DEEP DIVE ON CAPSTONE

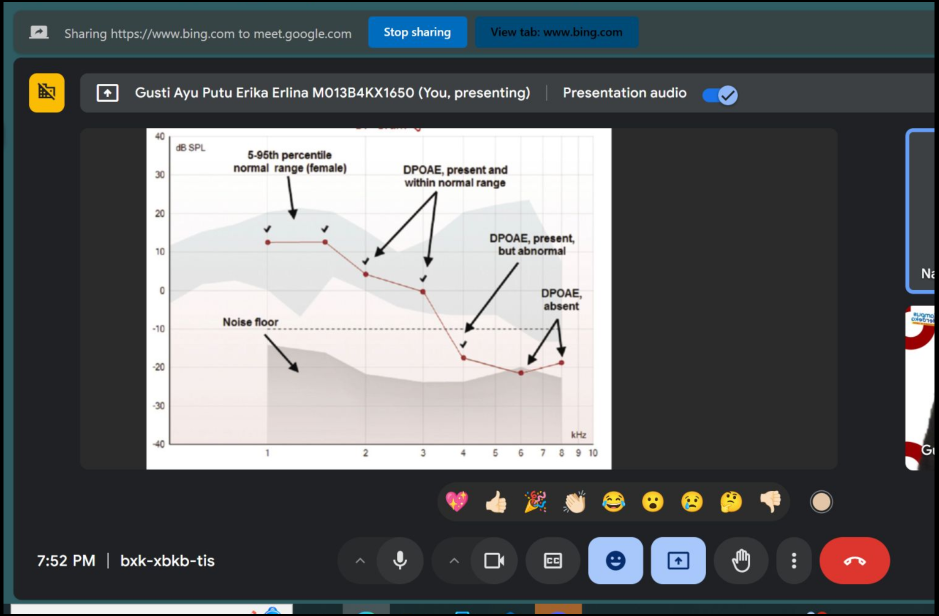
Hearity

C242-PS043

Hearity is an app focused on preventing hearing loss, primarily targeting workers in noisy environments or heavy machinery sectors, where companies provide hearing tests.

Our preventive approach includes daily forecasting based on the latest test results. Hearing loss occurs gradually over time, so our app generates daily audiograms that track hearing deterioration across frequencies. This helps raise awareness among workers about the importance of using safety hearing protocols. Additionally, our app offers features to educate workers and assist those who have already experienced hearing impairment and i will talk about this in the end of the slide.

VALIDATE DATA


Sharing https://www.bing.com to meet.google.com
Stop sharing
View tab: www.bing.com
Gusti Ayu Putu Erika Erlina M013B4KX1650 (You, presenting)
Presentation audio
dB SPL
5-95th percentile normal range (female)
DPOAE, present and within normal range
DPOAE, present, but abnormal
DPOAE, absent
Noise floor
kHz
7:52 PM
bxk-xbkb-tis

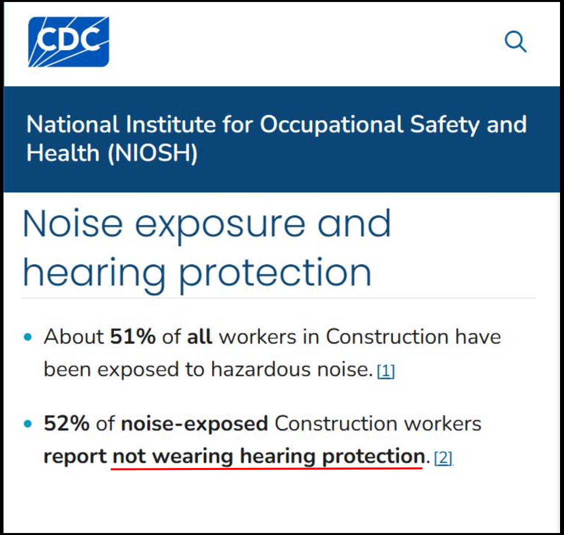
Data Validation Context:

Due to my limited work experience in this field, data validation was through insights from an audiologist who has conducted hearing tests for workers in noisy industrial environments.

Key insights from the meeting:

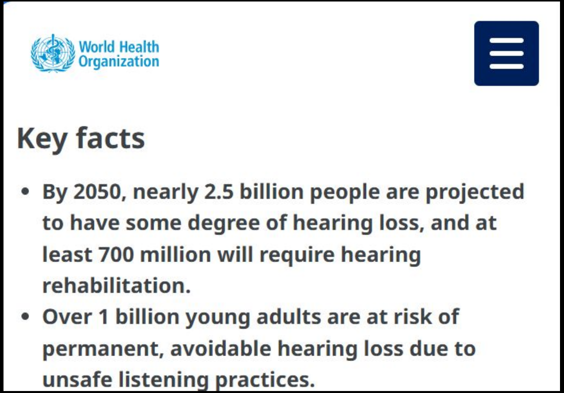
1. Hearing tests are performed as part of collaboration between companies and local hospitals.
2. Test intervals: every 6 months or 1 year for a test.
3. Hearing loss due to workplace noise is typically evident in the second test, especially at 4 kHz frequency.
4. Many workers neglect the use of hearing protection (earmuff, earbud, etc.).
5. Hearing deterioration occurs gradually over time, not as a sudden spike.
6. Prolonged exposure to noise levels exceeding 85 dB for approximately 8 hours daily or almost daily.
7. The data used consists of air conduction (AC) pure tone tests from audiometry without masking or white noise.

SUPPORTING EVIDENCE

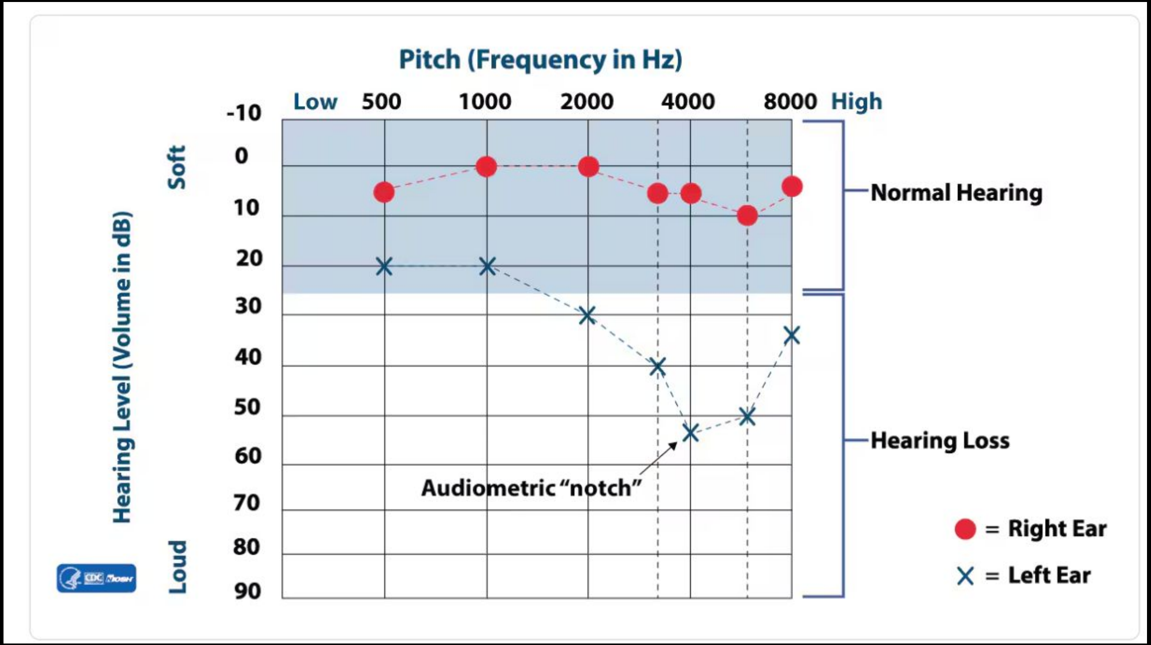

National Institute for Occupational Safety and Health (NIOSH)

