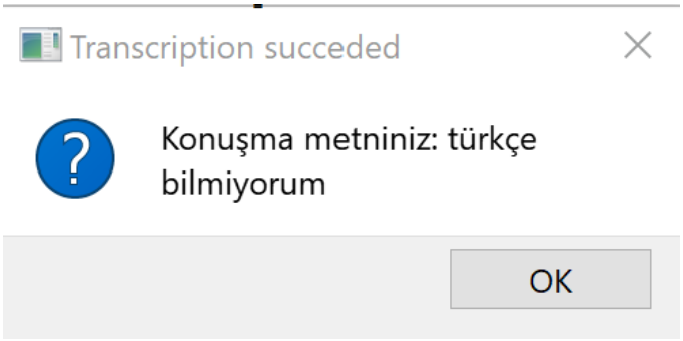
GROUP 14:
Buse Çeltik
Adam Moszczyński
Berra Bezer
N. Kağan Akdoğan

1. **Introduction to project:**

Python interface implementation that converts speech to text.
Our project is a simple and efficient Python interface application that can detect voice through devices like a microphone and convert it to a text. Our project can detect 5 languages (German, Turkish, English, Polish, Spanish) and display the speech of the spoken language on its user interface (UI). Our main target is to detect the spoken speech clearly and cancel the unnecessary background noise of the detected voice so that we can convert the speech to a text accurately and precisely. In addition, we kept the user interface very simple and elementary for easy usage for anyone. Only a mouse is needed to control the application. The button placements are well thought of to give an easy access to the user. The font chosen for the text is simple to read. Instructions and information of the state of the transcription is given to the user at every step of the application experience. Display of the transcribed speech is given specifically in its own language. For instance, a Turkish speaking user will see the final text after the explanation title “Konuşma Metniniz”, meanwhile an English speaking user will see the final text after the explanation title “Your text:”.



To use this application, you can simply check the language checkbox and then click on the button with a red dot and start talking. After you are done, the program will show you the text and automatically copy it.

2. **Code details**:

Libraries used in the project:

speech_recognition - Library for performing speech recognition

PyQt5 - Qt is a set of cross-platform C++ libraries that implement high-level APIs for accessing many aspects of modern desktop and mobile systems. These include location and positioning services, multimedia, NFC and Bluetooth connectivity, a Chromium based web browser, as well as traditional UI development.

sys - This module provides access to some variables used or maintained by the interpreter and to functions that interact strongly with the interpreter

Our program is working as an App class using QTWidgets and our own STT_ui user interface. Program contains 3 files: Speech_to_text.py, STT_ui.ui, STT_ui.py

```
class App(QWidget, STT_ui.Ui_Dialog):
    '''
    Class of dialog window
    '''
    def __init__(self, parent=None):
        '''
        This method initializes GUI
        '''
        super(App, self).__init__(parent)
        self.setupUi(self)
        self.app_working()
```

Code above is initializing App class and Ui. Then it launches the app_working method.

```
def app_working(self):
    '''
    This method reacts to the record button click
    '''
    self.recordButton.clicked.connect(self.on_click)
    self.show()
```

This method constantly checks if the record button was clicked and updates ui. In the case that user clicked the button. The on_clicked method is called.

```
def on_click(self):
    try:
        lng = self.check_language()
        self.STTConversion(lng)
    except UnboundLocalError:
        self.recordLabel.setText("Please select language")
```

This method tries to call check_language method and save it to the “lng” variable. Next it calls the main method of speech to text conversion with a given language. When “try” will throw an error that we are expecting. Label in the Ui will change its text.

```
def check_language(self):
    '''
    This method checks which language is choosen to recognition of speech
    and returns language variable.
    '''
    if self.German_tick.isChecked() == True:
        language = "de"
        txt = "Dein text ist: "
    elif self.Turkish_tick.isChecked() == True:
        language = "tr"
        txt = "Konuşma metniniz: "
    elif self.Polish_tick.isChecked() == True:
        language = "pl"
        txt = "Twój tekst to: "
    elif self.Spanish_tick.isChecked() == True:
        language = "es"
        txt = "Su texto es: "
    elif self.English_tick.isChecked() == True:
        language = "en"
        txt = "Your text: "
    return language, txt
```

check_language contains conditions which are checking what language the user chooses by ticking the checkbox. Then it returns a tuple of strings version of the language's shortcut and message box text in each language.

```
def copy_to_clipboard(self, text):
    '''
    This method does a copy of text and puts it to clipboard
    '''
    cb = QApplication.clipboard()
    cb.clear(mode=cb.Clipboard)
    cb.setText(text, mode=cb.Clipboard)
```

