ICP11

Keenan Flynn – [kpfxn8@umsystem.edu](mailto:kpfxn8@umsystem.edu) – https://github.com/kfly2fly/Web-Mobile-Spring-2022

Jasmine Naraine- [jnytc@umstyem.edu](mailto:jnytc@umstyem.edu) -<https://github.com/JNaraine/490-Mobile>

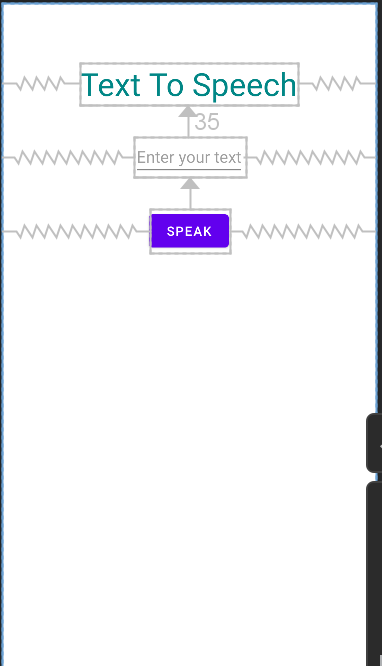
In this ICP, we followed along in class to create a simple app that allows the user to use the Text to Speech function of Android, which comes natively to the OS. The Text-To-Speech function sends text to the audio output. There is only one TTS engine per device so we need to make sure that permissions are being asked for and given up correctly.

The User Interface consists of 3 views. There is a TextView that is static and acts as the title of the app. There is an EditText view. This view allows the user to enter a string which will be then spoken through the speaker. Finally there is a Button View that when pressed, passes the text in the EditText view to the Java code which then speaks the entered text.

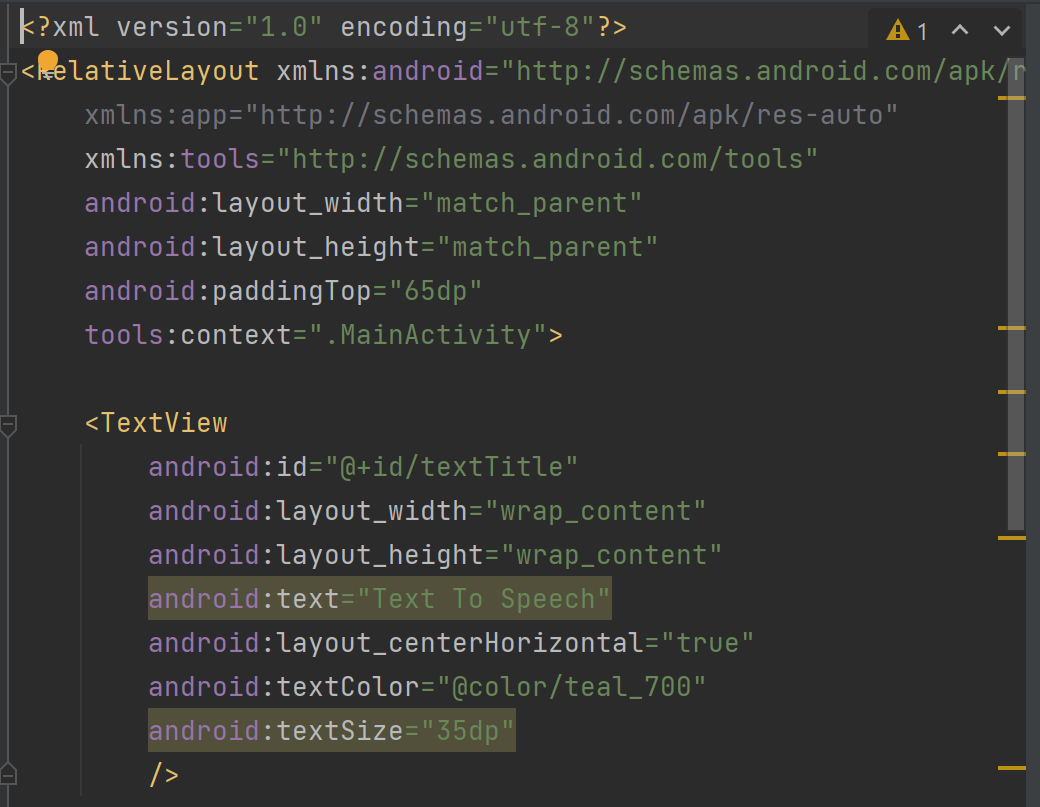
The Java backend is where the Text-to-Speech instance is created. When the button is pressed, the app creates a new TTS object using the MainActivity context. This new object has a method called onInit() which passes a success variable that we can use to troubleshoot the application. We saw this during class as the Initialization was failing until we properly connected the button. In the onInit() method, we set the Language to English and catch any errors with an if-else branch. If there are no errors, the application will connect to the speaker and output as audio the user's text input.

Finally we have to clean up our TTS object so that other applications can use this function. We override the onPause() and onDestroy() methods so that we can stop the TTS object.