Noise exposure and hearing protection

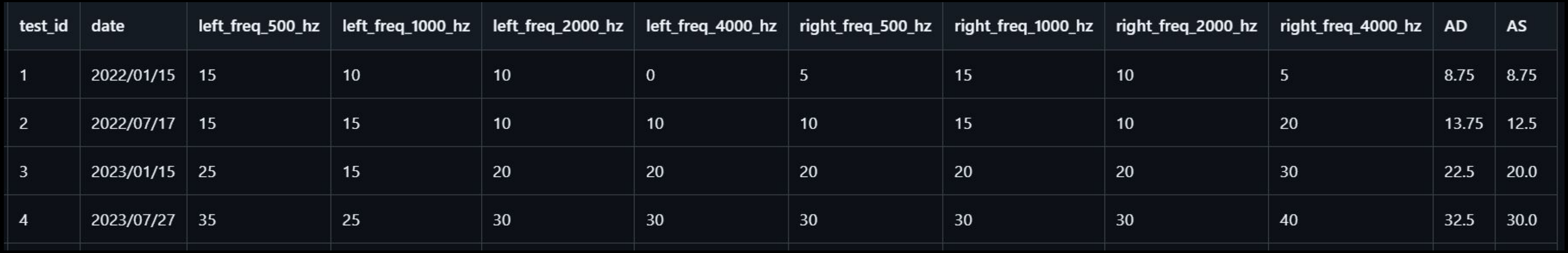
- About **51%** of **all** workers in Construction have been exposed to hazardous noise. [1]
- **52%** of **noise-exposed** Construction workers **report not wearing hearing protection**. [2]


Key facts

- **By 2050, nearly 2.5 billion people are projected to have some degree of hearing loss, and at least 700 million will require hearing rehabilitation.**
- **Over 1 billion young adults are at risk of permanent, avoidable hearing loss due to unsafe listening practices.**



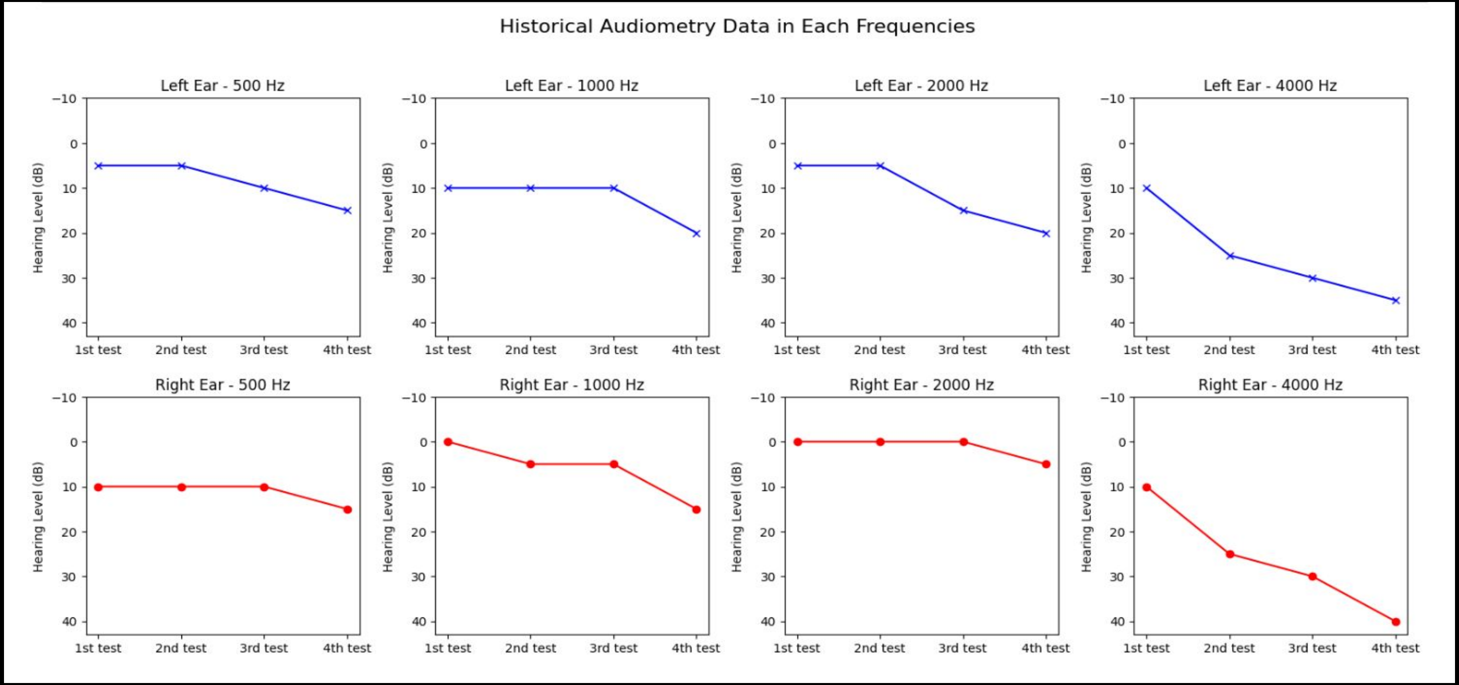
DUMMY DATA



test_id	date	left_freq_500_hz	left_freq_1000_hz	left_freq_2000_hz	left_freq_4000_hz	right_freq_500_hz	right_freq_1000_hz	right_freq_2000_hz	right_freq_4000_hz	AD	AS
1	2022/01/15	15	10	10	0	5	15	10	5	8.75	8.75
2	2022/07/17	15	15	10	10	10	15	10	20	13.75	12.5
3	2023/01/15	25	15	20	20	20	20	20	30	22.5	20.0
4	2023/07/27	35	25	30	30	30	30	30	40	32.5	30.0

Our hearing loss dataset tracks factory workers exposed to noise levels over 85 dB during 8-hour shifts. Initial tests included workers with normal hearing (0-25 dB) and slight impairment (26-40 dB) from prior exposure. Over 2 years, with tests every 6 months, consistent noise and lack of ear protection caused progressive hearing loss, particularly at 4 kHz, visible by the fourth test in both ears (AD and AS). Workers with prior damage experienced faster decline, underscoring the critical need for hearing safety in industrial settings.

Machine Learning



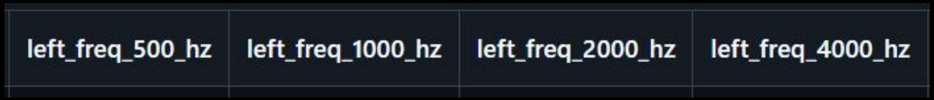
We use time series forecasting for machine learning, leveraging historical data that depends on time for monitoring purposes. Since Bangkit requires the use of TensorFlow, we chose CNN over other time series approaches like RNN, LSTM, and GRU because the data we are processing has simple patterns without seasonality, focusing mainly on trends and time. Additionally, the dataset only about 500-600 data points.

We specifically focus on the 1D convolutional layer in CNN because it excels at extracting relevant features from sequential data by applying filters along the time axis, making it well-suited for time series forecasting. This layer enables the model to efficiently capture patterns and trends over time without the need for complex structures, ensuring fast training and effective performance with our relatively small dataset.

FEATURE SELECTION

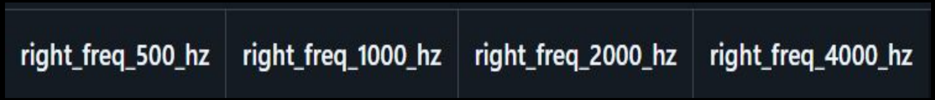
In the audiometry test, each ear is tested at 4 frequencies, resulting in 4 features for the left ear and 4 features for the right ear. This means the model will predict 8 outputs in total—4 for the left ear and 4 for the right ear. Additionally, the date column will be selected to facilitate the resampling process (using upsampling techniques). The patient_id column will also be included to ensure that the process is separated for each patient.

