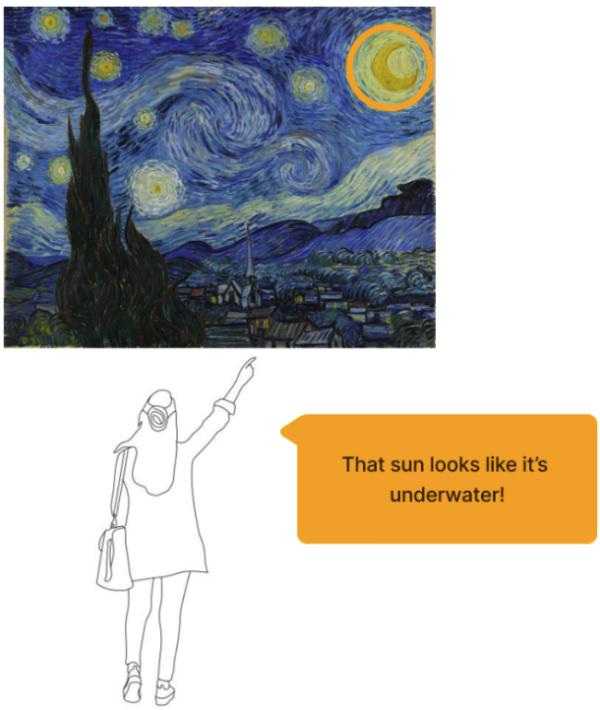
ArtTalk: Multimodal Interactions in Museums and Galleries

Maria Ascanio Aliño and **Trudy Painter** and **Andrew Stoddard**
mascanio@mit.edu, tpainter@mit.edu, apstodd@mit.edu



Abstract

ArtTalk is a web application designed to enhance visitors' engagement with artworks in museums and galleries. The application leverages a range of modalities to encourage interaction, including gesture and speech recognition.

The primary aim of ArtTalk is to encourage visitors to interact with artworks in a more meaningful and engaging way. The incorporation of multiple modalities makes the experience of interacting with art more intuitive and accessible for a wider range of visitors.

One of the key features of ArtTalk is gesture recognition, which allows users to place comments on specific points of interest simply by pointing at them. The application tracks the user's hand and identifies the pointing gestures, enabling them to easily and seamlessly place comments without the need for any additional equipment. In addition to gesture recognition, ArtTalk also uses speech recognition to identify and confirm the comments users place on specific points of interest. This feature ensures that the comments are accurately identified, minimizing the risk of misunderstandings or errors.

To further enhance the user experience, ArtTalk provides audiovisual feedback to users throughout the interaction process. This feedback makes the entire process of interacting with artworks more intuitive and engaging, providing visitors with a unique and memorable way to experience art.

ArtTalk is a web application that can transform the way visitors interact with artworks in museums and galleries.

1 Introduction

Museums are looking for ways to increase visitor engagement to ensure they still have a place in the modern world. A couple of our group members took the MIT CMS class Extending the Museum. During the class, we met with curators, archivists, gallery owners, and museum designers. Over and over, they stressed that they wanted to make museum visits more exciting and engaging for visitors. They want to reframe the museum as a participatory space: accessible to everyone.

One effective way to achieve this is by designing inviting interactive exhibits. The current museum model, where communication is one-directional from the museum to the visitor, is becoming outdated as phone and internet use have shifted people's habits. One way museums have looked to increase engagement is by trying to turn the museum inside out. They do this by allowing visitors to feel like they too are leaving a contribution to the museum, and not only can visitors learn about the works, but they can learn more about themselves and the other people in their community. We are proposing ArtTalk, a solution to increase visitor engagement in museums by allowing visitors to directly interact with all of the works, and learn from the people around them in the process.