**User view**

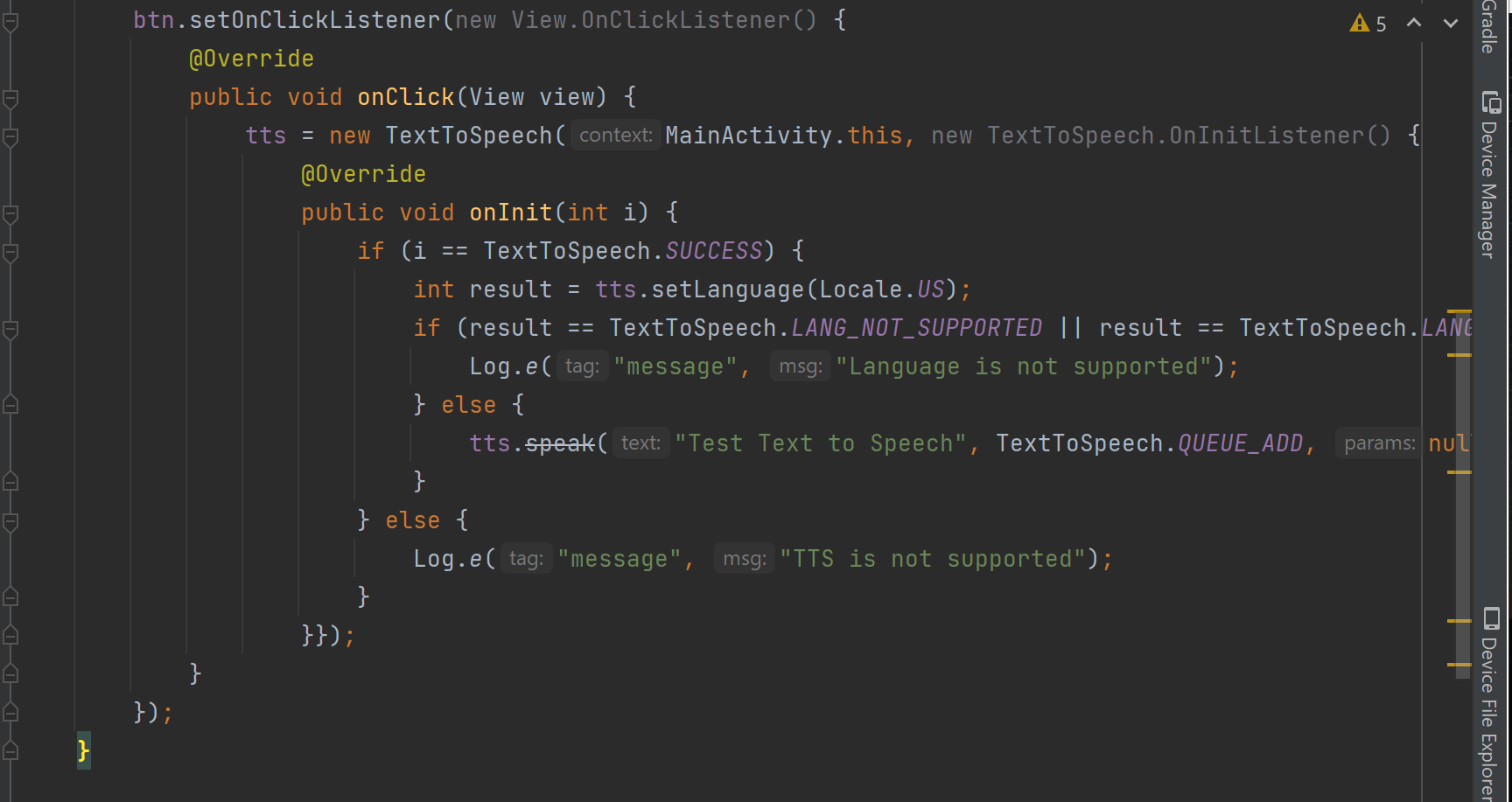
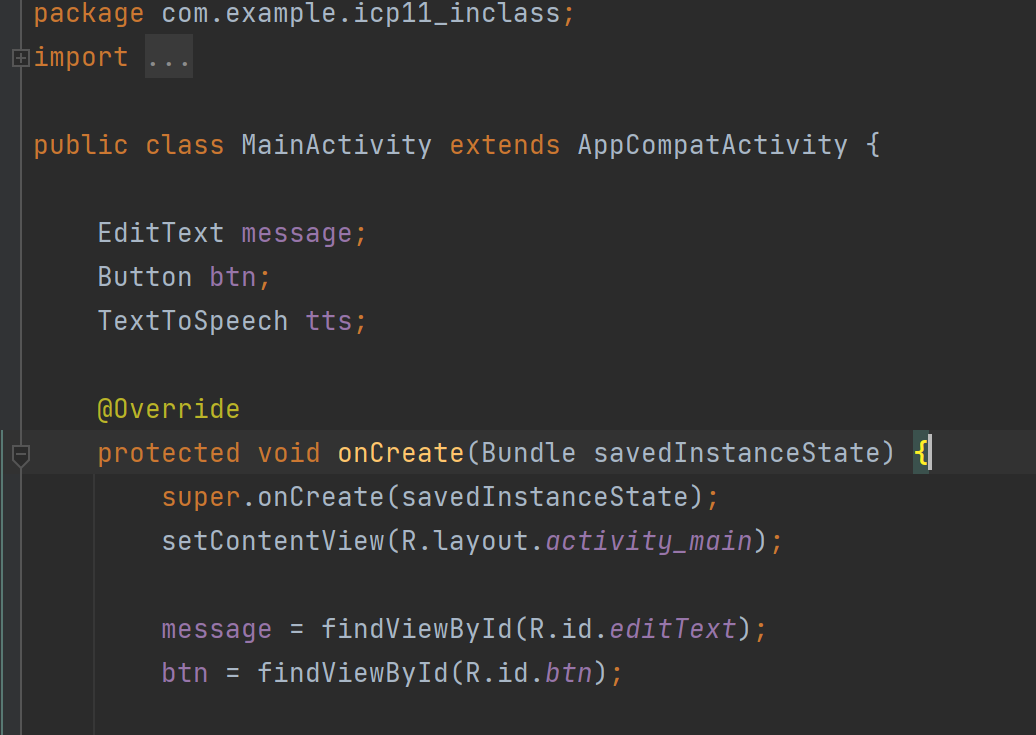
****