4 features in left ear



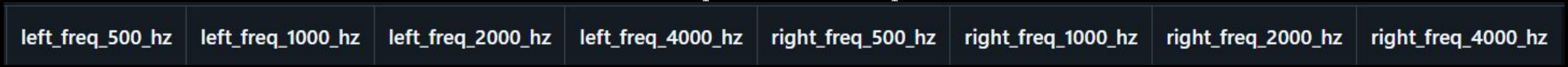
left_freq_500_hz	left_freq_1000_hz	left_freq_2000_hz	left_freq_4000_hz

4 features in right ear



right_freq_500_hz	right_freq_1000_hz	right_freq_2000_hz	right_freq_4000_hz

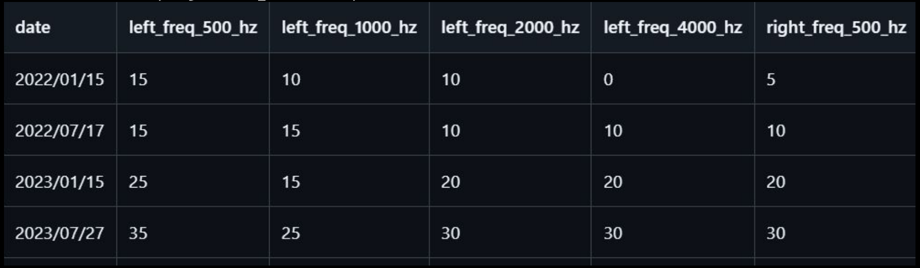
So, the model will take 8 features as input and the output will also consist of the same 8 features.



left_freq_500_hz	left_freq_1000_hz	left_freq_2000_hz	left_freq_4000_hz	right_freq_500_hz	right_freq_1000_hz	right_freq_2000_hz	right_freq_4000_hz

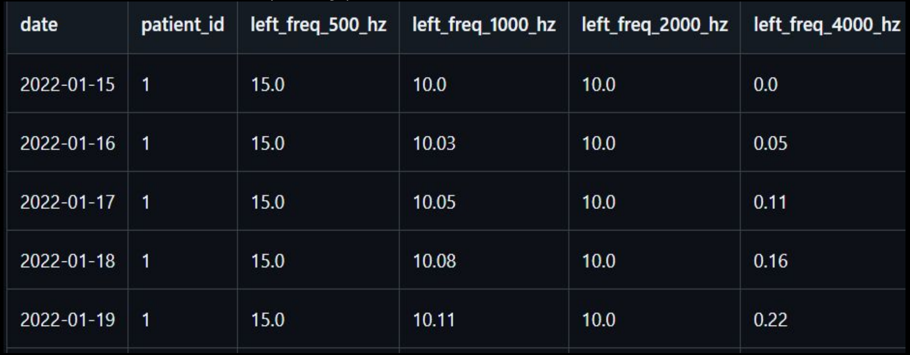
PREPROCESSING

raw data (2 year period)

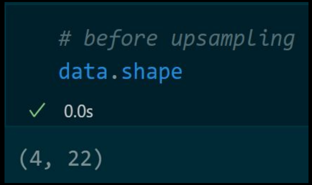


date	left_freq_500_hz	left_freq_1000_hz	left_freq_2000_hz	left_freq_4000_hz	right_freq_500_hz
2022/01/15	15	10	10	0	5
2022/07/17	15	15	10	10	10
2023/01/15	25	15	20	20	20
2023/07/27	35	25	30	30	30

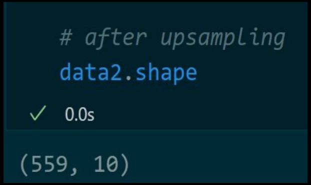
transformed data (daily)



date	patient_id	left_freq_500_hz	left_freq_1000_hz	left_freq_2000_hz	left_freq_4000_hz
2022-01-15	1	15.0	10.0	10.0	0.0
2022-01-16	1	15.0	10.03	10.0	0.05
2022-01-17	1	15.0	10.05	10.0	0.11
2022-01-18	1	15.0	10.08	10.0	0.16
2022-01-19	1	15.0	10.11	10.0	0.22

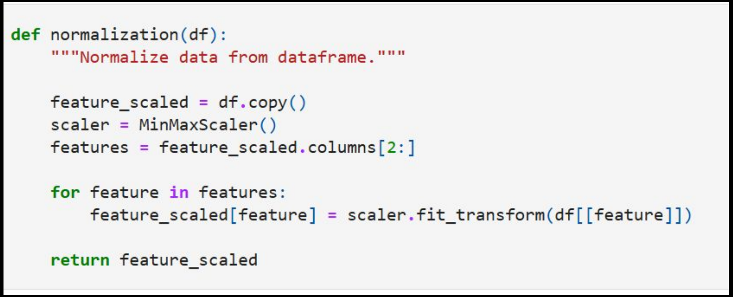


```
# before upsampling
data.shape
✓ 0.0s
(4, 22)
```



```
# after upsampling
data2.shape
✓ 0.0s
(559, 10)
```

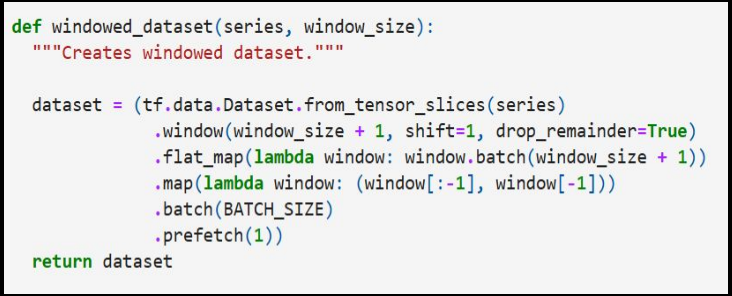
The dataset consists of four data points, each representing a test conducted every six months over a two-year period. To make this limited data more suitable for machine learning, the dataset need to get resampling and the type of it is upsampling, generating new daily data points through interpolation between the existing six-monthly values. This approach enriches the dataset with more granular details, allowing for better trend analysis and making the data more applicable for machine learning modeling, despite its initial sparsity.



```
def normalization(df):
    """Normalize data from dataframe."""

    feature_scaled = df.copy()
    scaler = MinMaxScaler()
    features = feature_scaled.columns[2:]

    for feature in features:
        feature_scaled[feature] = scaler.fit_transform(df[[feature]])

    return feature_scaled
```

Because the model uses eight features with distinct patterns and scales, each feature will be normalized individually to ensure equal contribution to the model's learning process. Normalization is crucial because the 4 kHz frequency in both ears shows significant spikes compared to other frequencies. This process prevents features with larger scales from disproportionately influencing the results. By normalizing each feature separately, the model can better capture the unique patterns of each feature.

