

**General Engineering Department**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

SoundSense

**To Create an Interactive Virtual Reality Visualizer for Synesthesia**

**Version 2.0**

**Project Team Members:** Nuzhat Bhuiyan, Sachin Iyer, Bridget Kegelman, and Izabella Orozco

**1. INTRODUCTION**

1.1 Purpose of Project

1.2 Background of the Experiment

**2. Requirements**

2.1 Physical Components

2.2 Software Components

**3. Procedures**

3.1 Physical Construction

3.2 Software Setup

3.3 Software Troubleshooting

**4. Milestone and Final Product Requirements**

4.1 Benchmark A Requirements

4.2 Benchmark B Requirements

4.3 Final Submission Requirements

4.6 Human Resources and Training

**5. Results**

5.1 Benchmark A Results

5.2 Benchmark B Results

5.3 Difficulties Experienced

**6. Conclusion**

6.1 Results of Project

6.2 Future Improvements

**1. INTRODUCTION**

1.1 Purpose of Project

SoundSense by Wave is a virtual reality experience that replicates the synesthesia experience where the synesthete can see sound. Wave has created an Interactive Virtual Reality Visualizer for Synesthesia that takes the loudness and pitch recognition of an audio input and converts it into a digital display. The display is in the form of a sphere and can change size or color depending on movements made throughout the audio input.

1.2 Background of the Experiment

Synesthesia is a neurological condition where pitch can trigger a multitude of senses like color or taste. This project narrows in on the experience of interpreting sounds into color. It offers an insight to experience a new representation of sound. Sound to sight synesthesia is one of the most common and well known versions. Some examples that show synesthesia is the Bowie VR Exhibit which shows digital art and the Apple visualizer which shows music streaming. The David Bowie is an AR/VR app that was made to show a visual richness of music. We want this project to be adaptable as a display, to be included in music shows, and to show synesthesia in virtual reality exhibits.



Figure 1: VR Bowie Exhibit taken from “digitalbodies.net”

**2. Requirements**

2.1 Physical Components

The physical component of this project is a VR headset.The 3D printed Virtual Reality headset was made to fit a phone in the slot on the top to experience the synesthetic visualizer.



Figure 2: VR Headset

2.2 Software Components

The software components of this project were the visualizer, the website, and the Raspberry Pi Desktop OS. The microcontroller used in this project is Raspberry Pi Desktop OS. This microcontroller was chosen because it was free to download, it allows people to develop programs on Java, Scratch, or Python, and it has the ability to browse the internet.

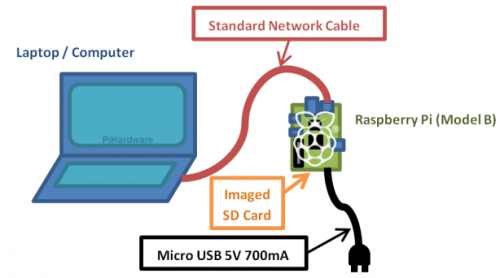


Figure 3: Raspberry Pi Wiring Diagram taken from “raspberrypi.org”

