The Best Text Editor Ever

Henry Post

Lara Teagues

ActivityDownloadFile.kt

```
package cooldomainname.com.thebesttexteditorever.activities

import android.content.Intent
import android.os.Bundle
import android.support.v7.app.AppCompatActivity
import android.view.inputmethod.EditorInfo
import android.widget.EditText
import android.widget.ProgressBar
import android.widget.TextView
import cooldomainname.com.thebesttexteditorever.BetterFile
import cooldomainname.com.thebesttexteditorever.Library.*
import cooldomainname.com.thebesttexteditorever.R
import java.io.File
import java.io.FileNotFoundException
import java.io.FileOutputStream
import java.io.InputStream
import java.net.MalformedURLException
import java.net.URL

class ActivityDownloadFile : AppCompatActivity() {

    companion object {
        // Key for storing our uri once it's downloaded.
        const val BUNDLE_KEY_FILE_URI = "downloaded file"
    }

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_download_file)

        // Set up UI elements
        val textViewTitle = findViewById<TextView>(R.id.textViewTitle)
        val editTextFilename = findViewById<EditText>(R.id.editTextFilename)
        val progressBar = findViewById<ProgressBar>(R.id.progressBar)
        val textViewStatus = findViewById<TextView>(R.id.textViewStatus)

        // Focus the filename EditText
        editTextFilename.requestFocus()

        // Show the keyboard
        showKeyboard(this@ActivityDownloadFile)

        // When they do something with the keyboard,
        editTextFilename.setOnEditorActionListener(TextView.OnEditorActionListener {
view, actionId, event ->
            //If it's 'done' or 'next',
            if (actionId == EditorInfo.IME_ACTION_DONE || actionId ==
EditorInfo.IME_ACTION_NEXT) {

                val url = view.text.toString()
                val filesafeurl = fileSafeString(url)


                val tempDir = File(applicationContext.filesDir.toString(), "/temp")

                var outputFile = BetterFile(tempDir, filesafeurl).createIfNotExists()

                // If, for some reason, it's a directory, make it not be one.
                if (outputFile.isDirectory) {
                    outputFile.deleteIfExists()
                    outputFile =
BetterFile(outputFile.absolutePath).createIfNotExists()
                }

                //TODO make progress bar work. Put incremental stuff inside of this
thread.
                Thread(Runnable {

                    var input: InputStream? = null
                    val output = FileOutputStream(outputFile)

                    try {
                        input = URL(url).openStream() //TODO error trapping for 403,
no internet, etc.


                    } catch (e: FileNotFoundException) { //If 404,
                        runOnUiThread {
                            toastLong("HTTP Error 404, not found.",
applicationContext)
                        }
                        logException(e)
                        return@Runnable

                    } catch (e: MalformedURLException) { // Malformed URL
                        runOnUiThread {
                            toastLong(
                                "URL is malformed. Check your colons and slashes.",
                                applicationContext
                            )
                        }
                        logException(e)
                        return@Runnable

                    } catch (e: Exception) { // General exception
                        runOnUiThread {
                            toastLong(
                                """Other exception occurred:
                                |${e.localizedMessage}""".trimMargin(),
applicationContext
                            )
                        }
                        logException(e)
                        return@Runnable
                    }

                    assert(input != null)

                    textViewStatus.append("Opened URL $url.\n")

                    //FIXME no progress
                    // Copy input to output.
                    val bytesCopied = input.use {
                        output.use {
                            input.copyTo(output)
                        }
                    }

                    // We're done!
                    progressBar.progress = 100

                    textViewStatus.append("$bytesCopied bytes copied.\n")

                    // Return back to the parent activity our downloaded file's URL.
```

```
                        val intent = Intent()
                        intent.putExtra(BUNDLE_KEY_FILE_URI, outputFile.absolutePath)
                        setResult(RESULT_OK, intent)

                        finish()

                    }).start()


                    hideKeyboard(this@ActivityDownloadFile)

                    return@OnEditorActionListener true
                }

                false
            })
        }
    }
```

