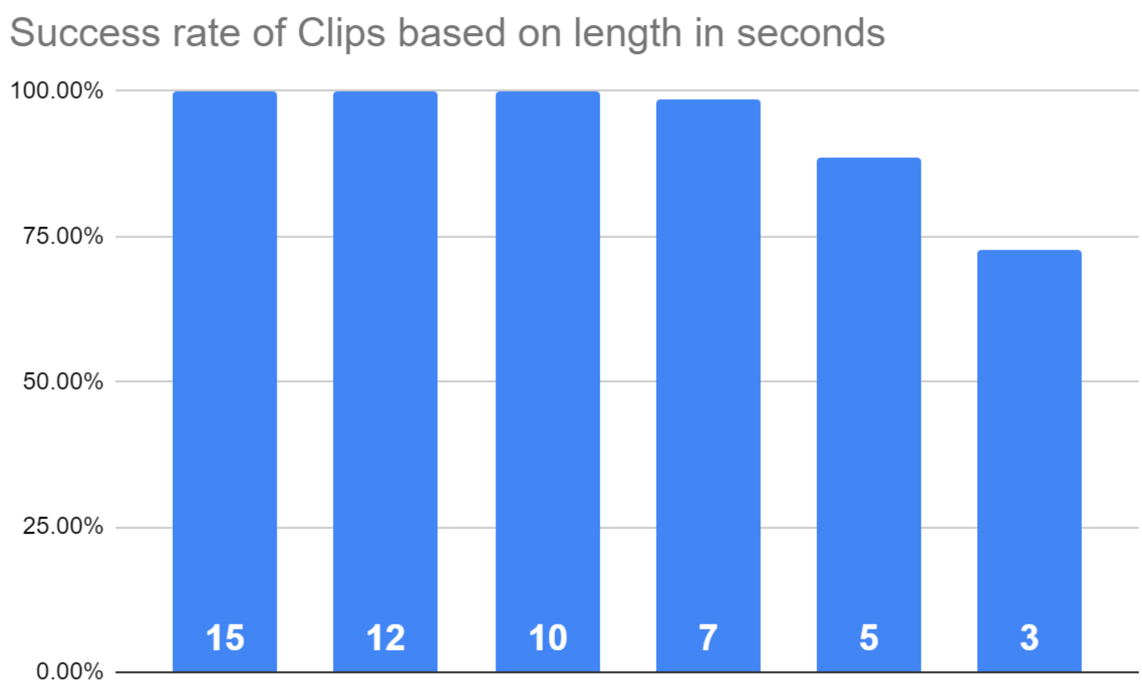
Testing Documentation

Project name: Sham Shazam
Student 1: Georgina Scanlon ID Number: 19392373
Student 2: Rhea Varkey ID Number: 19452962

Clip length experiments

We ran tests to determine the optimal clip length to use. We wanted to use the shortest length possible, while still maintaining a high probability of the API being able to correctly identify the song. In order to test this, we ran experiments using random song clips of varying length to determine what the shortest length we could use would be. We used a random spotify 'top songs' playlist in order to get songs that would be suited for our use case. From that playlist, we took a 15 second clip from each song starting at 1:00 in. We tested each of these clips with the API and recorded whether they could be identified or not. We did the same for 12, 10, 7, 5 and 3 second clips and compared the results.

Results:



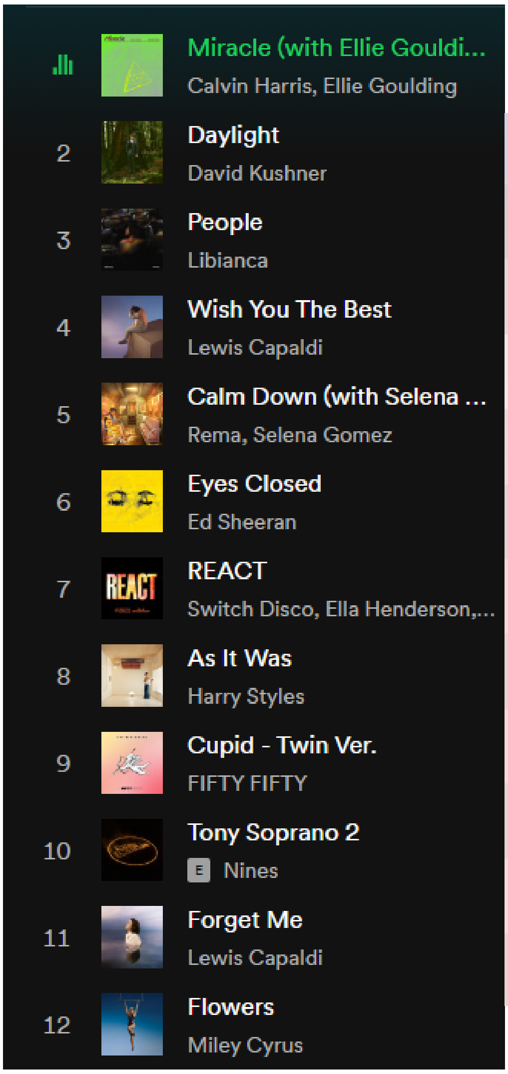
As can be seen from the above graph, we observed that the 15, 12 and 10 second clips all had a 100% success rate at being identified. The 7 second set had a 98.5% success rate, the 5 second set a 88.5% success rate and the 3 second set a 72.8% success rate.