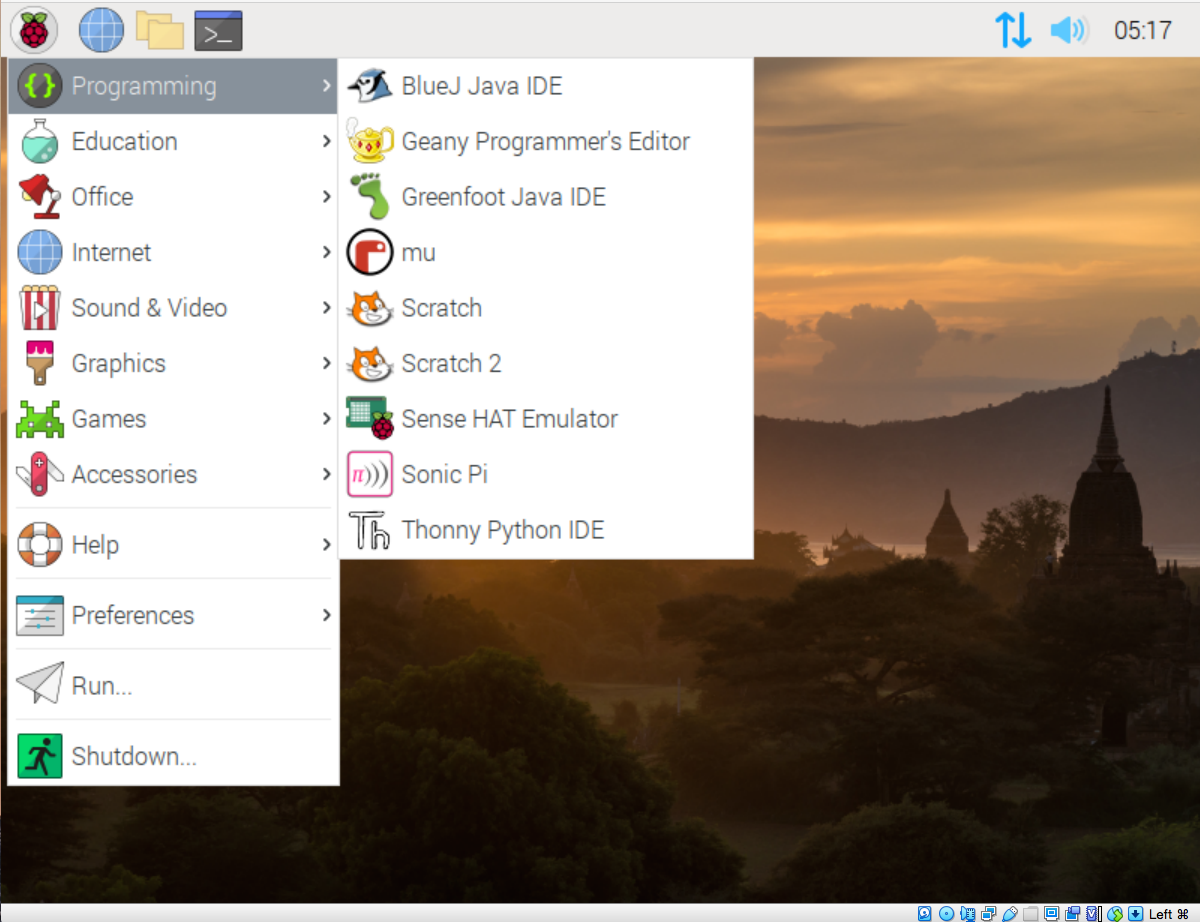


Figure 4: Raspberry Pi Desktop on Computer

**3. Procedures**

3.1 Physical Construction

The logo for the company was created in the shape of an eye with a play button in the center and 3D printed onto the Virtual Reality headset. The headset has a spot to place a phone where the visualizer can be opened and turned into a Virtual Reality experience. The file was then submitted to the EG1003 Lab for 3D printing.