The windowed_dataset function prepares a time series for supervised learning by creating input-target pairs from historical data. It converts the series into a TensorFlow dataset, slices it into overlapping windows, separates each window into inputs (x) and targets (y), batches the data, and uses prefetching to optimize performance by loading one batch in advance.



```
def windowed_dataset(series, window_size):
  """Creates windowed dataset."""

  dataset = (tf.data.Dataset.from_tensor_slices(series)
             .window(window_size + 1, shift=1, drop_remainder=True)
             .flat_map(lambda window: window.batch(window_size + 1))
             .map(lambda window: (window[:-1], window[-1]))
             .batch(BATCH_SIZE)
             .prefetch(1))
  return dataset
```

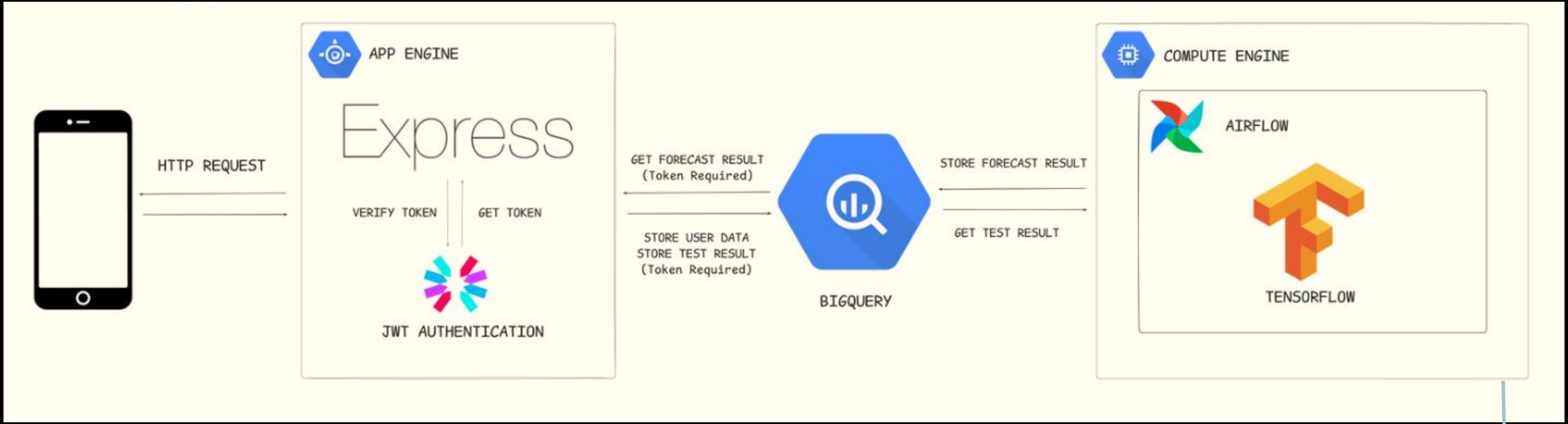
After developing the model and fine-tuning the optimizers, layers, and parameters for time series forecasting using Conv1D layers to achieve the target MSE and MAE, we collaborated with the cloud computing team to create an API with Flask. The application was containerized using Docker and deployed on the cloud platform (GCP).

Machine Learning Implementation

The approach was implemented because real audiometry data cannot be directly fed into the model due to its limited frequency (tests conducted every six months, resulting in sparse data points over several years). To enable daily forecasting, the data is upsampled to a daily frequency, enriching the dataset and aligning it with the model's required shape of (1, 30, 8).

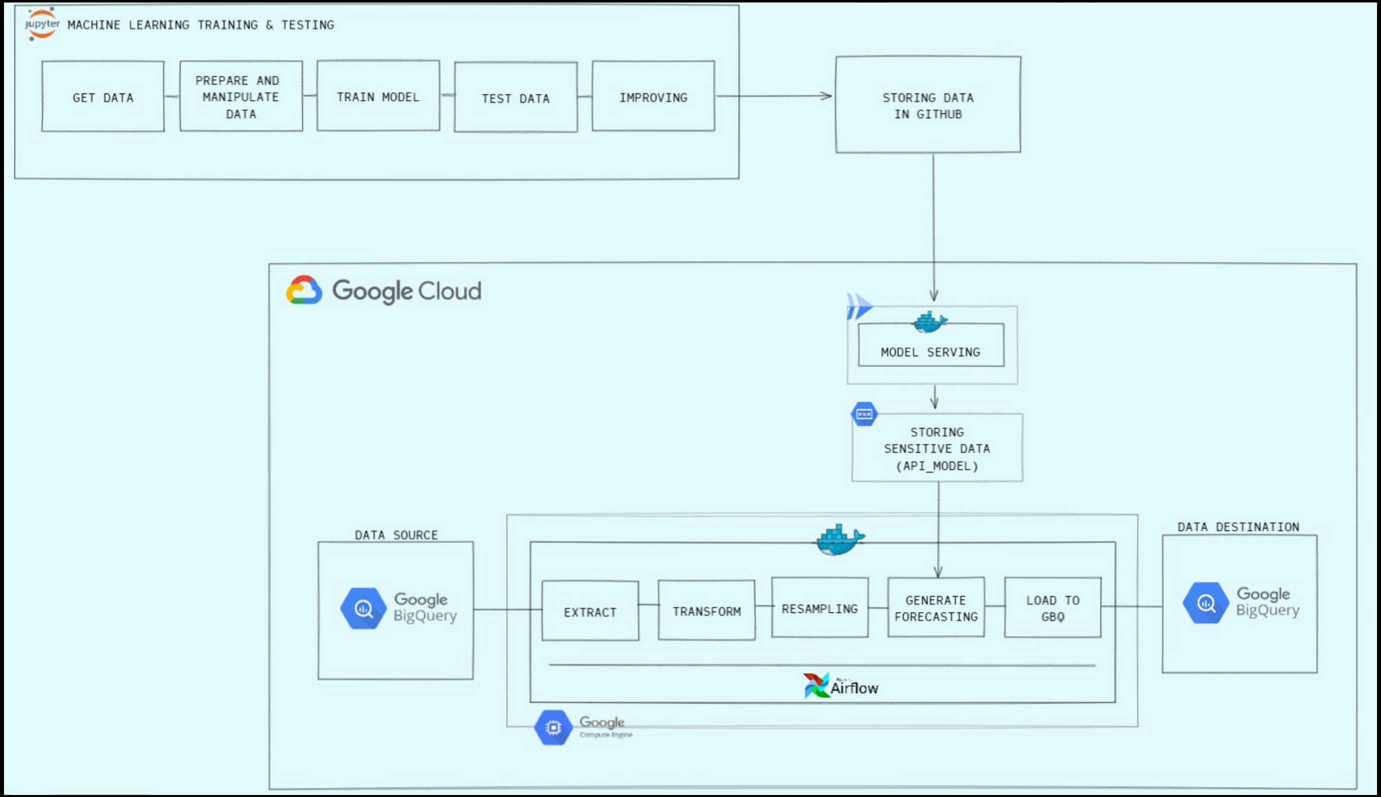
Since the forecasting type is daily (not real-time), each user only requires one output per day which displayed as an audiogram in the app. This justifies the use of batch processing, with the process scheduled to run daily at 00:00 GMT+7 to suit the local (Indonesia-only for now) user base. The forecasting workflow is executed in the cloud, where stored data is extracted, reshaped to fit the model, generate forecasting, and then reloaded into the storage. Given the $100 credit limit, the solution is implemented on a VM using Apache Airflow to optimize resources and minimize costs.

SYSTEM PIPELINE



This is our team's system pipeline, but this slide will focus only on the ML and its implementation, so I'll cover this part.

DATA PIPELINE



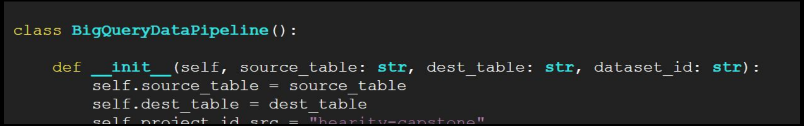
With permission from Bangkit, I confidently carry out the tasks in the cloud, as my work closely aligns with the machine learning (forecasting) flow. Due to characteristic of the data, the tasks involve repeating preprocessing approaches used in ML on local systems. My primary focus is on ensuring that raw data is preprocessed and shaped correctly to meet the model's input requirements. Using Python, numpy, and pandas, I handle data preparation and incorporate forecasting.

Again and again, why scheduled? We’ve received so many questions about this

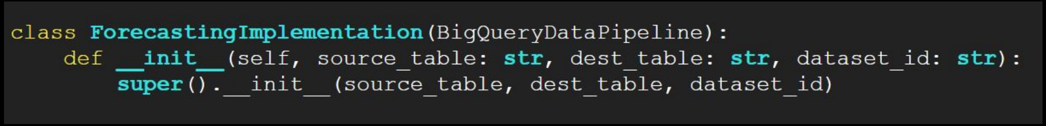
Our primary focus is on workers in heavy machinery environments and the companies providing those services, where most users are tested on the same day, and the results are inputted into the app immediately after testing. The concept is simple: the test day is *day-0, the starting point*. Today’s test results are valid and accurately reflect the latest hearing condition after noise exposure on the d-day. So, what’s the need to predict day-0? There isn’t one.