We ultimately decided that 7 second clips would be the best to use based on this data. With a 98% success rate, we were satisfied that the vast majority of the clips could be identified. With the 5 and 3 second clips there was quite a big drop in success rate, so we decided these lengths would not be suitable. The 10, 12 and 15 second clip sets were all 100% successful at identifying the songs, but we decided that the performance of the 7 second clips was good enough and that using this shorter length would better suit our application.

Testing song identification

In order to test that the application was robust and effective at identifying songs, we simply let a random spotify playlist play through while the application was running and recording. We then checked to make sure that all songs that played had been saved successfully and were in the song history. We also wanted to ensure that songs were only saved once per play. The playlist we tested was the current 'UK Top 40' playlist on spotify. We chose this playlist because we were satisfied it would cover our use case, as we are mainly interested in identifying current pop songs.

Results:
Our test was successful with all songs being correctly identified and no errors appearing in the song history. While the main objective of this test was to ensure the songs were saved and appeared correctly in the song history, we also regularly checked that the correct song was being shown as 'Currently playing' and again found no errors during the test.



Title	Artist
Forget Me	Lewis Capaldi
Tony Soprano 2	Lime
Cupid (Twin Ver.)	FIFTY FIFTY
As It Was	Harry Styles
React	Switch Disco
Eyes Closed	Ed Sheeran
Calm Down (with Selena Gomez)	Rema
Wish You The Best	Lewis Capaldi
People (Remix) feat. Cian Ducrot	Libianca
Daylight	David Kushner
Miracle	Calvin Harris

Battery Usage:

We were interested to know how intensive our application would be on battery life. As our main use case intended for the application to be running for a long period of time, we were conscious that it shouldn't be too intensive on the battery usage. We observed the battery usage on both our laptops while leaving the application running for an hour. On Rhea's laptop approximately 30% of the battery life was used in this time and on Georgina's laptop, approximately 26% was used. We then observed how much battery was used when the laptops had no applications open, so that we had a baseline to judge off. Georgina's laptop used approximately 22% and Rhea's used 30% again. While this testing method was far from perfect and in an ideal situation we would test on multiple devices and in a more controlled environment, we were satisfied that the application was likely not extremely battery intensive.

Testing Denoising Method

We found it quite difficult to accurately test the effectiveness of our denoising implementation. In an ideal world, we would be able to have the application recording in a public space while playing music, but we couldn't really do that due to the disruption we would cause. Instead, we used random clips of noise and combined them with music clips so that they would overlap using a digital audio editing program called Audacity. This allowed us to test multiple versions of our denoising algorithm with the same clips of noise and music.

User Testing

An important part of our testing process was asking other people to try out the application and using their feedback to make improvements. Overall, we asked 12 people to test the application and asked them to fill out a form describing their experience.

Results:

Overall, we were happy with the results. The majority of users found the application easy and intuitive to use, which had been one of our goals. We used the feedback to make some changes. Some users suggested having a way to play the songs that were saved in the history or find out more details. Based on that feedback, we were able to add a link provided by the AudD API which links to various streaming services for the relative song in the song history page. With the added feature, users would easily be able to find the song they previously listened to, if they wanted to listen to it again or easily find out more information about it.

End-to-End Testing

We manually performed end-to-end testing on the application by simulating the behaviour of a new user. We ran through the signup process, identified a few songs, logged out and back in again and checked the song history. While doing this, we were focused on the overall usability of the application and how the UI looked.

Unit Testing

We created unit tests for some of our components which were used to test the components individually to ensure that they behave correctly. We used Jest and the standard React Testing Library to easily create and run tests. We were able to test our database models by passing mock data in to create mock documents and see if errors were handled correctly. For example, in some cases we would need to ensure all fields were provided (i.e users must provide a password when signing up). In other cases, we would need to ensure that the document was only created if the right data types were provided. Our unit tests needed to cover all these situations.