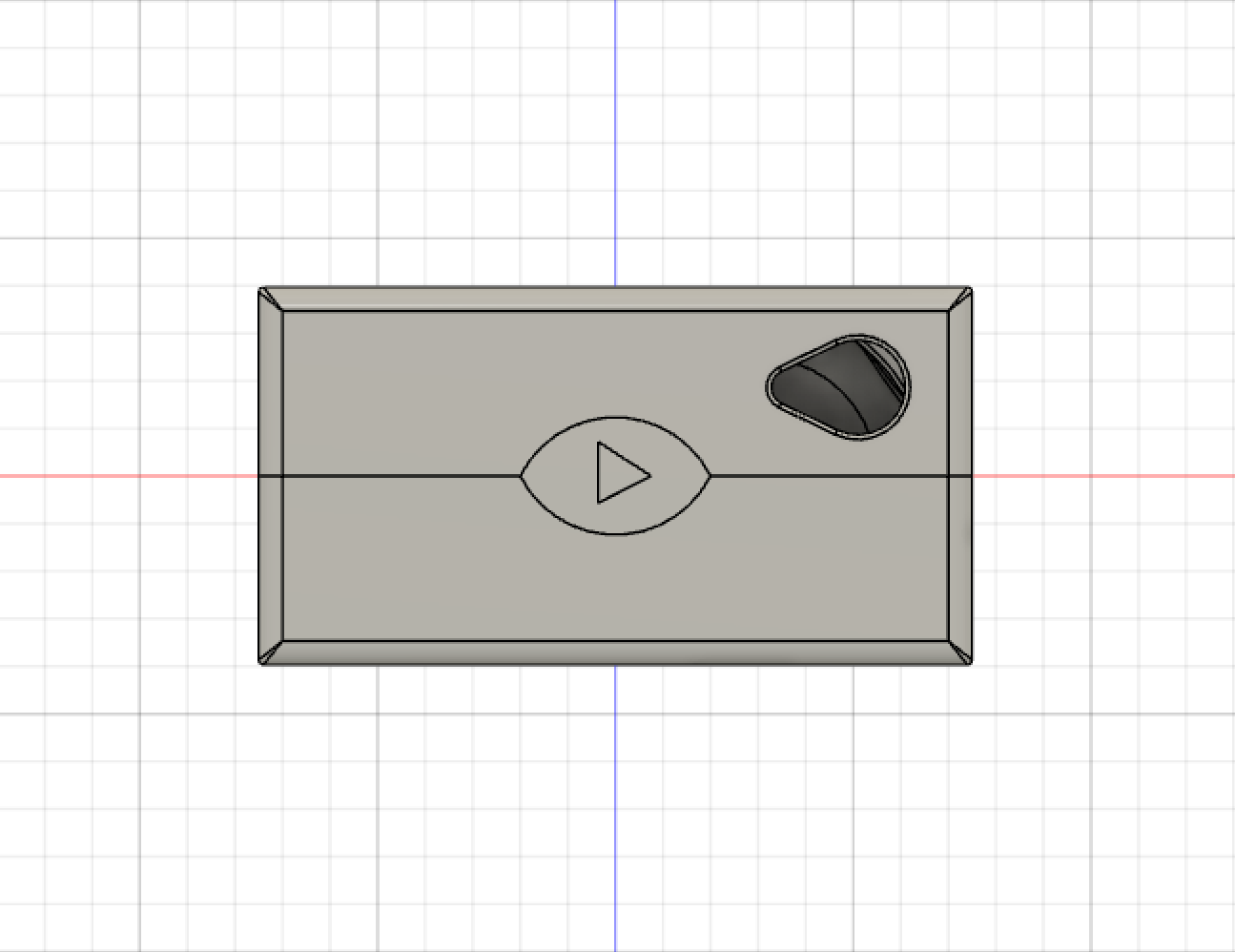


Figure 5: VR Headset (Front View)

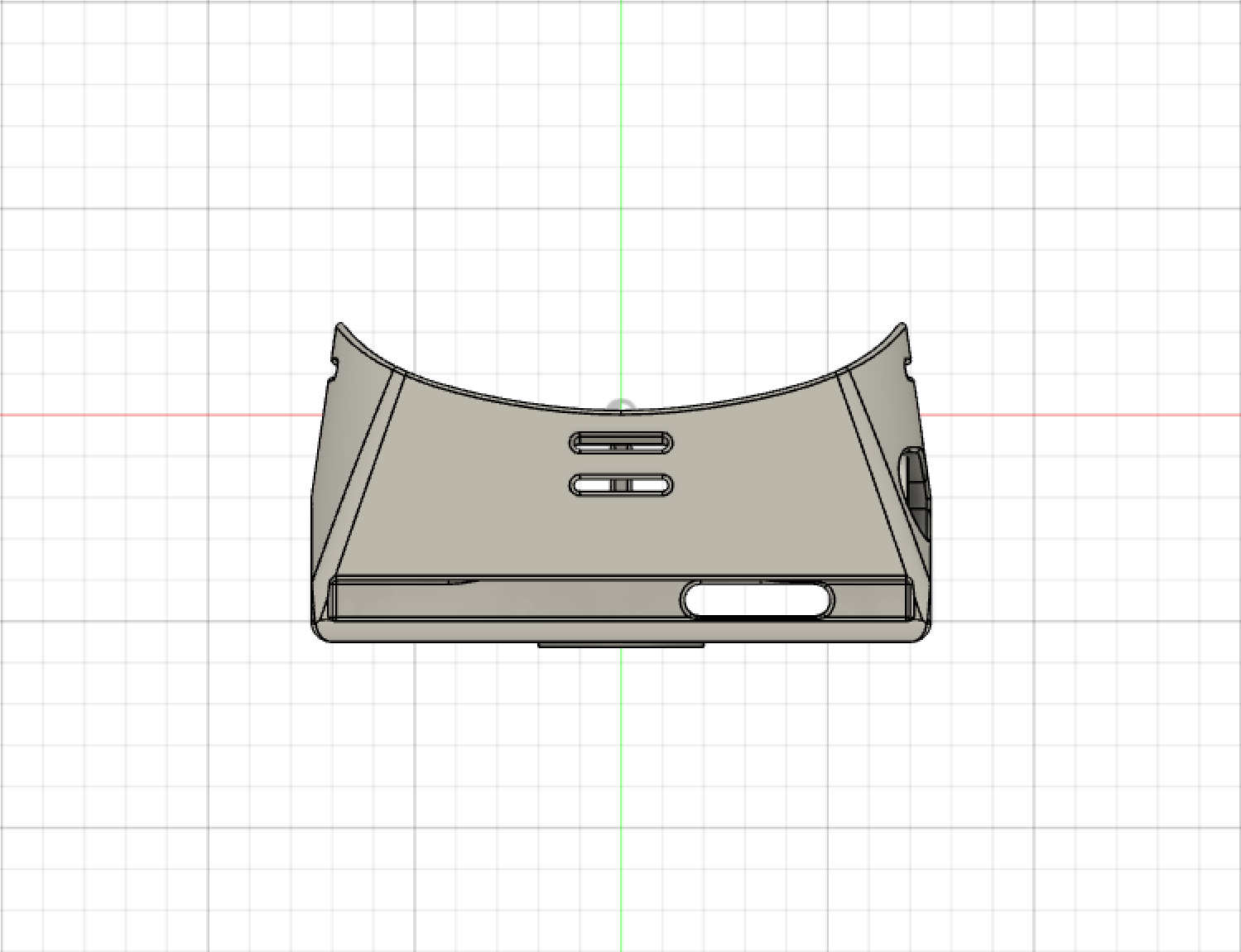


Figure 6: VR Headset (Top View)