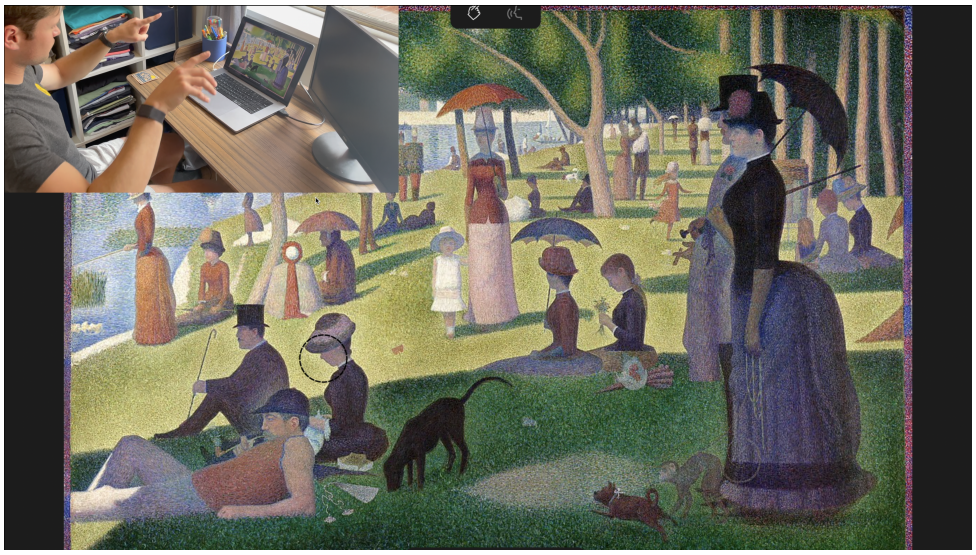


Figure 1: User pictured top left leaving a comment with his finger

2 System Overview

In a museum using ArtTalk, a user can walk up to a painting. When the system detects a person standing in front of it, the system announces to the user that they can start gesturing at a part of the painting that catches their attention. The user notices a part of the painting that they really like. They wonder what other people have also thought about that part of the work. They point and hold their hand still at that location.

The system announces to the user, "Someone else said the brush strokes are really intricate and the layering adds a 3D element to the canvas."

The user decides to add their own comment to that part of the work. They continue to point at that spot and say "I think the colors blend together really well." The system repeats their comment back, and the user confirms that it heard them correctly and has saved their comment for other museum patrons to view.

Pointing and talking about paintings are actions that people are already very familiar with. When looking at a painting with a friend, when one wants to discuss a part of the painting, they point it out as a reference to what they will say, and then they share their comments on the work. We aim to replicate this natural gesture and allow all visitors to connect with one another by leaving comments and exploring others' comments on the work.

3 How Does It Work

3.1 Core Technology

ArtTalk is a Next.js web application, written in TypeScript and hosted on Vercel. It employs a variety of modern technologies and techniques to provide a seamless interactive experience. Google's MediaPipe library is used for Hand Landmark detection. Built-in WebKit APIs are used for speech synthesis and recognition. A Prisma database is used to store comments and the location at which they were added. Finally, we use p5.js for heatmap visualization, rendering graphics to show where most comments are located. This visual representation provides an easy way to identify areas of high engagement or interest.

