UNIVERSITY OF LEICESTER

# School of Computing and Mathematical Sciences

# CO7201 Individual Project

## Preliminary Report

## SongAssist

**Gabriel Knight**

**gk247@student.le.ac.uk**

**249047672**

**Project Supervisor: Newman Lau**
**Principal Marker: Dr Martina Palusa**

**Word Count: 3,003**
27/06/25

## Contents

## Aims and Objectives

Learning a new song as a guitarist, even in the age of modern technology, can present significant challenges. One of the most significant barriers in achieving this is often replicating nuances and intricacies in the songs guitar part which requires a deep understanding of the sound and technique *[1]*. By simply listening to a track it can be difficult to pick up on such subtleties, especially with all other instruments playing alongside the guitar track. Since the late 2010s stem-separation technology has been widely available as a proposed solution to this issue whereby each instrument in any given track can be isolated using deep learning AI models *[2]*.

While existing stem-separation applications can isolate guitar tracks, no leading applications in this field cater specifically to guitarists by offering further insight on the isolated part; this leads to an incomplete user-experience where users can hear the isolated guitar but have to navigate to a separate app in order to fully learn how to play it through chord-sheets and tablature notation. This report covers '*SongAssist'*, a stem-separation web application that aims to provide a comprehensive practice environment specifically targeted at guitarists by featuring AI-powered qualitative insights and advice in addition to stem-splitting and other helpful audio-manipulation tools.

Research has shown that it can take over 20 minutes to re-enter deep focus after an interruption from context-switching *[3],* for a guitarist constantly switching between a stem-splitting and notation application this can make for an inefficient learning workflow; *SongAssist* exists to streamline this process by providing all the tools necessary in one space. The application is to be built around a pedagogical feedback loop in which the user can upload an mp3 file, isolate the guitar track from other instruments in order to hear specifically what is being played and proceed to use *SongAssist*'s AI-assistant to generate tablature and answer any questions the user may have on replicating the guitar technique required. Additional features include speed adjustment, bookmarking and playlist management.

*SongAssist'*s stem-separation will be powered by *demucs,* a powerful open-source deep-learning model that works directly on the raw audio waveform of a file to isolate different parts using a U-Net algorithm [*4*]. Google's *Gemini* model will be utilized through the *Vertex API* for *SongAssist*'s assistant due to its strong performance and fine-tuning options.

The integration of multiple useful AI-tools that combine to produce a complete user-experience is what makes *SongAssist* unique from the fragmented workflow offered by existing software.

### Challenges

Development of *SongAssist* will involve a number of significant engineering and design challenges that must be assessed and resolved efficiently.

A primary concern is ensuring that the stem-separation model performs well; although *demucs* is shown to offer superior separation quality to other available open-source models, this can come at a great computational expense due to the size of the model [*5*]. With a lot of web browsers having limits in regards to processing resource-intensive content [*6*], running a powerful model like *demucs* client-side could potentially crash the page and lead to a vastly different UX depending on the power of the user's computer. Deploying the stem-splitting model through *Google Cloud Run* should ensure strong performance across all devices and also provides flexible deployment options [*7*].

The use of a Large Language Model (LLM) in Gemini generating advice and tablature for any given song presents a number of obstacles. Studies have revealed that without proper implementation, LLMs have a tendency to give incorrect information, especially with musical notation which it is not specifically trained in [8]. To combat such ‘hallucinations’, carefully structured prompt engineering must be employed whereby Gemini is supplied with clear chain-of-thought prompting; the AI-model should be allowed to focus on one aspect of the song at a time if this is correctly implemented [9]. Even with efficient prompt engineering in place, there is still a chance of encountering inaccuracy issues which is why it’s important to clearly convey to users that any information generated is from AI and not infallible.

*SongAssist* at its core is a Human-Computer Interaction (HCI) project that differentiates itself based on facilitation of an unfragmented workflow, without a straightforward and easy-to-use interface there are various similar products that can achieve superior stem-separation so this should remain a priority throughout the development life-cycle. As mentioned previously, *SongAssist* offers extensive audio manipulation features like speed-adjustment, implementing such features while keeping the UI simple and uncluttered will require the use of a contextual user-interface that ensures advanced features are only visible when needed [10]. Providing visual feedback in the user-interface is also crucial to reinforce the user’s understanding of the application.