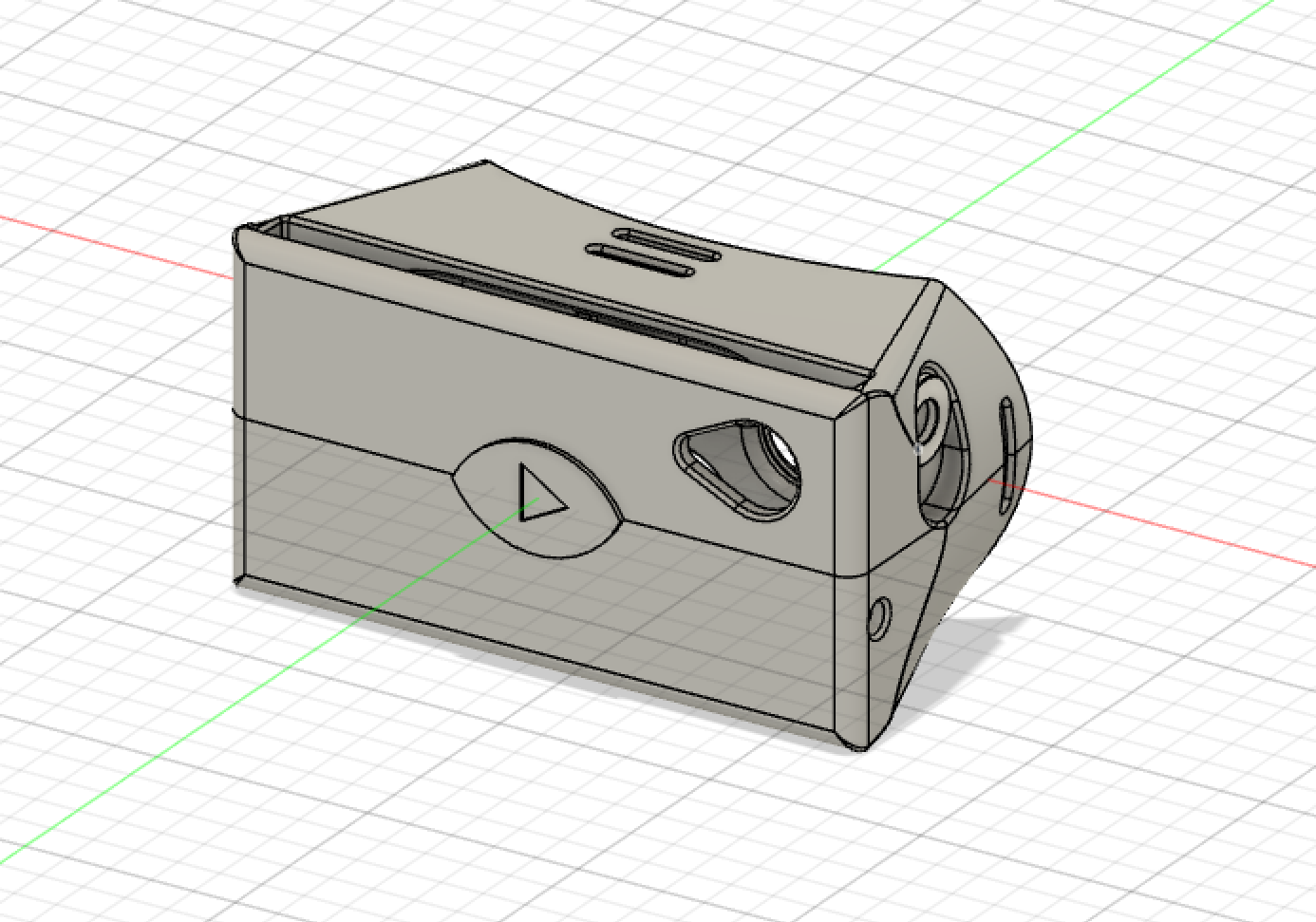


Figure 7: VR Headset (Isometric Front View)