ActivityEditText.java

```
package cooldomainname.com.thebesttexteditorever.activities;


import android.content.Intent;
import android.os.Bundle;
import android.support.annotation.Nullable;
import android.support.v4.app.FragmentManager;
import android.support.v7.app.AppCompatActivity;
import android.text.TextWatcher;
import android.util.Log;
import android.view.View;
import android.widget.*;
import cooldomainname.com.thebesttexteditorever.BetterFile;
import cooldomainname.com.thebesttexteditorever.BetterSpinner;
import cooldomainname.com.thebesttexteditorever.R;
import cooldomainname.com.thebesttexteditorever.TextBuffer;
import
cooldomainname.com.thebesttexteditorever.dialogfragments.OpenFileDialogFragment;
import
cooldomainname.com.thebesttexteditorever.dialogfragments.OpenFileDialogFragment.OpenFi
leDialogListener;
import
cooldomainname.com.thebesttexteditorever.dialogfragments.SaveFileDialogFragment;
import
cooldomainname.com.thebesttexteditorever.dialogfragments.SaveFileDialogFragment.SaveFi
leDialogListener;
import cooldomainname.com.thebesttexteditorever.syntaxhighlighting.LanguageIdentifier;
import
cooldomainname.com.thebesttexteditorever.syntaxhighlighting.TextWatchers.TextWatcherJa
va;
import
cooldomainname.com.thebesttexteditorever.syntaxhighlighting.TextWatchers.TextWatcherPo
rkdown;

import java.io.*;
import java.util.Arrays;
import java.util.HashMap;
import java.util.List;

import static cooldomainname.com.thebesttexteditorever.Library.toastLong;
import static
```

```
cooldomainname.com.thebesttexteditorever.activities.ActivityDownloadFile.BUNDLE_KEY_FI
LE_URI;
import static
cooldomainname.com.thebesttexteditorever.activities.ActivityEditText.RequestCode.OPEN_
URL;

public class ActivityEditText extends AppCompatActivity implements
OpenFileDialogListener, SaveFileDialogListener {

    /***
     * The TextBuffer that stores our text.
     */
    private TextBuffer textBuffer;

    /**
     * Actions you can perform on a file.
     */
    private List<String> listFileActions = Arrays.asList("", "open", "open url",
"save", "rename", "run tests");

    /**
     * Actions you can perform on text.
     */
    private List<String> listTextActions = Arrays.asList("", "cut", "select", "move");

    private EditText editText;

    /***
     * A listing of TextWatchers that can handle specific extensions.
     *
     * It maps file extensions to classes that can highlight EditText elements.
     */
    private HashMap<String, Class<? extends TextWatcher>> extensionTextWatcherMap =
            new HashMap<String, Class<? extends TextWatcher>>() {{
                put("porkdown", TextWatcherPorkdown.class);
                put("java", TextWatcherJava.class);
            }};

    /**
     * We want to open the 'save file' dialog.
     */
    private void openSaveFileDialog() {
        FragmentManager fm = getSupportFragmentManager();

        SaveFileDialogFragment saveFileDialogFragment = new SaveFileDialogFragment();

        Bundle bundle = new Bundle();
        bundle.putString("title", "Save File");

        saveFileDialogFragment.setArguments(bundle);

        saveFileDialogFragment.show(fm, "title");

    }

    /**
     * We want to open the 'open url' dialog.
     */
    private void openOpenUrlDialog() {
        startActivityForResult(new Intent(this, ActivityDownloadFile.class),
OPEN_URL.ordinal());
    }
    /**
     * We want to open the 'open file' dialog.
     */
    private void openOpenFileDialog() {
        FragmentManager fm = getSupportFragmentManager();

        OpenFileDialogFragment openFileDialogFragment = new OpenFileDialogFragment();

        Bundle bundle = new Bundle();
        bundle.putString("title", "Open File");

        openFileDialogFragment.setArguments(bundle);

        openFileDialogFragment.show(fm, "title");
    }

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

        if (requestCode == RequestCode.OPEN_URL.ordinal()) //If we returned because we
wanted to open a URL,

            if (resultCode == RESULT_OK) { //If we're good

                if (data.hasExtra(BUNDLE_KEY_FILE_URI)) { //If we have
                    String filepath = data.getStringExtra(BUNDLE_KEY_FILE_URI);

                    openFile(new BetterFile(filepath));
                }
            }
    }

    /**
     * When this Activity is created.
     */
    @Override
    protected void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_text_edit);

        // If we have a TextBuffer left over, use it.
        if (savedInstanceState != null) {

            if (savedInstanceState.get("TextBuffer") != null) {
                this.textBuffer = (TextBuffer) savedInstanceState.get("TextBuffer");
            }

        }

        // Our EditText element that has a bunch of text.
        editText = findViewById(R.id.editTextEditor);

        // Spinner for actions that can be performed on a file.
        final BetterSpinner spinnerFileActions =
findViewById(R.id.spinnerFileActions);
        ArrayAdapter<String> adapterFileActions = new ArrayAdapter<>(this,
android.R.layout.simple_spinner_item, listFileActions);

adapterFileActions.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_it
em);
        spinnerFileActions.setAdapter(adapterFileActions);

        spinnerFileActions.setOnItemSelectedListener(new
AdapterView.OnItemSelectedListener() {

            @Override
            public void onItemSelected(AdapterView<?> parent, View view, int position,
long id) {

                //Prevents initialization from triggering this {@link
ArrayAdapter.onItemSelected} event.
                if (spinnerFileActions.initialized) {

                    String selection = ((String) parent.getItemAtPosition(position));

                    Toast.makeText(getApplicationContext(), String.format("Your file
action selection is '%s'.", selection), Toast.LENGTH_SHORT).show();

                    switch (selection.toLowerCase()) {

                        case "open": {
                            openOpenFileDialog();
                            break;
                        }

                        case "open url": {
                            openOpenUrlDialog();
                            break;
                        }

                        case "save": {
                            openSaveFileDialog();
                            break;
                        }

                        case "rename": {
                            break;
                        }

                        case "run tests": { // TODO make MORE TESTS!!!
                            testEditTextFileSaving();
                            toastLong("Tests worked. Woo!", getApplicationContext());
                        }

                        default: {
                            break;
                        }
                    }
                } else {
                    spinnerFileActions.initialized = true;
                }
            }

            @Override
            public void onNothingSelected(AdapterView<?> parent) {
                // TODO Auto-generated method stub
            }
        });

        // Spinner for actions that can be performed on text.
        final BetterSpinner spinnerTextActions =
findViewById(R.id.spinnerTextActions);
        ArrayAdapter<String> adapterTextActions = new ArrayAdapter<>(this,
android.R.layout.simple_spinner_item, listTextActions);

adapterTextActions.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_it
em);
        spinnerTextActions.setAdapter(adapterTextActions);

        /* Actions that can be performed on text. */
        spinnerTextActions.setOnItemSelectedListener(new
AdapterView.OnItemSelectedListener() {

            @Override
            public void onItemSelected(AdapterView<?> parent, View view, int position,
long id) {

                //Prevents initialization from triggering this {@link
ArrayAdapter.onItemSelected} event.
                if (spinnerTextActions.initialized) {
                    String selection = ((String) parent.getItemAtPosition(position));

                    Toast.makeText(getApplicationContext(), String.format("Your text
editing selection is '%s'.", selection), Toast.LENGTH_SHORT).show();

