; // Cached copy of words

    WordListAdapter(Context context) {
        mInflater = LayoutInflater.from(context);
    }

    @Override
    public WordViewHolder onCreateViewHolder(ViewGroup parent, int
viewType) {
        View itemView = mInflater.inflate(R.layout.recyclerview_item,
parent, false);
        return new WordViewHolder(itemView);
    }

    @Override
    public void onBindViewHolder(WordViewHolder holder, int position) {
        if (mWords != null) {
            Word current = mWords.get(position);
            holder.wordItemView.setText(current.getWord());
        } else {
            holder.wordItemView.setText(R.string.no_word);
        }
    }

    void setWords(List<Word> words) {
        mWords = words;
        notifyDataSetChanged();
    }

    @Override
    public int getItemCount() {
        if (mWords != null)
            return mWords.size();
        else return 0;
    }

    public Word getWordAtPosition(int position) {
        return mWords.get(position);
    }

    class WordViewHolder extends RecyclerView.ViewHolder {
        private final TextView wordItemView;

        private WordViewHolder(View itemView) {
```

```
            super(itemView);
            wordItemView = itemView.findViewById(R.id.textView);
        }
    }
}
```

WordRepository.java

```
package com.example.datamodel;

import android.app.Application;
import android.os.AsyncTask;

import androidx.lifecycle.LiveData;

import java.util.List;

public class WordRepository {

    private WordDao mWordDao;
    private LiveData<List<Word>> mAllWords;

    WordRepository(Application application) {
        WordRoomDatabase db = WordRoomDatabase.getDatabase(application);
        mWordDao = db.wordDao();
        mAllWords = mWordDao.getAllWords();
    }

    LiveData<List<Word>> getAllWords() {
        return mAllWords;
    }

    public void insert(Word word) {
        new insertAsyncTask(mWordDao).execute(word);
    }

    public void deleteAll() {
        new deleteAllWordsAsyncTask(mWordDao).execute();
    }

    public void deleteWord(Word word) {
        new deleteWordAsyncTask(mWordDao).execute(word);
    }
```

```
    private static class insertAsyncTask extends AsyncTask<Word, Void,
Void> {

        private WordDao mAsyncTaskDao;

        insertAsyncTask(WordDao dao) {
            mAsyncTaskDao = dao;
        }

        @Override
        protected Void doInBackground(final Word... params) {
            mAsyncTaskDao.insert(params[0]);
            return null;
        }
    }

    private static class deleteAllWordsAsyncTask extends AsyncTask<Void,
Void, Void> {
        private WordDao mAsyncTaskDao;

        deleteAllWordsAsyncTask(WordDao dao) {
            mAsyncTaskDao = dao;
        }

        @Override
        protected Void doInBackground(Void... voids) {
            mAsyncTaskDao.deleteAll();
            return null;
        }
    }

    private static class deleteWordAsyncTask extends AsyncTask<Word, Void,
Void> {
        private WordDao mAsyncTaskDao;

        deleteWordAsyncTask(WordDao dao) {
            mAsyncTaskDao = dao;
        }
        @Override
        protected Void doInBackground(final Word... params) {
            mAsyncTaskDao.deleteWord(params[0]);
            return null;
        }
    }
}
```

WordRoomDatabase.java

```
package com.example.datamodel;

import android.content.Context;
import android.os.AsyncTask;

import androidx.annotation.NonNull;
import androidx.room.Database;
import androidx.room.Room;
import androidx.room.RoomDatabase;
import androidx.sqlite.db.SupportSQLiteDatabase;

@Database(entities = {Word.class}, version = 1, exportSchema = false)
public abstract class WordRoomDatabase extends RoomDatabase {

    public abstract WordDao wordDao();

    private static WordRoomDatabase INSTANCE;

    public static WordRoomDatabase getDatabase(final Context context) {
        if (INSTANCE == null) {
            synchronized (WordRoomDatabase.class) {
                if (INSTANCE == null) {
                    INSTANCE =
Room.databaseBuilder(context.getApplicationContext(),
                                    WordRoomDatabase.class,
"word_database")
                            .fallbackToDestructiveMigration()
                            .addCallback(sRoomDatabaseCallback)
                            .build();

                }
            }
        }
        return INSTANCE;
    }

    private static RoomDatabase.Callback sRoomDatabaseCallback =
            new RoomDatabase.Callback() {

                @Override
                public void onOpen(@NonNull SupportSQLiteDatabase db) {
                    super.onOpen(db);
                    new PopulateDbAsync(INSTANCE).execute();
                }
            };
```

```
    private static class PopulateDbAsync extends AsyncTask<Void, Void,
Void> {

        private final WordDao mDao;

        private static String[] words = {"dolphin", "crocodile", "cobra",
"elephant", "goldfish",
                "tiger", "snake"};