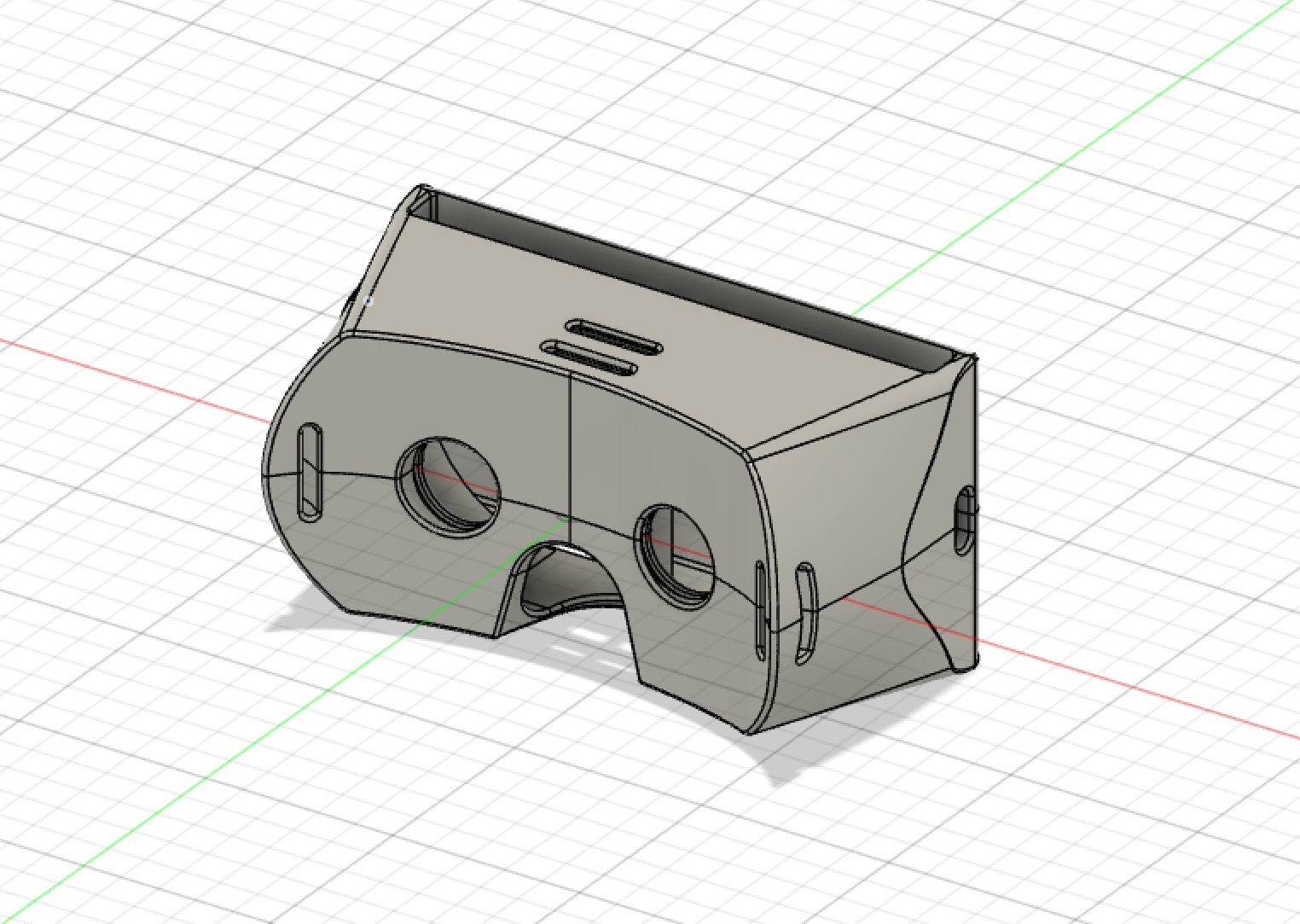


Figure 8: VR Headset (Isometric Back View)

3.2 Software Setup

In order to install Raspberry Pi Desktop, we must first download the Raspberry Pi Desktop ISO file from raspberrypi.org. While the ISO file is downloading, we must also download the correct platform package of VirtualBox from virtualbox.org. Once these files are downloaded, open VirtualBox. Select New and a window will pop up. Name the operating system to Raspberry Pi. Set the Type to Linux. Set the Version to Other Linux (64-bit). Select next and it shows a window for memory size. Set the memory size between 1000 MB and 1500 MB. Keep clicking next until the window closes; the options are already set to what we need for Raspberry Pi Desktop to work. Select Settings and click on Storage. Select Empty and click on the disk next to the Optical Drive. Choose the Raspberry Pi Desktop ISO file. Exit the Settings and select Start on the first window. The screen then shows seven options; choose Install. Configure your keyboard, select the partitioning method as “Guided - use entire disk”, and keep clicking enter until it asks to write the changes to the disk. Select Yes. Wait for the system to install. It then asks to install the GRUB boot loader. Select Yes and then select Enter device manually. Select Continue after this. Raspberry Pi Desktop is now functioning. Following the instructions before using it.

Table 1: Final Budget

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Resources** | **Cost per Unit** | **Quantity(M1)** | **Cost**  **(M1)** | **Quantity (M2)** | **Cost**  **(M2)** | **Quantity (M3)** | **Cost (M3)** |
| **Cloud Costs** | $30 | - | - | - | - | 1 | $30 |
| **Projected Labor** | $50 | 20 | $1000 | 40 | $2000 | 60 | $3000 |
| **Total** |  |  | $1000 |  | $2000 |  | $3030 |

Because there were not any physical components besides the VR Headset, the only costs associated are the Cloud Costs.