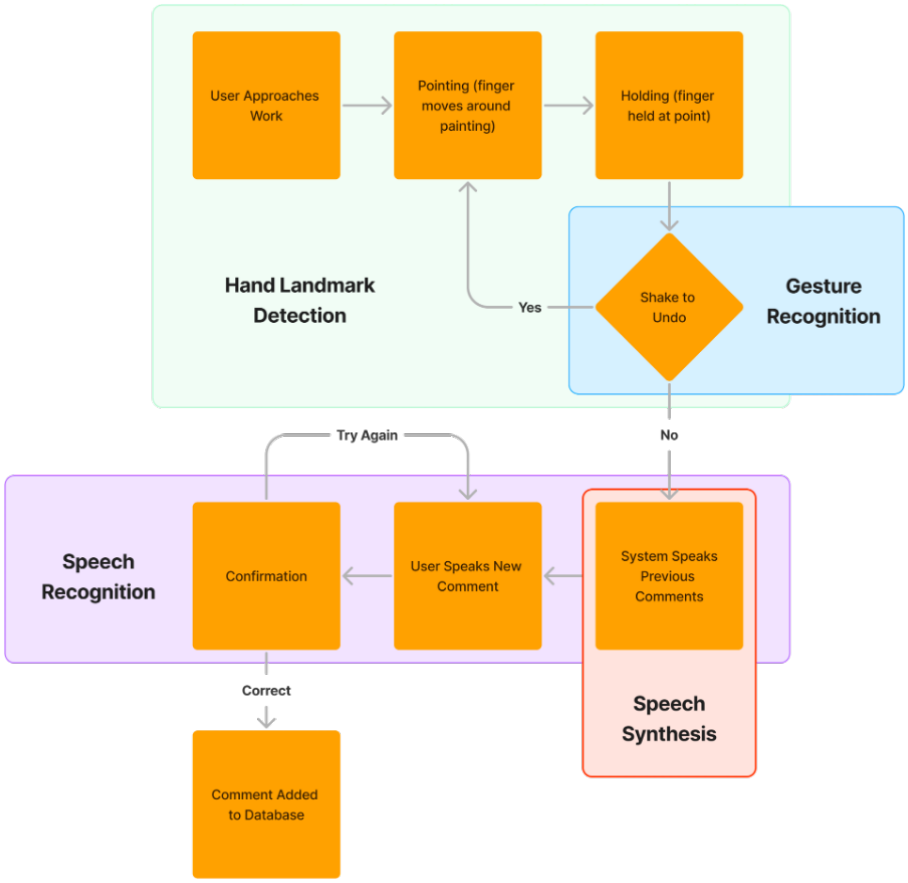


Figure 2: System Diagram for ArtTalk

3.2 Hand Landmark Detection

The process begins with the MediaPipe library, which is used for Hand Landmark detection. We capture video from the user's webcam, and with every frame, we pass the image to the MediaPipe model, which identifies and normalizes the location of the tip of the user's index finger. To smooth out movements, we calculate a running average of the finger location.

We then use this data to draw a circle on the painting representing what the user is pointing at. The system is designed to detect when the user's finger has paused at a location for more than a second, effectively "locking in" the location. If a pointer finger can not be detected for more than three seconds, the system resets back to its initial state.

At every step of this process, ArtTalk incorporates built-in browser speech synthesis that guides the user at different state points. It provides instructions for pointing, locking in, adding a new comment, and confirmation.

3.3 Comment Retrieval, Gesture Recognition, and Speech Recognition

Once locked in, ArtTalk initiates three processes: comment retrieval, gesture recognition, and speech recognition. Comments near a certain radius of where the user has pointed are retrieved from the Prisma database and displayed. If the system detects a shake-to-undo gesture, calculated from the slope of finger movement over the last 30 frames, it promptly resets back to the initial state. At the same time, we also employ WebKit speech recognition to transcribe any comments that the user wants to leave at that spot. A silence timer is used to detect when the user is done talking, prompting for verification that the comment is correct or whether the user wants to try again.

3.4 Features and Usability

3.4.1 Landmark Detection Evaluation and Limitations

ArtTalk's primary feature, the pointing mechanism, works remarkably well. It's intuitive and easy to use, providing a smooth experience for users. When being a couple of feet away from the camera and making sure only one hand was seen, the system was very good at tracking the pointing and locking in the location. This, however, became a problem when there was someone else in the frame or another hand. Since the system is currently only tracking one of the hands, it would get confused when more than one hand was visible.

The 'shake to undo' feature is another highlight of our system. It allows users to quickly reset and retry their interactions, enhancing the overall user experience. We did find, though, that it required considerable fine-tuning and testing to ensure its responsiveness and accuracy.

3.4.2 Speech Recognition Evaluation and Limitations

ArtTalk employs speech recognition for user input, which has proven to be quite efficient. However, its performance can be affected in loud environments, which is something we're looking into. When speaking into the microphone alone, the system did a very good job at recognizing the text the user wanted to comment on the artwork, but as soon as more people were around, the speech recognition was unable to filter the voices and would come up with incomprehensible comments.

Something that impressed us was how much easier it was to implement speech recognition and speech synthesis in a web application. Given that most browsers have built-in APIs that are able to do both, the logic of the speech synthesis and recognition was fluid.

3.4.3 Testing

To test the system we tested each individual modality and feedback as we worked on them. For gesture recognition, we were able to find its limitations and tested it within the context we were interested in. For speech synthesis, we tested multiple voices for the most human-like interaction possible, and for speech recognition, we stress-tested the system in loud environments as well as with long comments.

3.5 Interesting Failure

An interesting failure that we encountered in our final version of ArtTalk is how the system handles multiple hands or people in the frame. The Hand Landmark package is configured to only detect one hand, and provide the landmarks for that one hand. When two hands are in the frame, the system gets confused about which hand it is tracking. This can cause the pointing circle location feedback to jump back and forth between different places. The same thing can happen when there are multiple people in the frame. This compounds the issue, as there are now even more than two hands in the frame. In practice, this caused issues when someone was sitting down near the ArtTalk display, and had their hand resting in a place that was in view of the camera. This could cause the circle to get stuck in one place, as it was actually picking up on the static hand. One way we could remedy this issue is by allowing the detection of multiple hands. Google's MediaPipe has the ability to do so (however, only up to 2 hands). We can then only use the location of the hand that is moving, which can be determined by checking if the location of one of the hands is not changing, or changing very slowly. To fix the issue of multiple people in the frame, we would need to use a package other than MediaPipe. However, our current system design does not have a way to handle multiple people pointing at the same time. We could show multiple cursors on the screen,

but then we would need a way to match who is pointing to who is talking. This is an experiment for future versions.