## Requirements

### High Level Objectives

The main goal with *SongAssist* is to develop an Intelligent Tutoring System (ITS) catering specifically to guitarists. By leveraging multiple AI-technologies together, *SongAssist* looks to give users the advanced tools to learn a song without the distraction of using multiple applications at once, preventing a fragmented workflow.

### Implementation Requirements

**Essential:**

- *Provide high quality stem-separation.*
  - Allow the user to upload any MP3 file containing a fully-mixed song and isolate the guitar from the rest of the track.
  - Conversely, allow the user to isolate the rest of the track from the guitar which is optimal for practising learn guitar parts.
- *Provide an AI assistant that gives an advice on playing the song.*
  - Gemini provides the user with qualitative feedback on how to play the guitar in the song.
  - The model will use the title of the MP3 file to determine the song.
- *Develop functionality for generating guitar tablature based on the MP3 file provided.*
  - Generate musical notation in the form of guitar tablature.
- *Provide a straightforward user-interface.*
  - Provide a clear and intuitive UI that anyone with limited technical knowledge will have no issue navigating.
  - The interface aims to be contextual with little clutter.

**Desirable:**

- *Enable users to create, edit, and manage multiple playlists of songs they are learning.*
  - Let users quickly access all of the songs they’ve been learning, further reducing workflow interruption.
  - Google Firestore is being considered as an option for saving these playlists to use in future sessions.
- *Include an interactive stem mixer.*
  - Give the user control over the mix of the song, useful if they wish to turn a specific instrument up/down.

- *Incorporate a feature to adjust the playback speed of the MP3 file without distorting the pitch.*
    - Allow the user to speed-up and, more importantly, slow-down the song.
    - If the user wants to get a feel for the song without having to keep up, slowing down the track to play along to is a good option.
    - Additionally, slowing down a song can be helpful for analysing the guitar playing.
- *Implement a feature to add and delete bookmarks at specific timestamps within a song.*
    - If a user is struggling with a specific part of the song, they can bookmark it in order to quickly navigate to that section.
    - Improves workflow by reducing time spent repeatedly seeking the section.

**Optional:**

- *Develop a feature to allow users to pitch shift their MP3 files without affecting playback speed.*
    - Let users transpose the song, this can be beneficial if the guitarist wants to play the song in a lower tuning.
    - Not as necessary for most guitarists but is a very useful feature for those who need it.
- *Allow users to name or label their bookmarks for easier identification.*
    - If a user has many bookmarks this could be an important feature allowing them to easily distinguish each bookmark.
- *Implement a waveform display of the audio track to allow for more precise bookmark placement.*
    - Visualize where each bookmark is located throughout the track.
- *Develop a customizable user-interface.*
    - Provide functionality for changing the appearance (colours, theming etc.) of the user-interface.

## Technical Specification

### Architecture

As stated previously, *SongAssist* will implement a client-server architecture in which the back-end features are separated from the front-end in order to ensure consistent performance. Back-end features are also decoupled from each other which allows *demucs* to run in its own independent environment.

### *Front-End*

*SongAssist's* front-end will consist of a single page developed using *React*, this library was chosen for the wide variety of tools available. All user-interactions with the application will be handled through *SongAssist's* front-end. The web page will be hosted through *Netlify* due to its fast deployment.

### Back-End

FastAPI will be used due to its native asynchronous support, this is suitable for the heavy processing and frequent API calls that *SongAssist* will likely require. This API will be deployed as a container on Google Cloud Run, Firebase Storage will be used to temporarily store audio files being used by *SongAssist*. In order to process requests from the front-end, an API gateway will need to be implemented, this will receive the MP3 files from the front-end and generate a link to upload them to Firebase.

For stem-splitting, the *htdemucs_6s* model will be used for its ability to isolate guitar, this functionality isn't present in most other *demucs* models. The model is invoked and returns the *guitar.mp3* and *no_guitar.mp3* files to Firebase storage.

The back-end logic for the Gemini assistant will take user-prompts and engineer them specifically for advice on playing the song in the title of the MP3 file and looking up guitar tablature for this same song.

## Requirements Evaluation Plan

The stem-separation feature will be firstly evaluated functionally on its ability to retrieve an MP3 file and return two different tracks. Following deliverance of the two files, performance of the *demucs* model will be assessed based on how little bleed is present in each track and clearness of the isolation. A large number of differently produced songs will be tested with the application in order to ensure it performs consistently well.