                    switch (selection.toLowerCase()) {
                        case "cut": {
                            break;
                        }
                        case "select": {
                            break;
                        }
                        case "move": {
                            break;
                        }
                        default: {
                            break;
                        }
                    }
                } else {
                    spinnerTextActions.initialized = true;
                }
            }

            @Override
            public void onNothingSelected(AdapterView<?> parent) {
                // TODO Auto-generated method stub
            }
        });

    }

    /***
     * When the user wishes to save the file.
     * @param inputText The text that was input.
     */
    // 3. This method is invoked in the activity when the listener is triggered
    // Access the data result passed to the activity here
    @Override
    public void onFinishEditSaveDialog(String inputText) {
//        toastLong(String.format("We should save the file to '%s'.", inputText),
getApplicationContext());

        File file = new File(getApplicationContext().getFilesDir(), inputText);

        TextBuffer textBuffer = TextBuffer.fromEditText(editText);

        try {
            textBuffer.saveTo(file);
            String message = String.format("Saved file to '%s.'",
file.getAbsolutePath());

            toastLong(message, this.getApplicationContext());
            Log.i(this.getClass().getSimpleName(), message);
        } catch (IOException e) {
            e.printStackTrace();
            toastLong("Couldn't save file.", getApplicationContext());
        }
    }

    /**
     * Setup syntax highlighting for an {@link EditText} given a filename and a {@link
HashMap} containing
     * filename <-> {@link HashMap} mapping.
     *
     * @param extension A file extension.
     * @param editText  The {@link EditText} to apply the {@link TextWatcher} to.
     * @param hashMap   The mapping of file extensions to {@link TextWatcher}s.
     */
    public void setupSyntaxHighlighter(String extension, EditText editText,
HashMap<String, Class<? extends TextWatcher>> hashMap) {
        try {
            editText.addTextChangedListener(hashMap.get(extension).newInstance());
        } catch (IllegalAccessException e) {
            e.printStackTrace();
        } catch (InstantiationException e) {
            e.printStackTrace();
        }
    }

    public void openFile(BetterFile file) {
        try {
            // Get a TextBuffer from our file.
            textBuffer = TextBuffer.fromFile(file);

            // Empty out our EditText.
            editText.setText("");

            // Set up syntax highlighting, if we can.
            String extension = file.extension(); //TODO make Language object with
Set<String> knownExtensions, etc.

            // If we know how to highlight this file,
            if (extensionTextWatcherMap.containsKey(extension)) {
                // Try to do it!
                toastLong(String.format("'%s' language detected from extension '%s'!",
extension, extension), getApplicationContext());
                setupSyntaxHighlighter(extension, editText, extensionTextWatcherMap);
            } else { //We don't know by its extension, so...

                //Try to identify it by its contents!
                String detectedExtension =
LanguageIdentifier.Companion.identify(textBuffer.toString());

                // If it worked,
                if (detectedExtension != null &&
                        extensionTextWatcherMap.containsKey(detectedExtension)) {

                    toastLong(String.format("'%s' language detected from language
auto-detection!", extension), getApplicationContext());
```

```
                    // Setup syntax highlighting that way!
                    setupSyntaxHighlighter(extension, editText,
extensionTextWatcherMap);
                }
            }

            // Populate our EditText with its contents.
            textBuffer.populateEditText(editText);

        } catch (IOException e) {
            e.printStackTrace();
            toastLong("Couldn't open file.", getApplicationContext());
        }

    }

    /**
     * The user wishes to open a file.
     */
    @Override
    public void onFinishEditOpenDialog(String inputText) {

        File dir = getApplicationContext().getFilesDir();

        BetterFile file = new BetterFile(dir, inputText);

        // File doesn't exist.
        if (!file.exists()) {
            toastLong(String.format(
                    "File called '%s' does not exist at: \n" +
                            "'%s'.",
                    inputText, dir), getApplicationContext());

            return;
        }

        openFile(file);
    }

    /***
     * Test that we can save a file to disk given an {@link EditText} object with
text.
     */
    public void testEditTextFileSaving() {

        EditText editText = new EditText(getApplicationContext());
        CharSequence coolDelim = "WEGHDFGBDHGDFGHBDG";

        CharSequence coolText = ("I am a really cool snippet of text.\n" +
                "Also, I'm separated by something rad.\n" +
                "Don't you still like edge cases?\n" +
                "\n" +
                "I do.\n" +
                "F").replace("\n", coolDelim);

        editText.append(coolText);

        // Hide it.
        editText.setVisibility(View.GONE);

        // Make a file.
        BetterFile file = new BetterFile(getApplicationContext().getFilesDir(),
"pls_delet_kthxbai.txt").deleteIfExists();
```

```
        // Get the text from the EditText.
        TextBuffer textBuffer = TextBuffer.fromEditText(editText, coolDelim);

        // Try to save the file.
        try {
            textBuffer.saveTo(file);
        } catch (IOException e) {
            e.printStackTrace();

            // We should be able to save the file.
            throw new AssertionError("Something happened while saving file >:(");
        }

        if ((!file.exists())) throw new AssertionError(); // File should exist after
saving it.

        // Next, to read it back.
        BufferedReader reader;

        try {
            reader = new BufferedReader(new FileReader(file));

            // First char should be the first character of our text snippet.
            if ((reader.read() != coolText.charAt(0))) throw new AssertionError();

            CharSequence theRest = reader.readLine();

            // As we have no newlines (We replaced it with gobbledygook),
            // we should have consumed the rest of the input by asking for one line.
            if (reader.ready()) throw new AssertionError();

            // It also should NOT contain newlines, as we replaced 'em all.
            if (((String) theRest).contains("\n")) throw new AssertionError();

            // How many times the delimiter occurs.
            int delimOccurrences = ((String) theRest).split((String)
coolDelim).length;
            int preferredOccurrences = ((String) coolText).split((String)
coolDelim).length;

            // We should have as many as our original string does.
            if (delimOccurrences != preferredOccurrences)
                throw new AssertionError(String.format("%d != %d", delimOccurrences,
preferredOccurrences));

            // Last character should be the end of our text snippet.
            if (theRest.charAt(theRest.length() - 1) !=
coolText.charAt(coolText.length() - 1))
                throw new AssertionError();

            // Second-to-last character should be the end of our delimiter sequence.
            if (theRest.charAt(theRest.length() - 2) !=
coolDelim.charAt(coolDelim.length() - 1))
                throw new AssertionError();

        } catch (FileNotFoundException e) {
            e.printStackTrace();

            // We should be able to open the file.
            throw new AssertionError("Can't open da file?!");
        } catch (IOException e) {
            e.printStackTrace();
            throw new AssertionError("YA PIPE IS BROKEN!!!");
        }

    }


    enum RequestCode {
        OPEN_URL,
    }
}
```

