Tech Basics 2 Project: Zongwr!ter

WINTER SEMESTER 2024/2025
LEUPHANA UNIVERSITY
TECH BASICS 2
SARAH HAQ

SUBMITTED BY MATHIS SCHRÄDER

Table of Contents

Development Process

In my first TB course, I created an app called „Zongwr!ter". It is an easy-to-use application for beginners in the music production scene. In this app, I implemented a piano roll, some pages for notes and to-dos and an unfinished page for learning something about music theory. This semester I tried to implement my app to streamlit. Very early in the process I noticed that it would be very complex to implement a piano roll into streamlit. I also noticed that my design choices would need to change.

I first started with the introduction page, a simple page where the function of the application is explained and where the user can learn something about the developer (me) and his vision. Then I thought about other music apps I use in my free time and noticed that many of them ask for some information about the user first before they can use the app. This info can be used for the developers to adapt the application to the targeted audience or add features for the future. That's why I also implemented a short voluntary form, asking the user for their age, prior music knowledge and reason for using the app before he gets to the first page. For that, I used MongoDB. As we already implemented MongoDB in our classes, I only needed to create a new cluster for my app, copy the code from the seminars and change the questions and variables. Still, there were many small problems with the form. I worked from many different places on my app and every time I tested my app, the connection to MongoDB wasn't working. After googling what the issue is, I found out that I had to enable every IP-address on the MongoDB-website.

To have a better overview of my code I decided to put all the code for the different pages in a different python file, put the code for the form in another and then import the definitions and the code for the form to my main app file. I also have a separate file for the text to music AI. Unfortunately, this AI only works when the app is run locally, as it uses too much memory for the free streamlit version on the website.

With st.sidebar.selectbox I managed to organize the different pages and make it visually appealing. For my app, I thought about four different additional pages: The first one is for lyrics production. It offers inspiration prompts for lyrics writing and uses the API "pronouncing" to find rhymes to words. The second page is all about music theory. I

implemented Youtube Videos about basic music theory and below there are quiz' about the content. The third page is the music AI tool, which, when started locally via pycharm, can create short AI written music and therefore can be an inspiration for the users own production. The last page is a feedback page, which is also connected to MongoDB. There, the users can leave some feedback about the app, e.g., which features they want to see in the future or which pages can be worked further on.

For the general design of my app, I used most of the code from our classes: From containers to columns, expanders and buttons. For the colors and font I researched on the streamlit website and found out that you can change it with a config.toml file in the .streamlit folder.

Challenges and Limitations

The biggest challenge in the development of my streamlit app was working with the session.state. Before I used the st.sidebar, I tried to manage my entire app with session.state and it was really confusing and frustrating. Furthermore, the music AI tool doesn't work on the streamlit website. This was especially frustrating because for me it was the highlight of the app. I tried to find other AI but for most of them I would have to pay and others had the same problem with using too much memory. Another small problem, I unfortunately couldn't fix is that to submit the first form, you have to press the button twice. Regarding the requirements.txt, I also had some problems as I forgot to add the libraries, leading to the streamlit app not working.

Feedback from Peers

After finishing my code, I sent the link to my streamlit website to my peers to test it. In general, they told me that it is easy to use, fun and that they really like my design. However, they noticed that the music AI tool is not working; the app crashed when they submitted their prompt. Unfortunately, I could not fix this, as such AI tools use too much memory for the streamlit website. Furthermore, I changed some of the questions in the first form as they were not really useful. Regarding accessibility, it is rather hard to incorporate when creating a

music app. Nevertheless, I focused on theoretical parts and lyric creation of music making it accessible for deaf people.