4 Project modifications

We had general success with implementing our project. The entire tech stack was based on web libraries and is compatible with modern browsers.

4.1 Kinect

We originally wanted to implement ArtTalk using a Kinect sensor and projector around an original piece of artwork. We were able to set up the Kinect on our M1 Macbooks. However, using Kinect was overly complicated.

We realized we could instead track hand movements using a computer camera and web hand-tracking libraries. Although computer webcams do not provide depth information, we found that they were sufficient at estimating finger joint positions regardless. After extensive testing, we found that there was not any degradation in performance when compared to the Kinect output while providing a significant simplification in system design.

4.2 Shake To Undo

After the Implementation Studio, we noticed that users would often make mistakes and want to restart. We didn't have a way for them to redo their pointing position and comment.

To remedy this, we developed a "shake to undo" gesture. After a user has locked in their position for pointing, they can shake their hand in front of the camera to restart the process of pointing at a position in painting. To implement this function, we first needed to study what a shake-to-undo gesture actually looks like, in the form of hand landmark position data. We noticed that shake-to-undo has a few main components: rapid movement and multiple direction changes. Both of these can be determined from the velocity of finger movement. To determine exact values, we created another web endpoint that displays a graph of the slope of finger movement over the last 30 frames. We then took the absolute value of it, and then we want to count the number of peaks above a threshold. This captures both objectives - the slope must be fast enough (showing that the gesture is quick enough), and it must have a count above a different threshold (to indicate that there are enough back-and-forth movements).

4.3 Activity Heatmap

Figure 3: Heatmap of Realtime Comment Data

During our implementation studio, we received a suggestion that it would be interesting to see where users are most captivated by a painting. We developed a heatmap of real-time comment data overlaid on the original painting curators can access at https://arttalk.vercel.app/heatmap.

Curators can use this type of analysis tool to identify popular parts of a painting. It can be used to help them:

1. Adjust gallery layout to ensure high-engagement pieces are prominently displayed
2. Improve artwork descriptions to highlight the most compelling aspects of a piece or address common questions
3. Inform future acquisitions to match the interests of their patrons

5 User study

The purpose of our user study was to test the effectiveness of ArtTalk. We focused on the user experience of the application, assessing its ease of use, effectiveness, and overall impact on the visitors' engagement with the artwork.

We conducted the user study over the course of two weeks, and a total of 9 participants were recruited to take part in the study. Participants were asked to interact with selected artworks using ArtTalk and to provide feedback on their experience.

During the study, we found that ArtTalk was effective in enhancing visitors' engagement with artworks. Participants reported that the application was easy to use, and the gesture and speech recognition features were intuitive and seamless. Participants also mentioned that the audiovisual feedback provided by the application enhanced their overall experience of interacting with the artworks.

Something we learned during this experience was that the application was not as effective for certain types of artworks or exhibits (for example an abstract painting vs. a realistic painting). This made us realize the importance of tailoring the application to different artworks by either using a narrower radius for the area of interest or segmenting the images in certain scenarios.

We also learned that users sometimes felt "stuck" in the state of our application. This is if they incorrectly locked in on a spot and did not know how to "unlock" themselves. This feedback, combined with our feedback from the implementation studio, validated the need for us to implement the "shake-to-undo" gesture.

6 Performance

6.1 Crucial Insights

Throughout the development and evaluation of ArtTalk, we gained valuable insights into its performance and identified areas of success as well as limitations. The project offered a rich learning experience that allowed us to understand the practical challenges and trade-offs involved in designing an interactive system for museum engagement.

One notable aspect of ArtTalk's performance is its successful implementation of gesture and speech recognition. The hand landmark detection using MediaPipe proved to be effective in tracking users' pointing gestures and locking in the locations of interest on artworks. Users found this feature intuitive and seamless, enhancing their interaction with the artworks. Similarly, the speech recognition feature accurately transcribed users' comments, providing a convenient and efficient means of input.