3.3 Software Troubleshooting

One of the problems that can occur is when it sets itself to low power mode, and then the Raspberry Pi Desktop could not install successfully. The solution to this is to simply increase the memory size. The only problem is that the steps to install it must be done again.

3.4 Setting Up Cloud Infrastructure

We used API Gateway to trigger lambda functions that create an ec2 instance using a launch template. This ec2 instance is then added to an autoscaling group which is then added to a kubernetes cluster. Then a subdomain is attached to the ec2 instance and that is sent over.

3.5 Client Side Visualizer

We are using BabylonJS to render graphics that have a microphone input that is sent to a server side flask application that processes the audio with librosa.

3.6

|  |  |
| --- | --- |
| FROM tiangolo/uwsgi-nginx:python3.8 |  |
|  |  |
|  | LABEL maintainer="Sebastian Ramirez <tiangolo@gmail.com>" |
|  |  |
|  | RUN pip install pip --upgrade |
|  | RUN pip install flask |
|  | RUN pip install flask-socketio |
|  | RUN pip install eventlet |
|  | RUN pip install librosa |
|  | RUN apt update |
|  | RUN apt -y upgrade |
|  | RUN apt install -y libsndfile1 |
|  | RUN apt install -y letsencrypt |
|  | RUN pip3 install pip --upgrade |
|  | COPY default.conf /etc/nginx/conf.d/default.conf |
|  |  |
|  |  |
|  | # URL under which static (not modified by Python) files will be requested |
|  | # They will be served by Nginx directly, without being handled by uWSGI |
|  | ENV STATIC\_URL /static |
|  | # Absolute path in where the static files wil be |
|  | ENV STATIC\_PATH /app/static |
|  |  |
|  | # If STATIC\_INDEX is 1, serve / with /static/index.html directly (or the static URL configured) |
|  | # ENV STATIC\_INDEX 1 |
|  | ENV STATIC\_INDEX 0 |
|  |  |
|  | # Add demo app |
|  | COPY ./app /app |
|  | COPY ./ssl\_cert/wildcard /wildcard |
|  | WORKDIR /app |
|  |  |
|  | # Make /app/\* available to be imported by Python globally to better support several use cases like Alembic migrations. |
|  | ENV PYTHONPATH=/app |
|  |  |
|  | # Move the base entrypoint to reuse it |
|  | RUN mv /entrypoint.sh /uwsgi-nginx-entrypoint.sh |
|  |  |
|  | # Copy the entrypoint that will generate Nginx additional configs |
|  | COPY entrypoint.sh /entrypoint.sh |
|  | COPY ./ssl\_cert/nginx.conf /etc/nginx/nginx.conf |
|  | RUN chmod +x /entrypoint.sh |
|  |  |
|  | ENTRYPOINT ["/entrypoint.sh"] |
|  |  |
|  | # Run the start script provided by the parent image tiangolo/uwsgi-nginx. |
|  | # It will check for an /app/prestart.sh script (e.g. for migrations) |
|  | # And then will start Supervisor, which in turn will start Nginx and uWSGI |
|  | CMD ["/start.sh"] |

**4. Milestone and Final Product Requirements**

4.1 Benchmark A Requirements

* Develop a concrete understanding of Librosa and be able to explain it
* Create a flowchart of the product functions
* Create an algorithm to import songs from the Million Songs database from Google Cloud to Librosa
* Create an algorithm to synthesize the data into an acceptable form to input to Librosa
* Present the PDI and the engineering notebook

4.2 Benchmark B Requirements

* The requirements for Benchmark B were to
* Create a design for the sphere that would be used in the visualizer
* Create and submit the file for the 3D printed Virtual Reality headset
* Demonstrate an output given microphone input

4.3 Final Submission Requirements

* Be able to get a visual output using the spherical tool created based on an input from a microphone.
* Extra credit: Setup a Raspberry Pi and display to show the visual output based on microphone input.
* Extra Credit: Create a landing page for the visualizer

4.6 Human Resources and Training

Before Benchmark A, a meeting was set up with Professor R. Luke Dubois, the Associate Professor of Integrated Digital Media at NYU. In that meeting, possible pitch recognition resources were discussed long with possible resources to create the visualizer. Professor DuBoois also demonstrated some sample work he had created with a voice recognition software with video, and a livetime voice recognition code he had created and taught about in his class.

**5. Results**

5.1 Benchmark A Results

The requirements were all met for Benchmark A. The flowchart for the visualizer was created showing the process in which the music or sound would be interpreted and how it would be converted into an input for the visualizer. Librosa was explored and some tutorials were done to learn it at a better level, and the algorithm to import songs from the Million Song Database was created. The algorithm to synthesize the data into an acceptable form to input to Librosa was also created. The PDI and Engineering Notebook was presented to the TAs.