The real concern starts tomorrow, on day-1, when noise exposure at the workplace begins to impact their hearing again, especially as habits like neglecting hearing protection continue. Hearing deterioration doesn’t happen suddenly, it occurs gradually over time. Day by day, hearing thresholds will show small but measurable changes across frequencies. This is where our app comes in: the forecasting runs daily, producing a set of predictions per user, which is then displayed in the mobile app as a 24-hour audiogram that captures real trends or spikes caused by loud noise exposure. At exactly 00:00 GMT+7, the model generates a new prediction based on the latest test and yesterday’s forecast, which bridges gaps since tests typically happen only once every six months. This ensures our forecasting reflects the actual impact of workplace noise on workers’ hearing conditions, day after day.

CLASS FOR AIRFLOW DAG



```
class BigQueryDataPipeline():

    def __init__(self, source_table: str, dest_table: str, dataset_id: str):
        self.source_table = source_table
        self.dest_table = dest_table
        self.project_id_src = "bearity-capstone"
```



```
class ForecastingImplementation(BigQueryDataPipeline):
    def __init__(self, source_table: str, dest_table: str, dataset_id: str):
        super().__init__(source_table, dest_table, dataset_id)
```

I made these classes on VM GCP and this class serves as the parent class, focusing on data extraction, transformation, and loading into a data warehouse like BigQuery. It extracts data from BigQuery, transforms it by selecting relevant features and the output as a DataFrame. The transformed data is then passed to the ForecastingImplementation class for further processing. Once the forecasting is completed, the results—consisting of eight features along with the calculated values for AD and AS based on these features—are loaded back into BigQuery. Store the forecasting results in BigQuery instead of visualizing them directly because they can serve as valuable data sources for future predictions.

The child class specializes in additional responsibilities such as resampling data, preparing it as input for a time series model, and making predictions through an API. Its primary focus is transforming the data (selected columns or features) into a format suitable for model inputs. As part of the preprocessing in ML, the raw data undergoes daily upsampling, and the timestamps are structured into 30-day windows, resulting in a shape of (1, 30, 8).

Sensitive data is securely managed using Google Secrets Manager to ensure security and the output of these are consistently stored in a DataFrame format to maintain a structured approach during the entire process. To achieve this, the class leverages super() to inherit methods from its parent (BigQueryDataPipeline class).

FUNCTION IN ForecastingImplementation CLASS

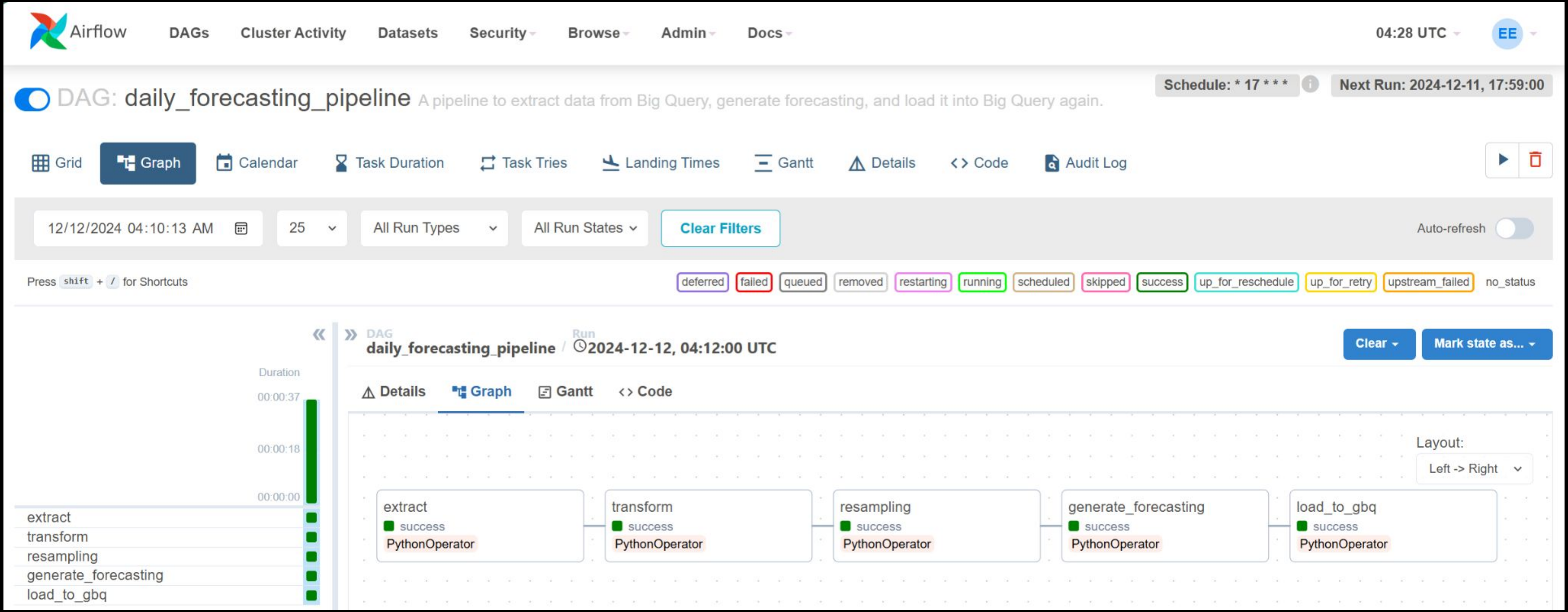
```python
def resampling_data(self, transformed_result: pd.DataFrame) -> pd.DataFrame:
    """Resampling test results for each patient."""

    df = transformed_result
    today = datetime.now(timezone.utc).date()
    df = df.sort_values(by=['user_id', 'date'])

    # get the last test per user
    last_test = df.groupby(['user_id']).tail(1)
    last_test['date'] = last_test['date'].dt.date

    if today in last_test['date'].values:
        logging.info(f'Using {self.source_table} as data source because of new audiometry test.')

        patients_today = last_test[last_test['date'] == today]['user_id']
        df = df[df['user_id'].isin(patients_today)]
        df_last_2_test = df.groupby('user_id').tail(2)
        df_last_2_test = df_last_2_test.sort_values(by=['user_id', 'date'])
        df_last_2_test['date'] = pd.to_datetime(df_last_2_test['date'])
        upsampling_data = self._upsampling(df_last_2_test)

        logging.info(f'Resampling data completed successfully.')

        return upsampling_data
```

In the resampling function, first i sort the values in ascending order to ensure they are not random for each user. Next, I retrieve the latest test from each user and compare it to today’s date and it’s always in UTC format because 00:00 GMT+7 is 17:00 UTC. If a user/patient has a test on today’s date, the data will filtered to include only those users tested today and selects the two most recent tests for each. Afterward, I sort the values again to maintain proper order, as I intend to interpolate the data into daily intervals. Before calling the upsampling function, I ensure the dates are in datetime format.

```
    else:
        logging.info(f'Using combination of {self.source_table} & {self.dest_table} tables as data sources.')

        # forecasting table
        source2 = self.extract(source='dest')
        df2 = self.transform(source2)
        df2['date'] = pd.to_datetime(df2['date'])
        test_date = df2['date'].dt.date.values

        if today in test_date:
            patients_today = df2.loc[df2['date'].dt.date == today, 'user_id']
            df_filtered = df2[~df2['user_id'].isin(patients_today)]
        else:
            df_filtered = df2

        # hearing test table
        df_last_2_test = df.groupby('user_id').tail(2)

        # concate data sources
        merged_df = pd.concat([df_last_2_test, df_filtered], ignore_index=True, axis=0)
        merged_df = merged_df.sort_values(by=['user_id', 'date'])
        merged_df['date'] = pd.to_datetime(merged_df['date'])
        upsampling_data = self._upsampling(merged_df)

        logging.info(f'Resampling data completed successfully.')

        return upsampling_data
```

The test occurs every six months, so to enable daily forecasting, I fill the gap by reusing yesterday's forecast as today's data source. Using inherited properties from the parent class, I extract and transform the data, compare the forecast date with today's date to prevent overlaps during retries and monitor model outputs. Finally, I retrieve the two most recent test results also yesterday's forecast (if it's exist), merge, sort, and upsample the data so it will make the data suitable to be model's input.

