# Text-to-Speech Converter using AWS Polly and Elastic Beanstalk

Project Overview

This project demonstrates the implementation of a Text-to-Speech (TTS) web application that uses AWS Polly to convert user input text into natural-sounding speech. It is developed with Python Flask as the backend framework and deployed on AWS Elastic Beanstalk for scalability and ease of deployment.
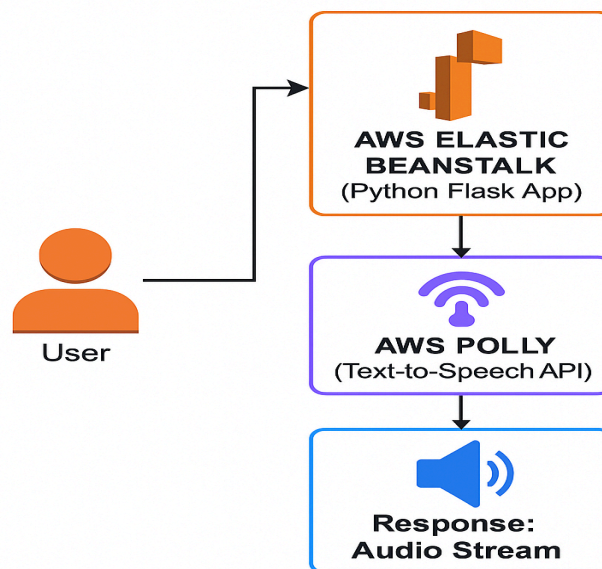
Key Features

- Text input and conversion to speech.

- Audio playback within the web interface.

- Audio download functionality.

- Progress bar for audio playback.

- Multilingual support (depending on Polly voices).

- Cloud-based deployment for ease of access and scaling.

Tech Stack

- Backend Language: Python

- Web Framework: Flask

- Text-to-Speech API: AWS Polly

- Deployment Platform: AWS Elastic Beanstalk

- Access Management: AWS IAM Roles and Policies

Use Cases

- **Accessibility** – Enables visually impaired users to hear text content.

- **Language Learning** – Assists with pronunciation and listening skills.

- **Content Creation** – Generates voiceovers for videos and blogs.

- **E-Learning** – Converts lessons and guides into spoken audio.

- **Chatbots & Assistants** – Adds natural speech to virtual agents.

- **Audiobooks** – Turns stories and articles into audio format.

- **Website Playback** – Lets users listen to web content.

- **Internal Tools** – Provides audio summaries of reports or memos.

- **Multilingual Support** – Delivers voice messages in various languages.

- **Public Announcements** – Automates clear and consistent messages.

