Module: “Technological Basics II”

Seminar: “Tech Basics II, Stream B (lecture)”

Teacher Name: Sarah Haq

Student Name: Anika Barop

Student ID: 4000794

Student E-Mail: anika.barop@stud.leuphana.de

Date of Submission: 22.02.2025

Project Report: “Coding a Music Recognition Application for Capoeira Music Using Streamlit”

Introduction

Music is an integral part of Capoeira: whether it plays back from a CD in the background during training, or it is played live during an event, the rhythm guides and controls whether we play fast or slow while the singer reflects on what is happening in the lyrics. Though many people think of Capoeira mostly as a sport, learning to sing and play different songs is just as important for progressing in Capoeira as is learning and perfecting the movements.

With my app "berimBAM", I want to help Capoeiristas find songs they like on the spot, making it easier for them to learn the lyrics and to sing or play along to them. On top of that, I wanted to provide additional resources to learn more about Capoeira history, theory and instruments, as well as creating a platform for Capoeiristas from all over the world to connect through music.

Development Process

The first step in the development process was to transfer the general layout of the app that I had created during Tech Basics I with tkinter to the Streamlit framework. My approach was that this would be the "web version" of my application, and my goal was to fully recreate my initial project within Streamlit and to make it usable. Existing music streaming apps like Apple Music, Spotify and Shazam were used as inspiration for the layout. I used the built-in function st.navigation to create a multipage app with one .py file for each page and put placeholder widgets on the different pages to mark where I would later on place certain widgets such as the text input bar on the search page.

I had decided early on that it would be possible to create your own account on the app to allow for some personalization and to, at some point in the future, be able connect to other people on the app. For this, I implemented a connection to MongoDB using the knowledge I had gained from Tech Basics II. I used two different collections: one containing the user's login data and one containing the profile data.

Then I spent a lot of time researching possible APIs that I could use to make the "Shazam" idea work. Some options would require me to build my own song database, while others directly worked with Shazam's database and had access to all the song data on there. Weighing out the pros and cons, I knew that if I wanted to create an app specifically and only for Capoeira songs, I would have to go with the first option and dedicate a lot of time to building a sizeable database with Capoeira music. However, considering the relatively small amount of time that was available to finish this project, and after checking that there was a reasonable amount of Capoeira songs available in Shazam's database, I decided to use the second option: A python library called shazamio. The documentation for this library was merely a README file in their GitHub repository that listed the different functions available in the library and an example of how to implement them. When used on their own,

the examples seemed to work just fine. However, as soon as I tried to use one of the examples with Streamlit, it didn't work. As shazamio is an asynchronous library, most of the functions are written using asyncio. The examples in the GitHub repository also use asyncio to create and run an event loop:

```
import asyncio
from shazamio import Shazam


async def main():
  shazam = Shazam()
  out = await shazam.recognize('dora.ogg')  # rust version, use this!
  print(out)

loop = asyncio.get_event_loop()
loop.run_until_complete(main())
```

(Code example taken from https://github.com/shazamio/ShazamIO/tree/master)

This doesn't seem to work in combination with Streamlit, so I tried a different approach by simply using asyncio.run() to execute the async function, which ended up working out perfectly.

After I had figured out the most important part – how to work with shazamio – I spent the remaining time fine-tuning, testing, finding errors and adding more content, particularly to the "Home" and the "Learn" page. I experimented with the different layout options provided by Streamlit, such as the expander, popover, and tabs, and implemented them in different parts of the application.

Challenges

Many of the challenges I faced throughout the development process were brought about by the limitations of the Streamlit framework. I was not able to customize the design of the app to extent that I would have liked – the config.toml file only allows for some minor theme customizations, but it is not possible to individually pick different fonts and colors unless you use custom CSS styling. Unlike in tkinter, you cannot decide where exactly you want to place a widget, and the widgets themselves do not have as many options for customization.

For example, I wanted to create a large button with the berimBAM icon for the "berimBAM" page, as I had previously done this with tkinter. The idea was to start recording sound after pressing the icon, so that the app could process the sound and potentially recognize a song. This would be useful from a branding perspective, as it would link the main purpose of the app, the sound recognition, to the app logo.

With Streamlit, it is not possible to display a large image on a button, neither does the st.image function provide an option to make an image clickable. At first, I used a custom component called st_click_detector by vivien to work around this limitation. However, after

working with shazamio, I realized that the easiest way to make it work with the smallest amount of needed clicks possible was to scrap the clickable icon and just use the st.audio_input widget. This meant sacrificing some style and branding to make the app easier to use and, conveniently, simpler for me to code.

Another challenge came with using st.navigation for my multipage app, as it proved difficult for me to create a page that would not show up in the navigation. I wanted to have a separate "song page" that would open if you clicked on a song but shouldn't be displayed in the sidebar. However, this is not possible using st.navigation, so I once again had to find a compromise without having to sacrifice the nice layout and style of the navigation bar. I settled on naming the page "Currently playing…" and having it display some placeholder text if you click on it without having selected a song.

Feedback

The testers for this app were a small group of people, some who do Capoeira and some not. For maximum efficiency, I made a questionnaire for them to fill in while testing the app. The feedback was mostly positive, people were generally satisfied with the layout and found the app to be intuitive and easy to use. Also, they didn't seem to have encountered any error messages.

Suggestions to improve the app included putting more of a highlight on the main purpose of the app – the sound recognition – by placing it directly on the home page, using bigger icons, and adding some functionalities like an option to bookmark favorite songs and share them with friends.

Another feedback that I will keep in mind when I try to improve on this app in the future is the fact that the music part and the learning part seem a bit disconnected. The tester kindly came up with a possible solution: Adding a list of related topics to learn about to the song you are listening to, or vice versa, adding related songs to some topics on the learn page. On top if this helpful suggestion, in the future I would also like to match lyrics to the songs so that users can learn them directly in the app. But that will have to wait until I have created my own database with only Capoeira songs.

References

asyncio documentation: https://docs.python.org/3/library/asyncio.html
pymongo documentation: https://www.mongodb.com/docs/languages/python/pymongo-driver/current/
shazamio documentation: https://github.com/shazamio/ShazamIO/tree/master
Streamlit documentation: https://docs.streamlit.io/develop/api-reference

Code referenced in comments.