However, we also encountered limitations and areas for improvement. One particular challenge was the system's handling of multiple hands or people in the frame. Currently, ArtTalk is designed to track and respond to a single hand, which can cause confusion and incorrect feedback when there are multiple hands present. This limitation restricts the application's usability in scenarios where multiple users are interacting with the system simultaneously or when users are in close proximity to each other. To address this limitation, future iterations of ArtTalk could explore incorporating multiple hand tracking using technologies like Google's MediaPipe, which supports the detection of up to two hands. By distinguishing and utilizing the movement of the hand that is actively pointing, the system could provide a more accurate and reliable user experience in multi-user environments.

6.2 Interesting Next Step

An interesting next step that is just out of reach for ArtTalk is the ability to facilitate collaborative interactions among users. Currently, the system allows users to individually leave comments on specific points of interest. However, it does not provide a mechanism for users to engage in a conversation or build upon each other's comments. Enabling collaborative interactions would open up new possibilities for visitors to engage with the artworks and with each other. For example, users could reply to existing comments, initiate discussions, or contribute additional insights related to specific points of interest. By fostering a sense of community and shared exploration, ArtTalk could deepen visitors' engagement and create a more dynamic and interactive museum experience.

6.3 Improvement Plans

To improve the system and take its performance to the next level, several steps can be taken:

1. Enhanced Multi-user Support: As mentioned earlier, incorporating multiple hand tracking and developing a mechanism to handle interactions from multiple users simultaneously would be a crucial improvement. This would involve accurately identifying and associating individual users' gestures, comments, and interactions with specific points of interest, enabling seamless collaboration and shared engagement.

2. Advanced Speech Recognition: Further advancements in speech recognition capabilities could help address challenges in noisy environments and improve the system's ability to filter and understand multiple voices. Techniques such as voice activity detection and speaker diarization could be explored to enhance speech recognition performance in real-world museum settings.

3. Fine-tuned Pointing and Commenting Mechanism: Refining the pointing mechanism to accommodate different types of artworks, such as abstract paintings or sculptures, would be valuable. This could involve adjusting the radius of the area of interest or implementing image segmentation techniques to identify specific regions for interaction.

4. User Interface and Experience Enhancements: Iterative design and user feedback could drive improvements in the user interface and experience of ArtTalk. Streamlining the user journey, providing clearer instructions, and refining the audiovisual feedback could enhance usability and overall satisfaction for visitors.

5. Robust Testing and Deployment: Conducting extensive testing and deploying ArtTalk in real museum settings would provide valuable insights into its performance, user engagement, and scalability. This would enable the refinement of the system based on real-world usage scenarios and feedback from museum visitors and staff.

By addressing these areas for improvement, ArtTalk can continue to evolve as a powerful tool for enhancing visitor engagement in museums and galleries, creating a more interactive and immersive museum experience.

7 Task Breakdown

7.1 Maria Ascanio Aliño

- Speech Recognition for comment recognition
- Speech Recognition to submit a comment or restart comment recognition (try again vs. correct)
- Speech synthesis logic for each individual FeedbackType

7.2 Trudy Painter

- User interface design
- Workflow and logic for different stages of pointing and leaving comments (React state management)
- Configuring Github repository for web application using Vercel, React, and Typescript
- Setting up backend database and data types to store comments using Prisma
- Heatmap for real-time comment insights

7.3 Andrew Stoddard

- CanvasWithGuesture Component
- Configuring and implementing hand landmark detection
- Algorithm and implementation to detect a "pause" in hand movement to "lock in"
- Algorithm and implementation for "shake to undo"
- Graph endpoint to determine thresholds for "shake to undo"

8 Acknowledgements

This project was influenced by Professor Kurt Fendt who taught the MIT class Extending the Museum (CMS.636).

9 Tools, Packages, and Libraries

Package/Tool/Library (name and version number):	NEXT.js (13.3.1)
What machine and OS version did you run it on (so people will know roughly what compute power and what environment it needs):	Mac M1 (but should be compatible with any OS)
Where is it available? (eg url):	https://nextjs.org/
What did you use it to do? (One word or phrase is fine if the answer is the obvious, eg speech recognition, face detection, etc., otherwise explain a little more):	We used it to handle routing for our full stack web system.
What was the data type of the input (eg image as a matrix, mp3 format for sound, etc.):	No input, this was used for web hosting.
How well did it work in terms of accuracy (as in rough percentage correct, in your experience, no need to run a formal evaluation), and speed (as in, fast enough to allow convenient use, or it slowed down the system):	This framework was efficient for spinning up a web server quickly.
Did it work out of the box? If not, what did you have to do to use it?	Yes, worked out of the box.