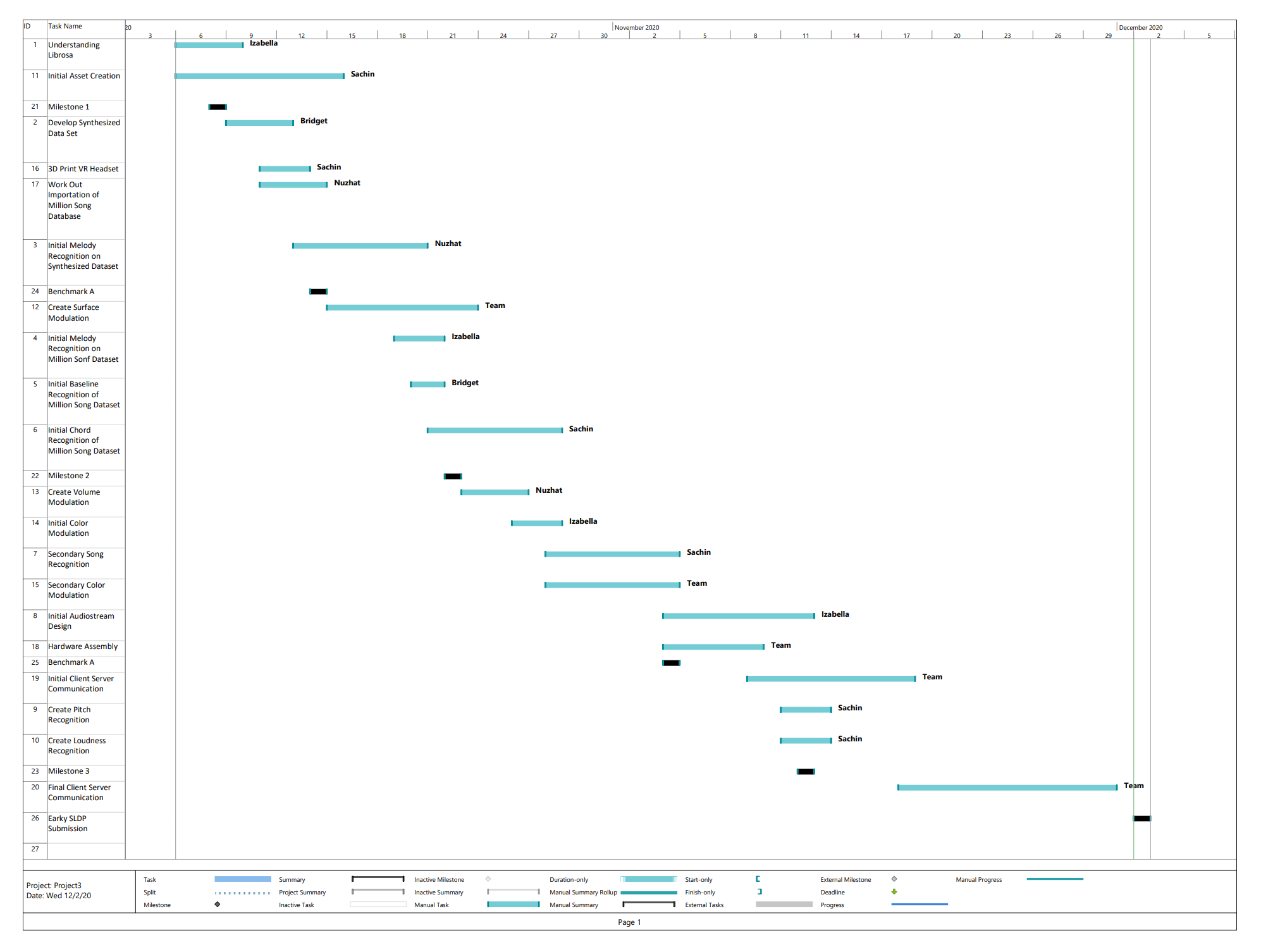


Figure 9: Project Schedule

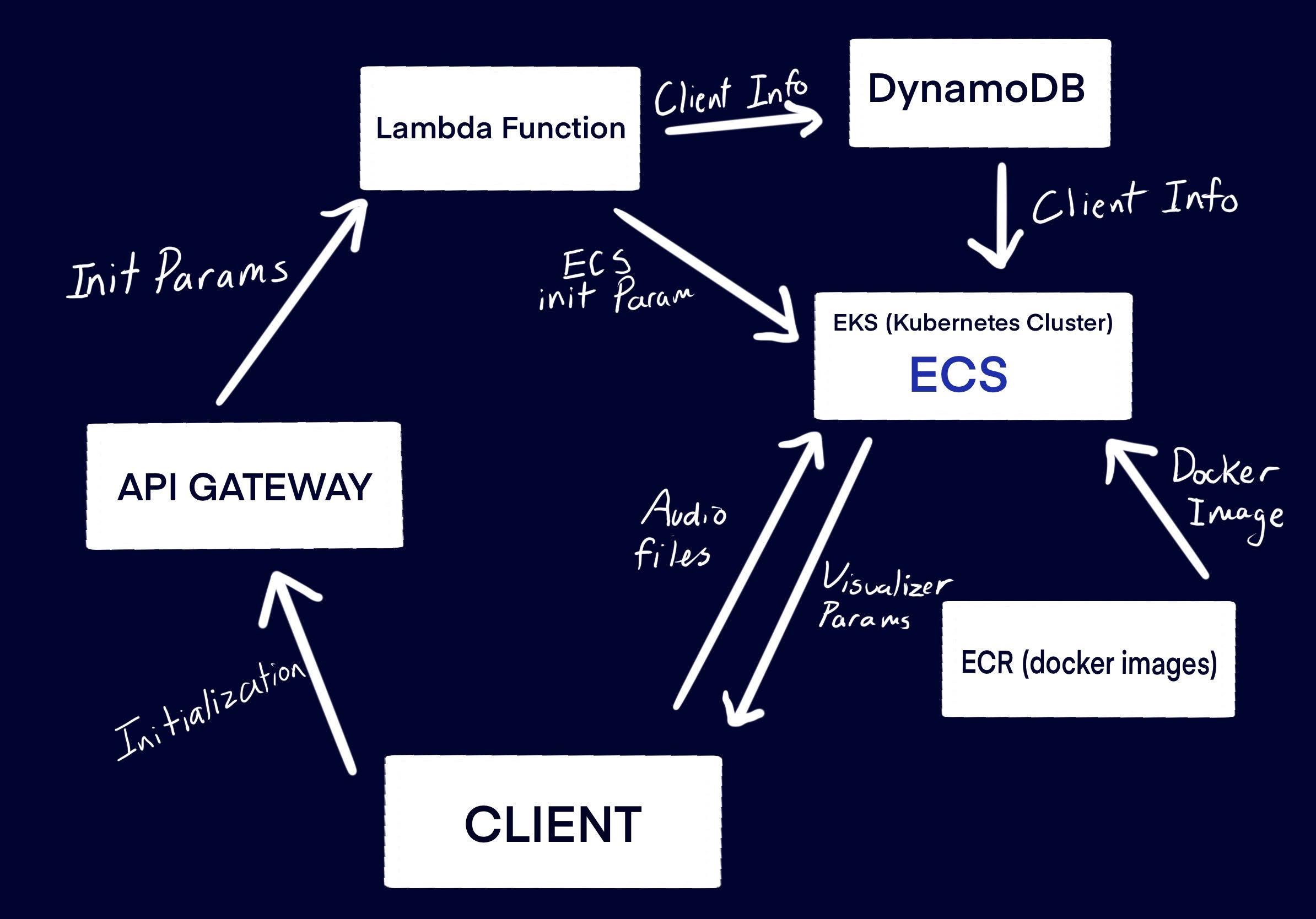


Figure 10: Flowchart

5.2 Benchmark B Results

The requirements for Benchmark B were met. The visualizer was designed to be a sphere that changes forms depending on the sound input. Its color would change according to the pitch and the texture of the surface of the sphere would change according to the loudness of the sound/music. The logo was designed and placed on the VR headset, and the file was submitted to be printed. The output was tested and demonstrated for a given microphone input.

5.3 Difficulties Experienced

We had a lot of difficulties in managing the web apis, and making sure that everything was up to code. There were some major changes in audio transfer in the middle of our project that required ssl certificates to be added to our containers and this took a lot of time.

We also had a lot of difficulties in actually creating a lot of our web infrastructure reliably.

**6. Conclusion**

6.1 Results of Project

We were able to create all of our cloud infrastructure, which includes creating generative and reactive allocation of resources that has a very low static cost. This includes creating an API Gateway that calls lambda functions that then create our ec2 instances that are then added to our autoscaling groups to be added to the kubernetes cluster and finally have subdomains attached to them. We also created nginx docker images that incorporate flask, wsgi, and socket.io websockets in order to transfer the audio from client to server and transfer the data back. Then we used BabylonJS to reactively create the graphics for the project. In this project we learned a lot about cloud infrastructure, web development, and basic audio processing.

6.2 Future Improvements

With the product fully developed, there are many possibilities of how it can be modified to serve different purposes. One way in which SoundSense can be utilized is to use it in a Virtual Reality exhibit in which the museum plays music in a room full of monitors or screens that display the visualizer around the experiencer. Another possibility would be to use the visualizer in an orchestra or a concert. The live music would be playing in the background while the audience wears the VR headset and experiences a live show depending on the music.