ExampleInstrumentTest.kt

```
package cooldomainname.com.thebesttexteditorever

import android.support.test.InstrumentationRegistry
import android.support.test.runner.AndroidJUnit4

import org.junit.Test
import org.junit.runner.RunWith

import org.junit.Assert.*

/**
 * Instrumented test, which will execute on an Android device.
 *
 * See [testing documentation](http://d.android.com/tools/testing).
 */
@RunWith(AndroidJUnit4::class)
class ExampleInstrumentedTest {
    @Test
    fun useAppContext() {
        // Context of the app under test.
        val appContext = InstrumentationRegistry.getTargetContext()
        assertEquals("cooldomainname.com.thebesttexteditorever",
appContext.packageName)
    }
}
```

SimpleTest.java

```
package cooldomainname.com.thebesttexteditorever;

import org.junit.Test;

import java.util.Iterator;

import static org.junit.Assert.assertEquals;
import static org.mockito.Mockito.mock;
import static org.mockito.Mockito.when;


public class SimpleTests {

    @Test
    public void iterator_will_return_hello_world() {
        //arrange
        Iterator i = mock(Iterator.class);
        when(i.next()).thenReturn("Hello").thenReturn("World");
        //act
        String result = i.next() + " " + i.next();
        //assert
        assertEquals("Hello World", result);
    }

}
```