Package/Tool/Library (name and version number):	P5.js (1.6.0)
What machine and OS version did you run it on (so people will know roughly what compute power and what environment it needs):	Mac M1 (but should be compatible with any OS)
Where is it available? (eg url):	https://p5js.org/
What did you use it to do? (One word or phrase is fine if the answer is the obvious, eg speech recognition, face detection, etc., otherwise explain a little more):	Rendering the circle for user's pointing feedback on top of the artwork image and rendering the heatmap of dots for user comments
What was the data type of the input (eg image as a matrix, mp3 format for sound, etc.):	It was (x, y) coordinate points
How well did it work in terms of accuracy (as in rough percentage correct, in your experience, no need to run a formal evaluation), and speed (as in, fast enough to allow convenient use, or it slowed down the system):	This was a rendering package. The rendering was simple and easy to use for the scope of our project.
Did it work out of the box? If not, what did you have to do to use it?	Yes, worked out of the box.

Package/Tool/Library (name and version number):	MediaPipe Tasks Vision (0.1.0-alpha-12)
What machine and OS version did you run it on (so people will know roughly what compute power and what environment it needs):	Mac M1 (but should be compatible with any OS)
Where is it available? (eg url):	https://www.npmjs.com/package/@mediapipe/tasks-vision
What did you use it to do? (One word or phrase is fine if the answer is the obvious, eg speech recognition, face detection, etc., otherwise explain a little more):	We used it for hand landmark detection
What was the data type of the input (eg image as a matrix, mp3 format for sound, etc.):	Image frames (taken one at a time from webcam stream) and associated timestamps
How well did it work in terms of accuracy (as in rough percentage correct, in your experience, no need to run a formal evaluation), and speed (as in, fast enough to allow convenient use, or it slowed down the system):	Very fast – worked in real time, and no visible processing delay. It had very good accuracy (maybe 95%), and when there was only one hand in the frame, it almost worked perfectly.
Did it work out of the box? If not, what did you have to do to use it?	Yes, just had to npm installl. Also had to download model from Google that detects hand landmarks to store on our own server.

Package/Tool/Library (name and version number):	Web Speech API (built-in to browser)
What machine and OS version did you run it on (so people will know roughly what compute power and what environment it needs):	Mac M1 (but should be compatible with any OS)
Where is it available? (eg url):	https://developer.mozilla.org/en-US/docs/Web/API/Web_Speech_API
What did you use it to do? (One word or phrase is fine if the answer is the obvious, eg speech recognition, face detection, etc., otherwise explain a little more):	Speech synthesis and recognition
What was the data type of the input (eg image as a matrix, mp3 format for sound, etc.):	For the speech synthesis the input is the text to synthesize. For the speech recognition the input is the audio
How well did it work in terms of accuracy (as in rough percentage correct, in your experience, no need to run a formal evaluation), and speed (as in, fast enough to allow convenient use, or it slowed down the system):	This worked well for a first version. If the user was in a quiet space, it worked well. However, outside conversation was also picked up.
Did it work out of the box? If not, what did you have to do to use it?	This worked well out of the box. It is native in modern browsers. You can check browser compatibility at https://developer.mozilla.org/en-US/docs/Web/API/SpeechRecognition#browser_compatibility.

Package/Tool/Library (name and version number):	Prisma (4.13.0)
What machine and OS version did you run it on (so people will know roughly what compute power and what environment it needs):	Mac M1 (but should be compatible with any OS)
Where is it available? (eg url):	https://www.prisma.io/
What did you use it to do? (One word or phrase is fine if the answer is the obvious, eg speech recognition, face detection, etc., otherwise explain a little more):	Store the comments and the location at which they were added
What was the data type of the input (eg image as a matrix, mp3 format for sound, etc.):	This library was used as a database. The inputs to the database were always comments with (x,y) location data and text comment data.
How well did it work in terms of accuracy (as in rough percentage correct, in your experience, no need to run a formal evaluation), and speed (as in, fast enough to allow convenient use, or it slowed down the system):	This database library perfectly suited our needs.
Did it work out of the box? If not, what did you have to do to use it?	This worked well out of the box. We had to make a data type of Comment to store in the database.