When evaluating performance of *SongAssist's* Gemini assistant, a primary concern is that the information generated is accurate and relevant to the song that has been uploaded. Reliability is another metric that is crucial, the assistant shouldn't suffer any authentication issues and this will be addressed if any are found. Additionally, the assistant should present generated tablature and advice in a logical and structured manner to ensure readability.

It is crucial that all of the features within *SongAssist* complement each other to form an enjoyable user-experience. The ease with which first-time users interact with the system will be closely inspected to indicate how intuitive the user-interface is and which aspects can be improved.

Unit testing will be used throughout development in order to verify that each component of *SongAssist* is functioning as intended. Following this, integration testing will ensure all components are communicating properly.

**Participants**
In order to gain valuable feedback, it is necessary to have other guitarists use *SongAssist* and analyse their interactions with it. Currently it is intended for 3 guitarists to test out the application once it is nearing completion; it is hoped that this provides a somewhat comprehensive overview on the usability of *SongAssist*.

## Background Research and Reading List

Research consistently shows that an excessive amount of information being presented to an individual at once can impair their ability to take in new information, this is known as *cognitive load theory [11]*. It is for this reason that a fragmented workflow is entirely unconducive to learning a new song as a guitarist. Developing a user-experience that supports simplicity and intuitiveness has shown to increase a user's capacity to learn with many users in this study stating their lowered barriers to engagement after experiencing a straightforward user-interaction [12].

With stem-separation having existed as a technology for over 20 years, it was mainly looked upon as a complex studio technique until the late 2010s. The introduction of deep neural networks led to significant advancements in this field, stem-separation went from being performed using frequency filtering to deep-learning models trained on specialized music datasets [13]. Some of the most powerful deep neural networks developed at this time include *demucs*, which features high quality at a high computational expense, and *Spleeter*, a simpler model with fast stem-separation at a lower quality [14].

*Moises.ai* is one of the leading stem-splitting web applications with its own proprietary high-quality separation models, it has been stated that the separation offered by the application can be ‘studio-quality’[15]. Although Moises.ai offers high-quality separation, it doesn’t provide any qualitative advice on how to replicate guitar tone or technique, another app is required for this functionality.

**Reading List:**

- Cheng, Lee. "The impact of generative AI on school music education: Challenges and recommendations." *Arts Education Policy Review* (2025): 1-8.
- Barreto, Laura, Paul Taele, and Tracy Hammond. "A stylus-driven intelligent tutoring system for music education instruction." *Revolutionizing education with digital ink: The impact of pen and touch technology on education* (2016): 141-161.
- Rhodes, Chris, et al. "Ground truths: challenges and opportunities in developing an AI guitar assistant." Frontiers in Computer Science 7 (2025): 154933
- Michałko, Aleksandra, et al. "Toward a meaningful technology for instrumental music education: Teachers’ voice." *Frontiers in Education*. Vol. 7. Frontiers Media SA, 2022.
- Lathkar, Malhar. "High-Performance Web Apps with FastAPI." *California: Apress Berkeley* (2023).
- Apaydinli, Köksal. "Intelligent tutoring systems in music education." Current studies in social science (2020): 3-32.
- *Sularso, Sularso, et al. "A Comprehensive Visualization for Music Education and Artificial Intelligence." JOIV: International Journal on Informatics Visualization 9.2 (2025): 718-727.*

## Time-Plan

| **Phase** | **Week(s)** | **Date(s)** | **Objectives** | **Milestone** |
|---|---|---|---|---|
| 1. Laying foundations | 1 | June 27th - July 4th | • Set up development environment (cloud project and server)<br>• Finalize tech stack (decide on using a database for state saving)<br>• Start to design user-interface | • Development environment initiated<br>• Tech stack decided |
| 2. Developing a Minimum Viable Product | 2-4 | July 5th – July 25th | • Implement basic front-end structure<br>• Develop API endpoint for accepting an MP3 and invoking *demucs*<br>• Implement AI Assistant<br>• Develop stem-mixer | • Web page accessible<br>• Front and back-end basic components in place |

| | | | | |
|---|---|---|---|---|
| 3. Feature Refinement | 5-7 | July 26th – August 15th | • Refine AI assistant to give detailed song-specific advice (including tablature generation)<br>• Playlist management implementation<br>• Develop bookmarking system<br>• Provide Playback Speed Control | Application is now fully-featured and ready for testing which will inform any future changes. |
| 4. Feedback and Evaluation | 8-9 | August 16th – August 29th | • Perform user acceptance testing<br>• Make any necessary bug fixes and UX refinements | *SongAssist* is a robust and user-friendly interactive tutoring system. |
| 5. Final Documentation | 10-11 | August 30th – September 12th | • Completion of final report<br>• Completion of viva<br>• Deployment of application | The final documentation for *SongAssist* is completed and it is ready to be deployed. |