This method copies to the previously cleared clipboard a text given as an argument. cb is a variable that is using QT's library function to save text to clipboard.

```
def STTConversion(self, lng):
    '''
    This method does all needed things to transcript the voice to speech in
    a given language.
    '''
    self.recordLabel.setText("Listening...") #Changes label text to Listening...
    QtCore.QCoreApplication.processEvents() #Makes gui reacts to every label change
    with speech.Microphone() as source: #Setting up voice recognition features
        SpeakToText = speech.Recognizer() #
        SpeakToText.adjust_for_ambient_noise(source)
        audio = SpeakToText.listen(source)
        try:
            self.recordLabel.setText("Transcripting...") #Changes label text to Transcripting...
            STT = SpeakToText.recognize_google(audio, language=lng[0]) #Does conversion
            self.recordLabel.setText("Transcription succeded") #Changes label text to Transcription succeded
            QMessageBox.question(self, 'Transcription succeded', lng[1] + STT.lower(), QMessageBox.Ok)
            self.copy_to_clipboard(STT) #CAlls function that copies to clipboard
        except speech.UnknownValueError: #Expects error
            self.recordLabel.setText("Could not understand")
```

This is the main method for the speech conversion. The label changes the text to “listening…”

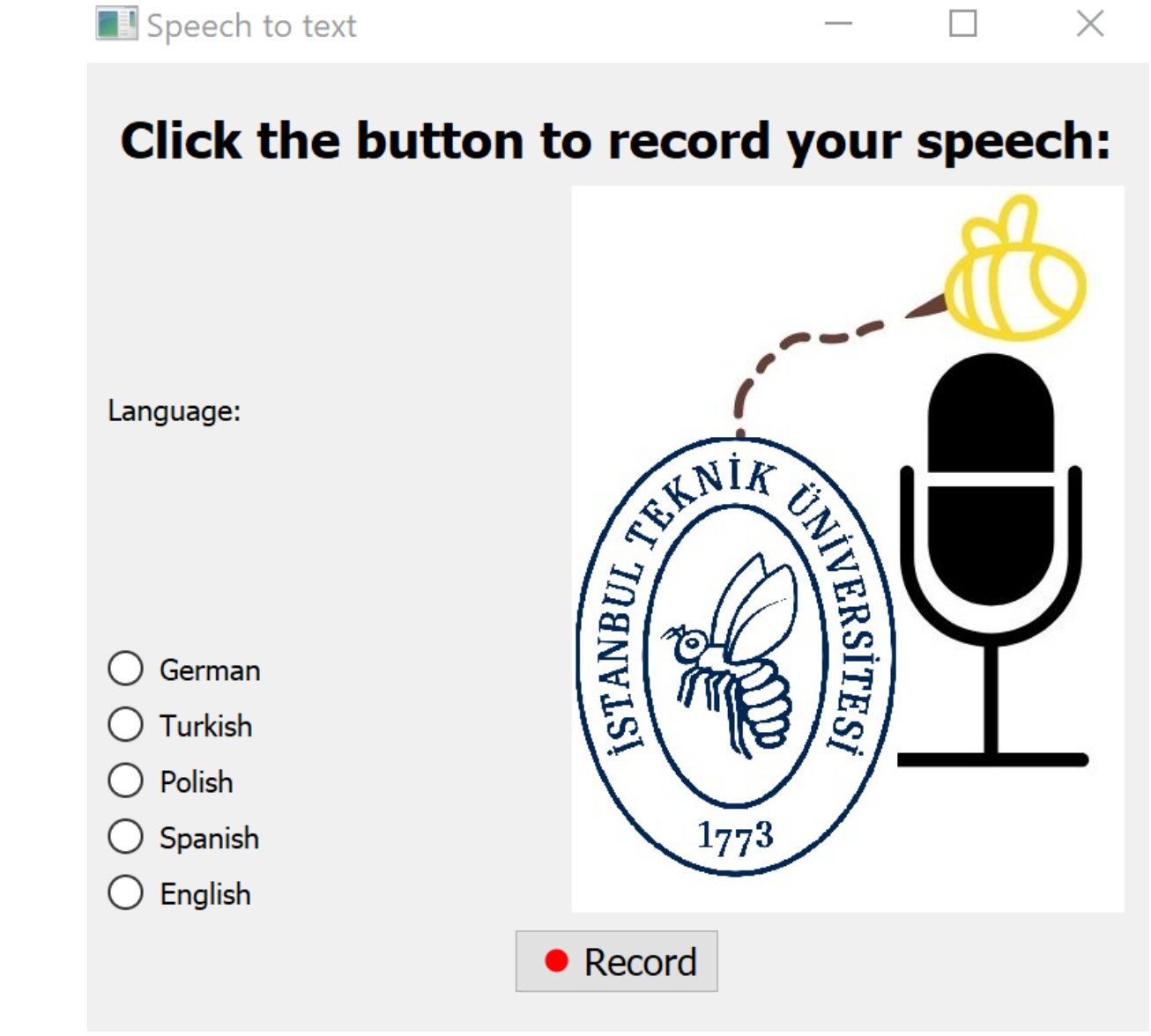
From **with** to **try** the code creates variables of recorded voice and cuts the ambient noise from microphone background. Between **try** and **except** the informing label is informing user about conversion in progres. After that the app calls a function from the speech_recognition library and give recorded audio and language as arguments. Then the user is informed about succes in transcription. In a second message box is created with text ready to copy. After closing the message box method copy_to_clipboard copies the text.