        PopulateDbAsync(WordRoomDatabase db) {
            mDao = db.wordDao();
        }

        @Override
        protected Void doInBackground(final Void... params) {
            if (mDao.getAnyWord().length < 1) {
                for (int i = 0; i <= words.length - 1; i++) {
                    Word word = new Word(words[i]);
                    mDao.insert(word);
                }
            }
            return null;
        }
    }
}
```

WordViewModel.java

```
package com.example.datamodel;

import android.app.Application;

import androidx.lifecycle.AndroidViewModel;
import androidx.lifecycle.LiveData;

import java.util.List;

public class WordViewModel extends AndroidViewModel {

    private WordRepository mRepository;

    private LiveData<List<Word>> mAllWords;

    public WordViewModel(Application application) {
        super(application);
        mRepository = new WordRepository(application);
```

```
        mAllWords = mRepository.getAllWords();
    }

    LiveData<List<Word>> getAllWords() {
        return mAllWords;
    }

    public void insert(Word word) {
        mRepository.insert(word);
    }

    public void deleteAll() {
        mRepository.deleteAll();
    }

    public void deleteWord(Word word) {
        mRepository.deleteWord(word);
    }
}
```

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:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <com.google.android.material.appbar.AppBarLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:theme="@style/AppTheme.AppBarOverlay">

        <androidx.appcompat.widget.Toolbar
            android:id="@+id/toolbar"
            android:layout_width="match_parent"
            android:layout_height="?attr/actionBarSize"
            android:background="?attr/colorPrimary"
            app:popupTheme="@style/AppTheme.PopupOverlay" />

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

    <include layout="@layout/content_main" />
```

```
    <com.google.android.material.floatingactionbutton.FloatingActionButton
        android:id="@+id/fab"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="bottom|end"
        android:layout_margin="@dimen/fab_margin"
        android:src="@drawable/ic_add_black_24dp"
        app:backgroundTint="@color/colorAccent" />


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

activity_new_word.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/colorScreenBackground"
    android:orientation="vertical"
    android:padding="24dp">

    <EditText
        android:id="@+id/edit_word"
        style="@style/text_view_style"
        android:hint="@string/hint_word"
        android:inputType="textAutoComplete" />

    <Button
        android:id="@+id/button_save"
        style="@style/button_style"
        android:text="@string/button_save" />
</LinearLayout>
```

content_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    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:background="@color/colorScreenBackground"
    app:layout_behavior="@string/appbar_scrolling_view_behavior"
    tools:context=".MainActivity"
    tools:showIn="@layout/activity_main">
```

```
    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/recyclerview"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_margin="16dp"
        android:layout_marginStart="16dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        tools:listitem="@layout/recyclerview_item" />

</androidx.constraintlayout.widget.ConstraintLayout >
```

recyclerview_item.xml

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

    <TextView
        android:id="@+id/textView"
        style="@style/text_view_style"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:elevation="8dp"
        tools:text="placeholder text" />
</LinearLayout>
```

main_menu.xml

```
<?xml version="1.0" encoding="utf-8"?>
<menu 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"
    tools:context="com.android.example.roomwordssample.MainActivity">
    <item
        android:id="@+id/clear_data"
        android:orderInCategory="100"
        android:title="@string/clear_all_data"
        app:showAsAction="never" />
</menu>
```

strings.xml

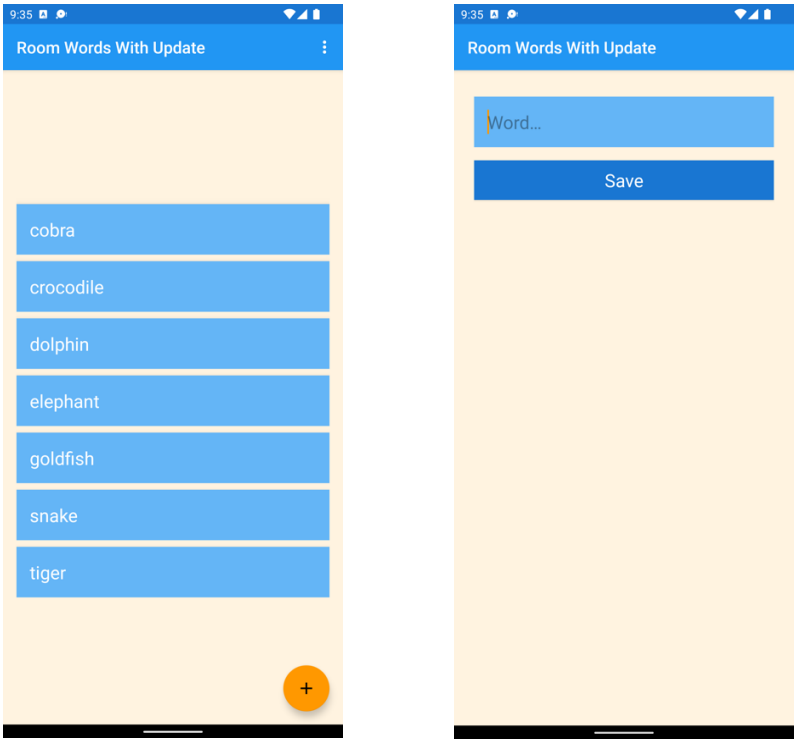
```
<resources>
    <string name="app_name">Room Words With Update</string>
    <string name="action_settings">Settings</string>
    <string name="hint_word">Word…</string>
    <string name="button_save">Save</string>
    <string name="empty_not_saved">Word not saved because it is
empty.</string>
    <string name="clear_all_data">Clear all data</string>
    <string name="delete_word_preamble">Deleting </string>
    <string name="clear_data_toast_text">Clear the data now!</string>
    <string name="no_word">No word</string>
    <string name="unable_to_update">Unable to update the word</string>
</resources>
```