## <u>Risk Plan</u>

| **Risk Description** | Likelihood | Impact | Mitigation Strategy |
|---|---|---|---|
| Poor stem-separation performance | Medium | High | Decouple the model from the front-end of the application in order to prevent crashes |
| LLM inaccuracies and hallucinations | High | High | • Enforce structured prompt-engineering<br>• Warn users that content is AI-generated |

| | | | |
|---|---|---|---|
| Cluttered user-interface | Medium | High | • Seek user feedback at every stage of development<br>• Prioritize core features<br>• Implement contextual UI |
| Excessive scope expansion | High | High | Implement core features first and focus on MVP |

## Conclusion

To conclude, *SongAssist* is an all-in-one practice app for guitarists to learn new songs and refine their skills in a user-friendly, distraction-free environment. The application implements cutting edge stem-separation models to isolate the guitar from the rest of the track as well as AI-powered advice and tablature generation to give guitarists a deeper understanding of the song they are learning. Following on from this, the libraries and technology stack defined in this report will need to be finalized and the user-interface must be drafted up to prepare for developing a minimum-viable-product.

## References

- [1] J. R. Keebler, et al., “Shifting the paradigm of music instruction: implications of embodiment stemming from an augmented reality guitar learning system,” *Frontiers in Psychology*, vol. 5, p. 471, May 2014, doi: 10.3389/fpsyg.2014.00471.
- [2] C. Kefalis and A. Drigas, “Web Based and Online Applications in STEM Education,” *International Journal of Engineering Pedagogy (iJEP)*, vol. 9, no. 4, pp. 76–85, 2019.
- [3] G. Mark, V. M. Gonzalez, and J. Harris, “No task left behind? Examining the nature of fragmented work,” in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 2005, pp. 321–330.
- [4] A. Défossez, *et al.*, “Demucs: Deep extractor for music sources with extra unlabeled data remixed,” *arXiv preprint arXiv:1909.01174*, 2019.
- [5] R. Hennequin, *et al.*, “Spleeter: a fast and efficient music source separation tool with pre-trained models,” *Journal of Open Source Software*, vol. 5, no. 50, p. 2154, 2020.
- [6] K. A. Khan, M. T. Iqbal, and M. Jamil, “The Impact of Built-in Ad-Blockers in Web Browsers on Computer Power Consumption,” *European Journal of Information Technologies and Computer Science*, vol. 4, no. 5, pp. 1-10, Nov. 2024, doi: 10.24018/compute.2024.4.
- [7] E. Bisong, “An overview of google cloud platform services,” in *Building Machine learning and deep learning models on google cloud platform: a comprehensive guide for beginners*, 2019, pp. 7–10.
- [8] J. Li, *et al.*, “The music maestro or the musically challenged, a massive music evaluation benchmark for large language models,” *arXiv preprint arXiv:2406.15885*, 2024.
- [9] L. Pond and I. Fujinaga, “Teaching LLMs Music Theory with In-Context Learning and Chain-of-Thought Prompting: Pedagogical Strategies for Machines,” *arXiv preprint arXiv:2503.22853*, 2025.
- [10] M. Dubiel, *et al.*, “A contextual framework for adaptive user interfaces: Modelling the interaction environment,” *arXiv preprint arXiv:2203.16882*, 2022.
- [11] J. L. Plass, R. Moreno, and R. Brünken, Eds., *Cognitive Load Theory*. Cambridge University Press, 2010.
- [12] A. Chu, *et al.*, “Usability of learning moment: features of an E-learning tool that maximize adoption by students,” *Western Journal of Emergency Medicine*, vol. 21, no. 1, p. 78, 2019.
- [13] E. Cano, *et al.*, “Musical source separation: An introduction,” *IEEE Signal Processing Magazine*, vol. 36, no. 1, pp. 31–40, 2018.
- [14] Quality, Separation Audio, and Erika Rumbold. "Computer Science Department." (2022)
- [15] K. N. Watcharasupat and A. Lerch, “A Stem-Agnostic Single-Decoder System for Music Source Separation Beyond Four Stems,” *arXiv preprint arXiv:2406.18747*, 2024.