activity_download_file.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"
              android:layout_width="match_parent"
              android:layout_height="match_parent" android:orientation="vertical">

    <TextView
            android:text="Download from URL"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:id="@+id/textViewTitle" android:layout_marginTop="8dp"
            android:layout_marginLeft="8dp" android:layout_marginStart="8dp"
            android:layout_marginEnd="8dp"
            android:layout_marginRight="8dp" android:gravity="center"/>
    <EditText
            android:hint="URL to download."
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:ems="10"
            android:id="@+id/editTextFilename"
            android:layout_margin="8dp"
            android:inputType="text"
            android:text="https://raw.githubusercontent.com/HenryFBP/the-best-text-
editor-ever/master/the-best-text-editor-
ever/app/src/main/java/cooldomainname/com/thebesttexteditorever/SyntaxHighlighting/SEU
tils.java"
            android:layout_marginTop="8dp"/>
    <ProgressBar
            style="?android:attr/progressBarStyleHorizontal"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:id="@+id/progressBar"
            app:layout_constraintEnd_toEndOf="parent"
            android:layout_margin="8dp"
            app:layout_constraintBottom_toBottomOf="parent"
            android:max="100"
            android:progress="0"/>
    <EditText
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:inputType="textMultiLine"
            android:clickable="false"
            android:focusable="false"
            android:scrollbars="vertical"
            android:background="@android:color/transparent"
            android:hint="Status messages."
            android:ems="10"
            android:id="@+id/textViewStatus"
            android:gravity="top"
            android:enabled="false"
            android:layout_margin="8dp"
            android:editable="false"/>
</LinearLayout>
```

activity_text_edit.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
        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">

    <LinearLayout
            android:orientation="vertical"
            android:layout_width="match_parent"
            android:layout_height="match_parent">
        <android.support.constraint.ConstraintLayout
                android:layout_width="match_parent"
                android:layout_height="wrap_content">

            <cooldomainname.com.thebesttexteditorever.BetterSpinner
                    android:id="@+id/spinnerTextActions"
                    android:layout_width="0dp"
                    android:layout_height="wrap_content"
                    android:layout_marginStart="8dp"
                    android:layout_marginLeft="8dp"
                    android:layout_marginTop="8dp"
                    android:layout_marginBottom="8dp"
                    android:background="@android:drawable/btn_dropdown"
                    android:spinnerMode="dropdown"
                    app:layout_constraintBottom_toBottomOf="parent"
                    app:layout_constraintStart_toStartOf="parent"
                    app:layout_constraintTop_toTopOf="parent"
                    app:layout_constraintVertical_bias="0.0"
                    app:layout_constraintEnd_toStartOf="@+id/spinnerFileActions"
                    android:layout_marginEnd="8dp" android:layout_marginRight="8dp"/>
            <cooldomainname.com.thebesttexteditorever.BetterSpinner
                    android:id="@+id/spinnerFileActions"
                    android:layout_width="0dp"
                    android:layout_height="wrap_content"
                    android:layout_marginTop="8dp"
                    android:layout_marginEnd="8dp"
                    android:layout_marginRight="8dp"
                    android:layout_marginBottom="8dp"
                    android:background="@android:drawable/btn_dropdown"
                    android:spinnerMode="dropdown"
                    app:layout_constraintBottom_toBottomOf="parent"
                    app:layout_constraintEnd_toEndOf="parent"
                    app:layout_constraintTop_toTopOf="parent"
                    app:layout_constraintVertical_bias="0.0"
android:gravity="center"/>
        </android.support.constraint.ConstraintLayout>
        <ScrollView
                android:layout_width="match_parent"
                android:layout_height="match_parent"
app:layout_constraintTop_toBottomOf="@+id/spinnerTextActions"
                app:layout_constraintBottom_toBottomOf="parent"
app:layout_constraintStart_toStartOf="parent"
                app:layout_constraintEnd_toEndOf="parent"
app:layout_constraintHorizontal_bias="0.0"
                app:layout_constraintVertical_bias="0.0">
            <EditText
                    android:layout_width="match_parent"
                    android:layout_height="wrap_content"
                    android:inputType="textMultiLine"
                    android:ems="10"
                    android:id="@+id/editTextEditor"
android:text="@string/defaultEditorText"
                    android:importantForAutofill="no"
                    android:hint="@string/editTextHint"/>
        </ScrollView>
    </LinearLayout>

</android.support.constraint.ConstraintLayout>
```