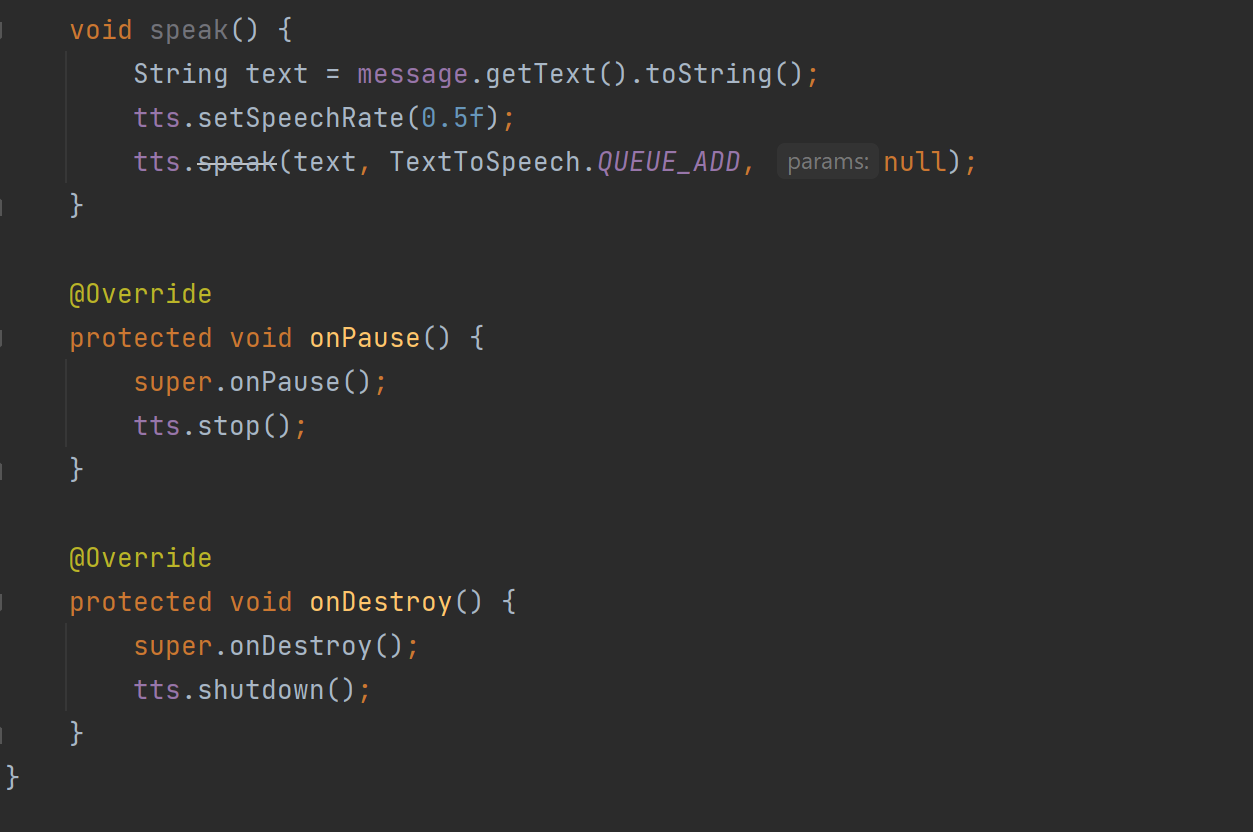
**Activity\_main.xml**





**MainActivity.java**

****

****