After creating the classes to handle all the processes, I wrote an Airflow DAG script in a separate file and scheduled it for 17:00 UTC, which corresponds to 00:00 AM in GMT+7. Then ran Apache Airflow in a Docker container. Once it was running smoothly, I accessed the Airflow webserver to ensure the workflow was functioning as expected.



DAG: daily_forecasting_pipeline A pipeline to extract data from Big Query, generate forecasting, and load it into Big Query again.

Schedule: * 17 * * *

Next Run: 2024-12-11, 17:59:00

12/12/2024 04:10:13 AM | 25 | All Run Types | All Run States

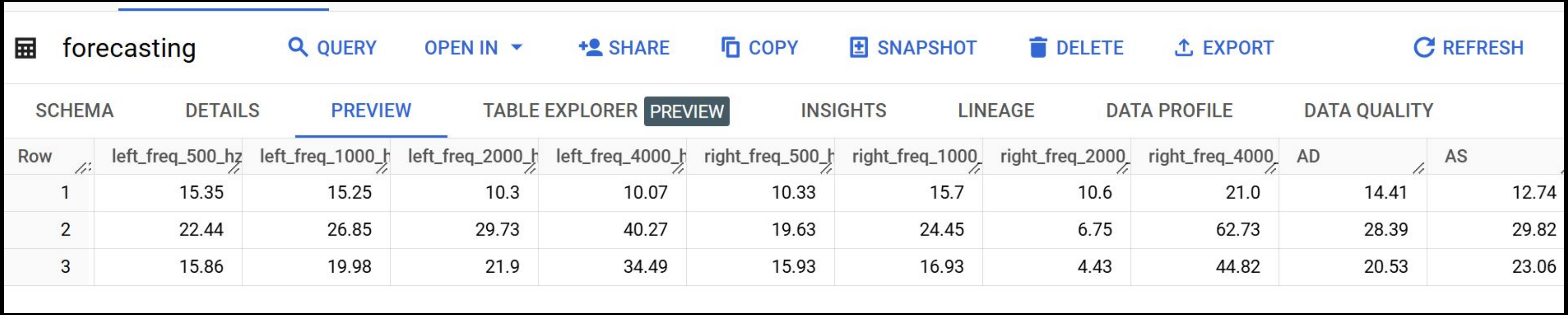
Press shift + / for Shortcuts

DAG daily_forecasting_pipeline / Run 2024-12-12, 04:12:00 UTC

Duration: 00:00:37, 00:00:18, 00:00:00

Layout: Left -> Right

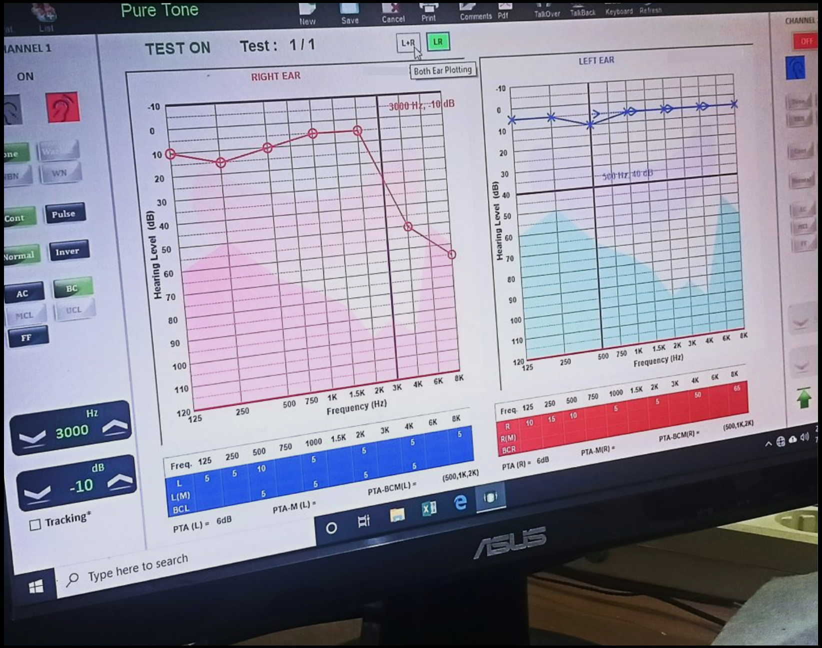
Task	State	Operator
extract	success	PythonOperator
transform	success	PythonOperator
resampling	success	PythonOperator
generate_forecasting	success	PythonOperator
load_to_gbq	success	PythonOperator

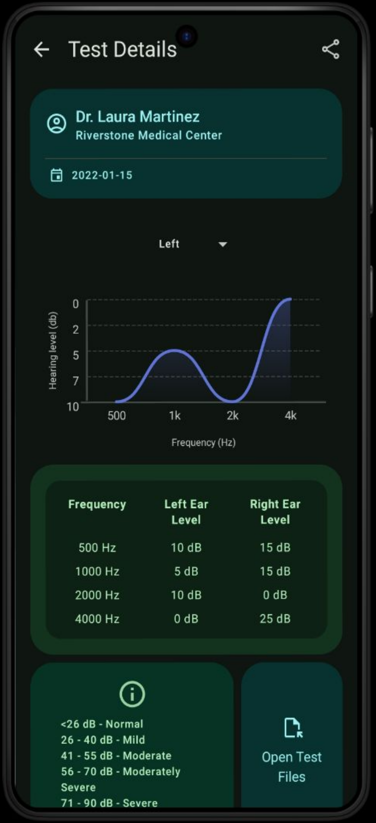


Row	left_freq_500_hz	left_freq_1000_h	left_freq_2000_h	left_freq_4000_h	right_freq_500_h	right_freq_1000_	right_freq_2000_	right_freq_4000_	AD	AS
1	15.35	15.25	10.3	10.07	10.33	15.7	10.6	21.0	14.41	12.74
2	22.44	26.85	29.73	40.27	19.63	24.45	6.75	62.73	28.39	29.82
3	15.86	19.98	21.9	34.49	15.93	16.93	4.43	44.82	20.53	23.06

After several steps, the forecast for each user is generated and stored back in BigQuery. These are previews of the data resulting from all scheduled processes running on the GCP VM (including the date, user_id, and id columns), along with the AD and AS results derived from simple calculations based on the model's output. This data then will be used for visualization by the mobile development team and is also reused for the next day's forecasting.