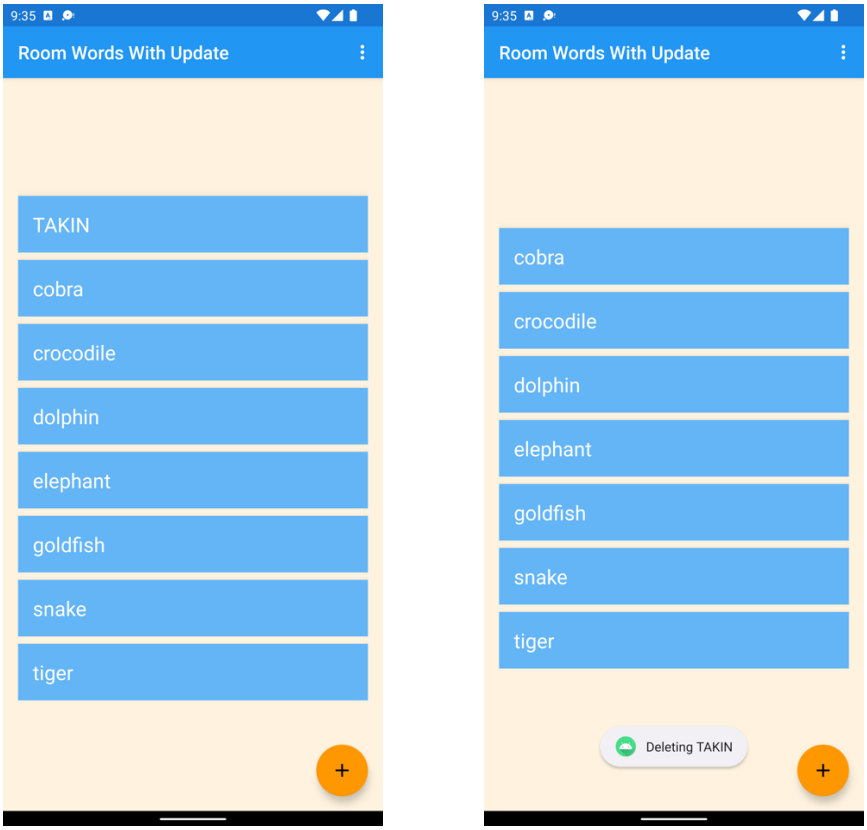
dimens.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <dimen name="fab_margin">16dp</dimen>
    <dimen name="small_padding">6dp</dimen>
    <dimen name="big_padding">16dp</dimen>
</resources>
```

Output

The app supports, update and delete operations.





Conclusion

By the completion of this lab I learnt to implement the concepts of ViewModel and LiveData using Room. I also became familiar with Android Architecture Components. I even learnt to populate the database with data and delete it by swiping the recycle view items to left. It was convenient to use the previously learnt concept of recycle view and menu. The above app has functionalities like updating the app to keep data when the app closes, all words by selecting an Options menu item, delete a specific word by swiping an item in the list and then adding items to list using FAB.

References

Android. (n.d.). *Android Studio Docs*. Retrieved 08 07, 2022, from Developers Android: https://developer.android.com/docs

Google Developer Training. (n.d.). *Room, LiveData, and ViewModel*. Retrieved Sept 20, 2022, from Google Developer Training: https://google-developer-training.github.io/android-developer-fundamentals-course-concepts-v2/unit-4-saving-user-data/lesson-10-storing-data-with-room/10-1-c-room-livedata-viewmodel/10-1-c-room-livedata-viewmodel.html#relatedpractical