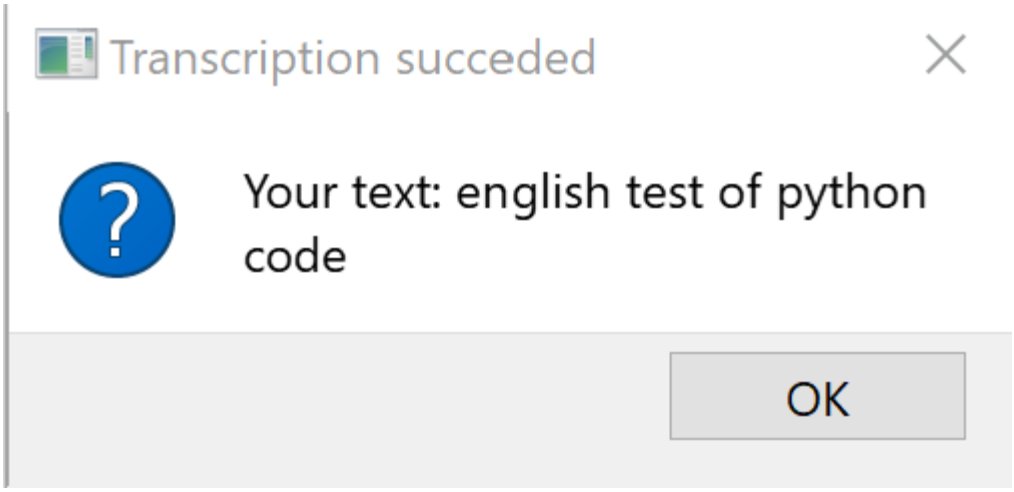
```
if __name__ == '__main__':
    app = QApplication(sys.argv)
    window = App() #Creating App class object
    window.show() #Launching UI
    sys.exit(app.exec_()) #Proper closing
```

Creating application object and showing working UI.

3. **Interface:**

   The interface consists of three simple parts to make the user experience smooth. The microphone icon refers to the voice detection part of the program which also has significant colors and symbols of ITU. The record button at the bottom middle part of the interface allows the user to start the voice detection. The text in the top middle of the interface explains what the user should do and guides the user at any instant. After the detection a new screen is shown where you can see the text version of your speech. If the “okay” button in the message box was clicked, converted text will be automatically copied to the user’s clipboard





4. **Problems faced**:

We encountered many problems while doing this assignment. However, only the biggest problems will be mentioned. Firstly, none of us had any experience on how to do an interface design. In order to solve this problem, the subject was investigated in depth by using internet resources. And this is how we managed to achieve the result

Secondly, sometimes background noises interfere with the speech and possibly block the speaking. For this issue we added a block to the code to ignore background noises. We tested the program many times to make sure it can cancel the background noise and convert it to a text in the precision we want.

5. **Summary** - In addition, the main purpose of our project, which is to put the listened voice into writing, was successful. Our program, which can detect 5 languages, provides convenience to the user by displaying the spoken language on the interface. Apart from this, we wanted to design a simple and understandable interface for ease of use. The control buttons, usage style and visual design of the program are designed for the user to use it comfortably.
   We thought of some features that could be added to improve the program to make it sustainable. One of the things we implemented is this: When the voice is translated into text, it is given to the user as "text" and the "okay" button appears. If the user clicks the OK button, the program automatically copies it to the user's clipboard. Other innovations can be: the number of languages detected by the program can be increased, the program can find and correct grammatical errors. With a button we will add, the program can directly search the text in search engines or present the matching examples to the user. We think that if our project is developed, it will be useful in many areas.