Comparison Between Real Test and Our App





Dr. Laura Martinez
Riverstone Medical Center

2022-01-15

Frequency	Left Ear Level	Right Ear Level
500 Hz	10 dB	15 dB
1000 Hz	5 dB	15 dB
2000 Hz	10 dB	0 dB
4000 Hz	0 dB	25 dB

<26 dB - Normal
26 - 40 dB - Mild
41 - 55 dB - Moderate
56 - 70 dB - Moderately Severe
71 - 90 dB - Severe

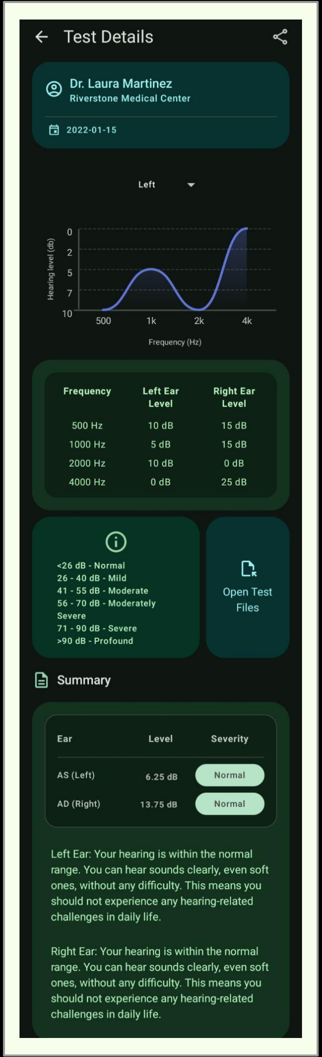
- The test results displayed are **dummy data** generated specifically for the purpose of this capstone project.
- **No actual patient data** was used or leaked, as all the data shown is **artificially created** and **does not represent real individuals**.

The audiogram is generated based on 8 features—4 for the left ear and 4 for the right ear—from the input (real test) and the model's outputs.

Since the test is conducted once every six months, the audiogram will display results from the actual test (day 0) and forecasted results for the next 180 days.

Our app provides complete severity levels based on AD and AS.

One key advantage of our application is its ability to provide a detailed hearing loss summary for each ear, unlike real tests that only show the severity. This enables more personalized and informed decision-making.



Dr. Laura Martinez
Riverstone Medical Center

2022-01-15

Frequency	Left Ear Level	Right Ear Level
500 Hz	10 dB	15 dB
1000 Hz	5 dB	15 dB
2000 Hz	10 dB	0 dB
4000 Hz	0 dB	25 dB

<26 dB - Normal
26 - 40 dB - Mild
41 - 55 dB - Moderate
56 - 70 dB - Moderately Severe
71 - 90 dB - Severe
>90 dB - Profound

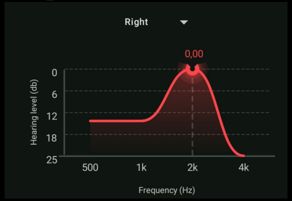
Summary

Ear	Level	Severity
AS (Left)	6.25 dB	Normal
AD (Right)	13.75 dB	Normal

Left Ear: Your hearing is within the normal range. You can hear sounds clearly, even soft ones, without any difficulty. This means you should not experience any hearing-related challenges in daily life.

Right Ear: Your hearing is within the normal range. You can hear sounds clearly, even soft ones, without any difficulty. This means you should not experience any hearing-related challenges in daily life.

dropdown for left, right, and both

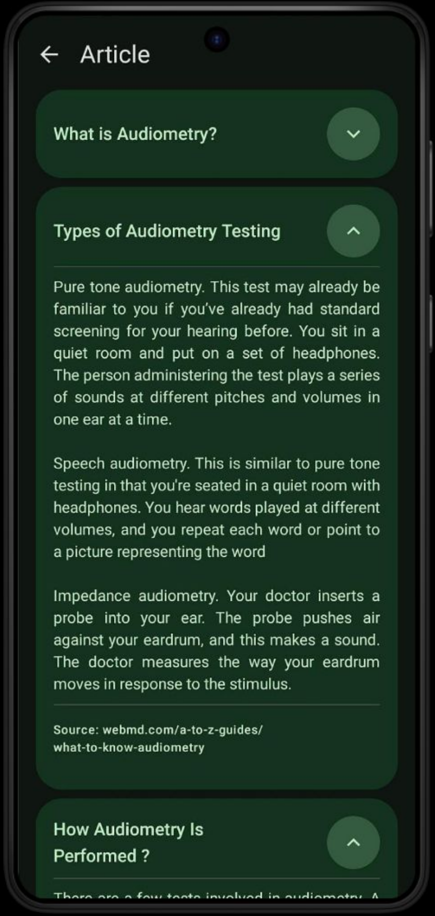


Allows users to store pictures of their test results, ensuring easy access in case they forget where they saved them.

Provides segmentation (based on AD and AS) to classify the severity level or degree of hearing for each day per user.

We also provide an interactive chatbot, but what if the user or patient doesn't have any ideas about hearing health?

To address this, we have created a special section with articles related to hearing health and educational explanations, which can help trigger the user to ask questions or clarify any confusion.



Article

What is Audiometry?

Types of Audiometry Testing

Pure tone audiometry. This test may already be familiar to you if you've already had standard screening for your hearing before. You sit in a quiet room and put on a set of headphones. The person administering the test plays a series of sounds at different pitches and volumes in one ear at a time.

Speech audiometry. This is similar to pure tone testing in that you're seated in a quiet room with headphones. You hear words played at different volumes, and you repeat each word or point to a picture representing the word

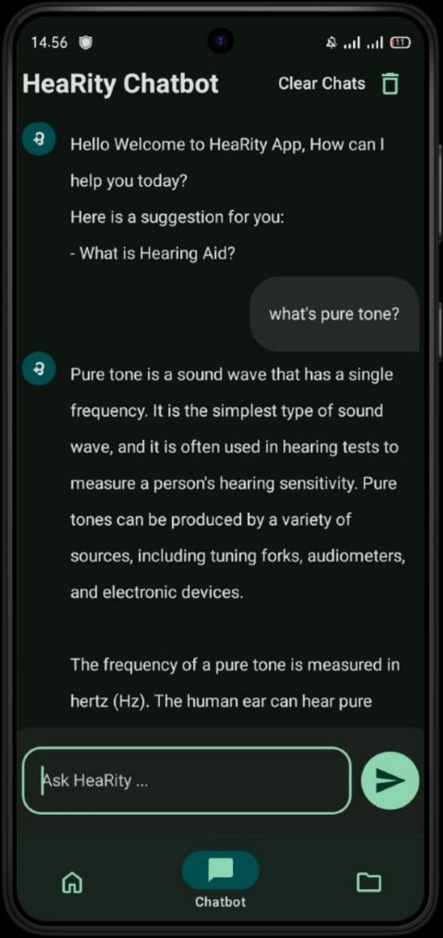
Impedance audiometry. Your doctor inserts a probe into your ear. The probe pushes air against your eardrum, and this makes a sound. The doctor measures the way your eardrum moves in response to the stimulus.

Source: webmd.com/a-to-z-guides/what-to-know-audiometry

How Audiometry Is Performed ?

If, after reading the articles, the user still has questions or doesn't fully understand the content, the chatbot is available to address their curiosity.

This interactive chatbot provides further explanations related to hearing health and offers suggestions to help guide the user in asking more specific questions, ensuring they get the information they need.


14.56

HeaRity Chatbot

Clear Chats

Hello Welcome to HeaRity App, How can I help you today?

Here is a suggestion for you:

- What is Hearing Aid?

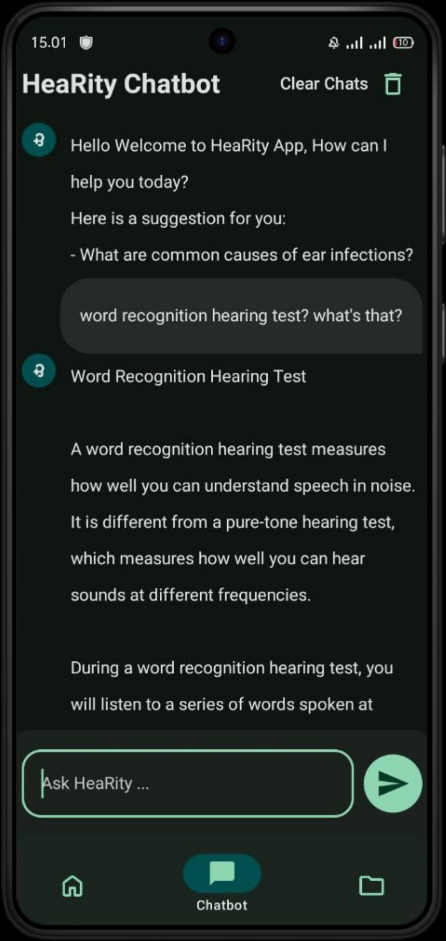
what's pure tone?