fragment_dialog_file.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:orientation="vertical">

    <TextView
            android:text="File name"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:id="@+id/textViewTitle" android:layout_marginTop="8dp"
            android:layout_marginLeft="8dp" android:layout_marginStart="8dp"
            android:layout_marginEnd="8dp"
            android:layout_marginRight="8dp" android:gravity="center"/>
    <EditText
            android:layout_width="match_parent"
            android:hint="File name"
            android:layout_height="wrap_content"
            android:ems="10"
            android:id="@+id/editTextFilename"
            android:layout_marginLeft="8dp"
            android:layout_marginStart="8dp" android:layout_marginEnd="8dp"
android:layout_marginRight="8dp"
            android:inputType="text" android:text="temp_file.txt"
            android:layout_marginTop="8dp"/>
    <ListView
            android:layout_width="match_parent"
            android:layout_height="wrap_content" android:id="@+id/listViewFiles"
            android:layout_marginLeft="8dp"
            android:layout_marginStart="8dp" android:layout_marginEnd="8dp"
android:layout_marginRight="8dp"
            android:layout_marginTop="8dp"
            android:scrollbars="vertical"
            android:layout_marginBottom="8dp">
    </ListView>
</LinearLayout>
```

colors.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="colorPrimary">#008577</color>
    <color name="colorPrimaryDark">#00574B</color>
    <color name="colorAccent">#D81B60</color>
</resources>
```

strings.xml

```
<resources>
    <string name="app_name">the-best-text-editor-ever</string>
    <string name="defaultEditorText">Hello! I\'m text!</string>
    <string name="editTextHint">The text editor\'s content.</string>
</resources>
```

styles.xml

```
<resources>

    <!-- Base application theme. -->
    <style name="AppTheme" parent="Base.Theme.AppCompat.Light.DarkActionBar">
        <!-- Customize your theme here. -->
        <item name="colorPrimary">@color/colorPrimary</item>
        <item name="colorPrimaryDark">@color/colorPrimaryDark</item>
        <item name="colorAccent">@color/colorAccent</item>
    </style>

</resources>
```

AndroidManifest.xml

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

    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>
    <uses-permission android:name="android.permission.SEND_SMS"/>
    <uses-permission android:name="android.permission.INTERNET"/>

    <application
            android:allowBackup="true"
            android:fullBackupContent="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="cooldomainname.com.thebesttexteditorever.activities.ActivityEditText">
            <intent-filter>
                <action android:name="android.intent.action.MAIN"/>
                <action android:name="android.intent.action.VIEW"/>
                <category android:name="android.intent.category.LAUNCHER"/>
                <category android:name="android.intent.category.DEFAULT"/>
            </intent-filter>
        </activity>
        <activity
android:name="cooldomainname.com.thebesttexteditorever.activities.ActivityDownloadFile
">
        </activity>
    </application>

</manifest>
```