Pure tone is a sound wave that has a single frequency. It is the simplest type of sound wave, and it is often used in hearing tests to measure a person's hearing sensitivity. Pure tones can be produced by a variety of sources, including tuning forks, audiometers, and electronic devices.

The frequency of a pure tone is measured in hertz (Hz). The human ear can hear pure

Ask HeaRity ...

Chatbot


15.01

HeaRity Chatbot

Clear Chats

Hello Welcome to HeaRity App, How can I help you today?

Here is a suggestion for you:

- What are common causes of ear infections?

word recognition hearing test? what's that?

Word Recognition Hearing Test

A word recognition hearing test measures how well you can understand speech in noise. It is different from a pure-tone hearing test, which measures how well you can hear sounds at different frequencies.

During a word recognition hearing test, you will listen to a series of words spoken at

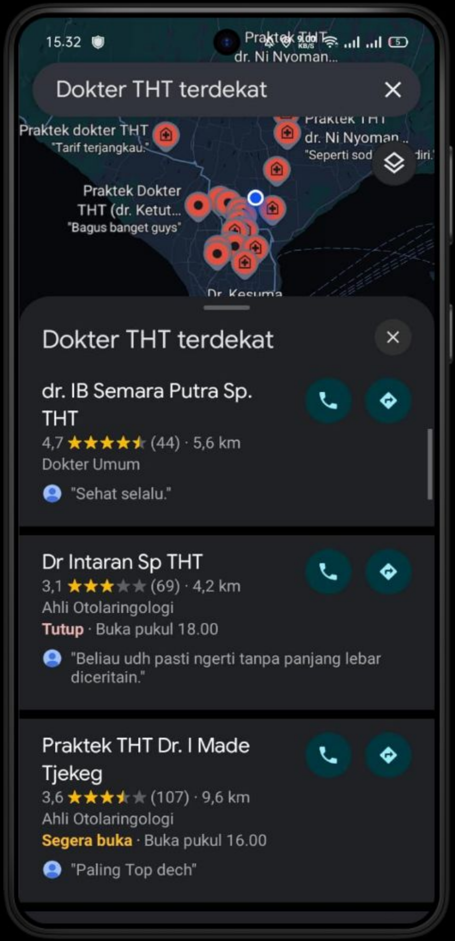
Ask HeaRity ...

Chatbot

What actions can be taken after monitoring to improve hearing health quality?

Our app includes a feature that allows users to simply click and instantly find for nearby ENT (Ear, Nose, and Throat) clinics on Google Maps. The search bar is automatically populated with relevant terms, making it easy for users to find the closest healthcare providers with no extra effort.

This feature is particularly helpful for workers in heavy machinery environments, as they often rely on company-provided tests for hearing monitoring. Since these tests are typically conducted every six months, many workers are unsure where to go for follow-up tests or consultations. Our app guides them directly to the nearest ENT clinics, ensuring they receive the care they need.

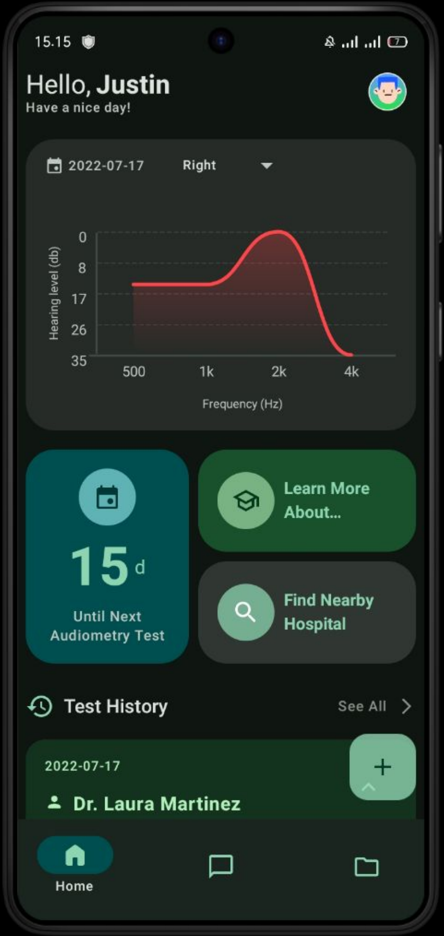


Dokter THT terdekat

dr. IB Semara Putra Sp. THT
4,7 ★★★★★ (44) · 5,6 km
Dokter Umum
"Sehat selalu."

Dr Intaran Sp THT
3,1 ★★★★★ (69) · 4,2 km
Ahli Otolaringologi
Tutup · Buka pukul 18.00
"Beliau udh pasti ngerti tanpa panjang lebar diceritain."

Praktek THT Dr. I Made Tjekeg
3,6 ★★★★★ (107) · 9,6 km
Ahli Otolaringologi
Segera buka · Buka pukul 16.00
"Paling Top dech"


15.15

Hello, Justin
Have a nice day!

2022-07-17 Right

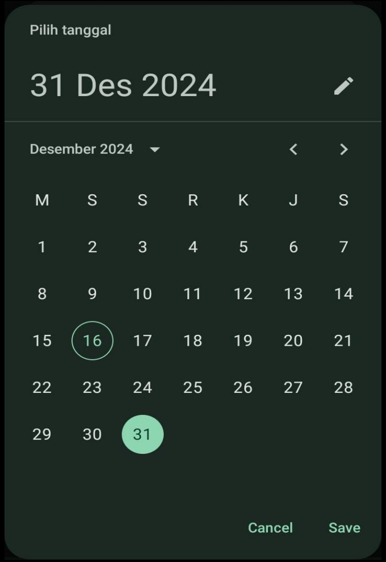
15 d
Until Next Audiometry Test

Learn More About...

Find Nearby Hospital

Test History
See All

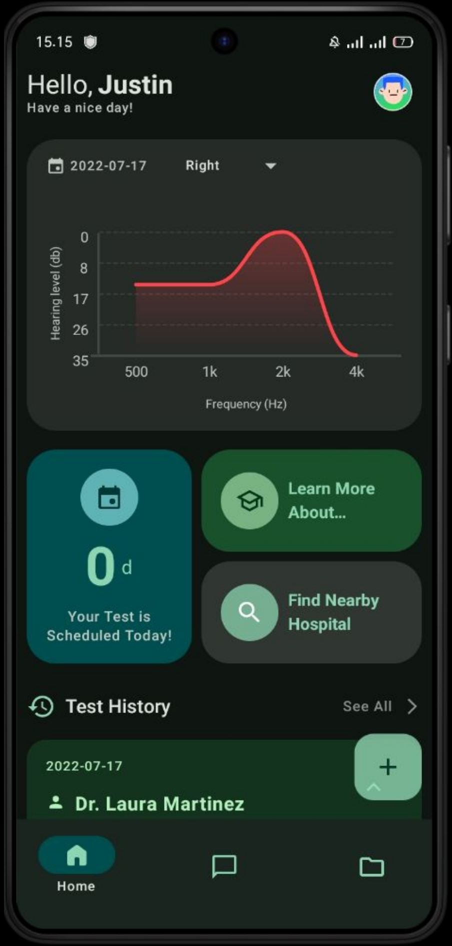
2022-07-17
Dr. Laura Martinez


Pilih tanggal

31 Des 2024

Desember 2024

M	S	S	R	K	J	S
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31				


15.15

Hello, Justin
Have a nice day!

2022-07-17 Right

0 d
Your Test is Scheduled Today!

Learn More About...

Find Nearby Hospital

Test History
See All

2022-07-17
Dr. Laura Martinez

Our app offers customizable reminders, allowing users to mark their ENT consultation date. Once booked, the reminder counts down the days until the appointment, ensuring users stay on track with their hearing health and never miss a test or consultation.