Step-by-Step Guide

Objective

- To build an accessible and user-friendly web interface that allows users to convert text to audio using AWS Polly.
- Provide playback and download options for the generated audio.
- Ensure scalability and deployment using AWS Elastic Beanstalk for high availability and cloud-based hosting.
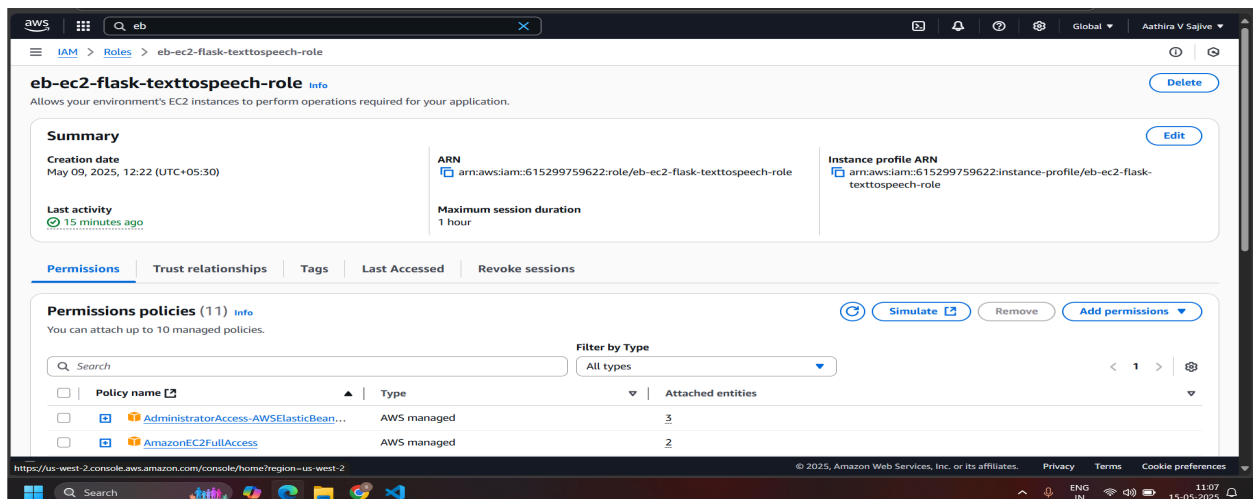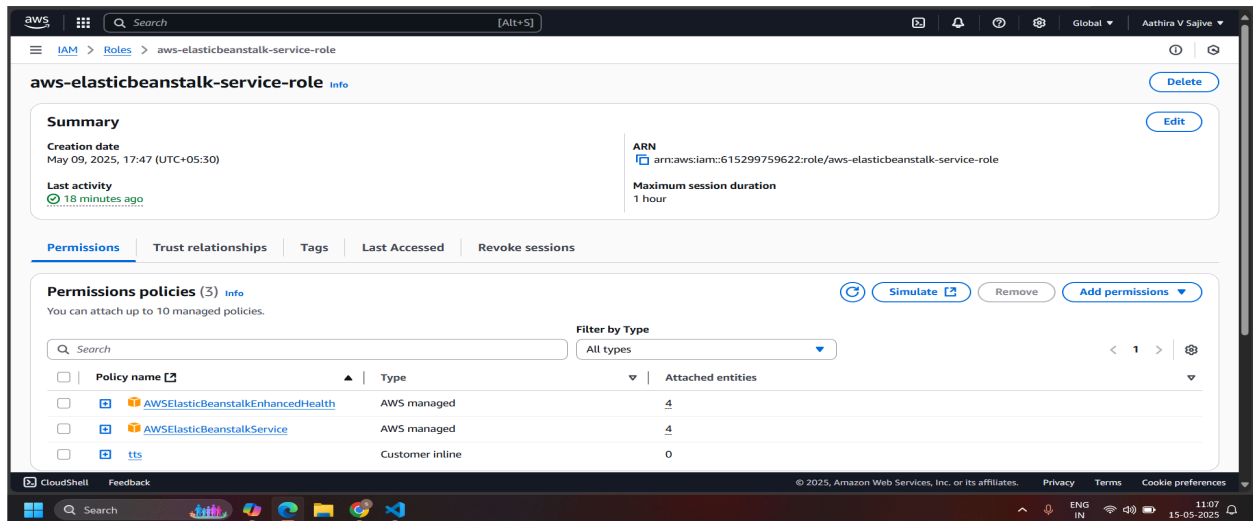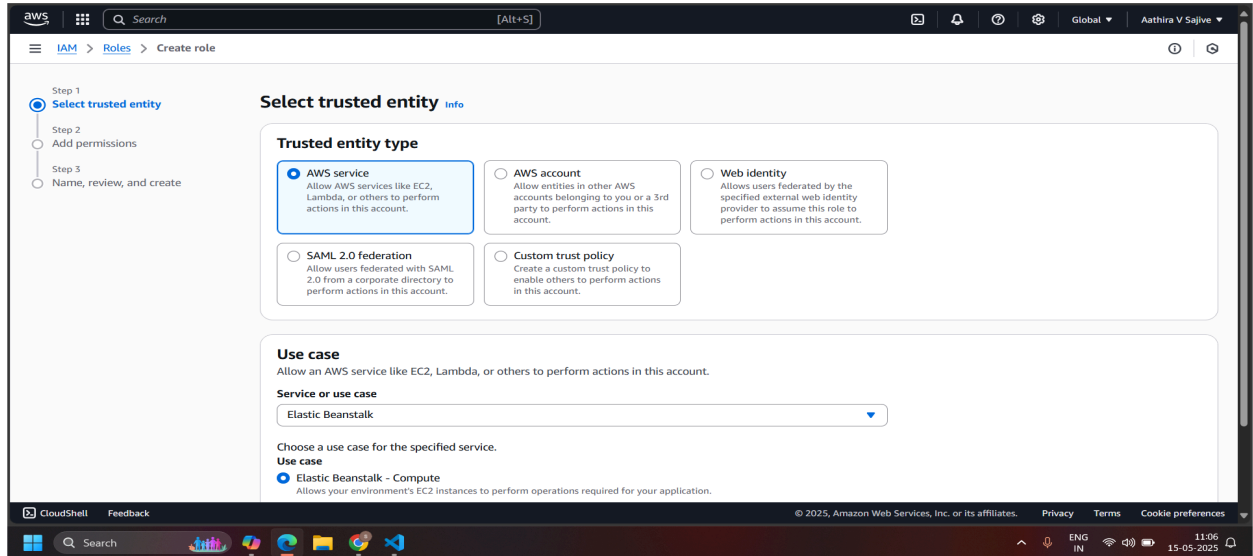
Prerequisites:

Step 1: Set Up an AWS Account
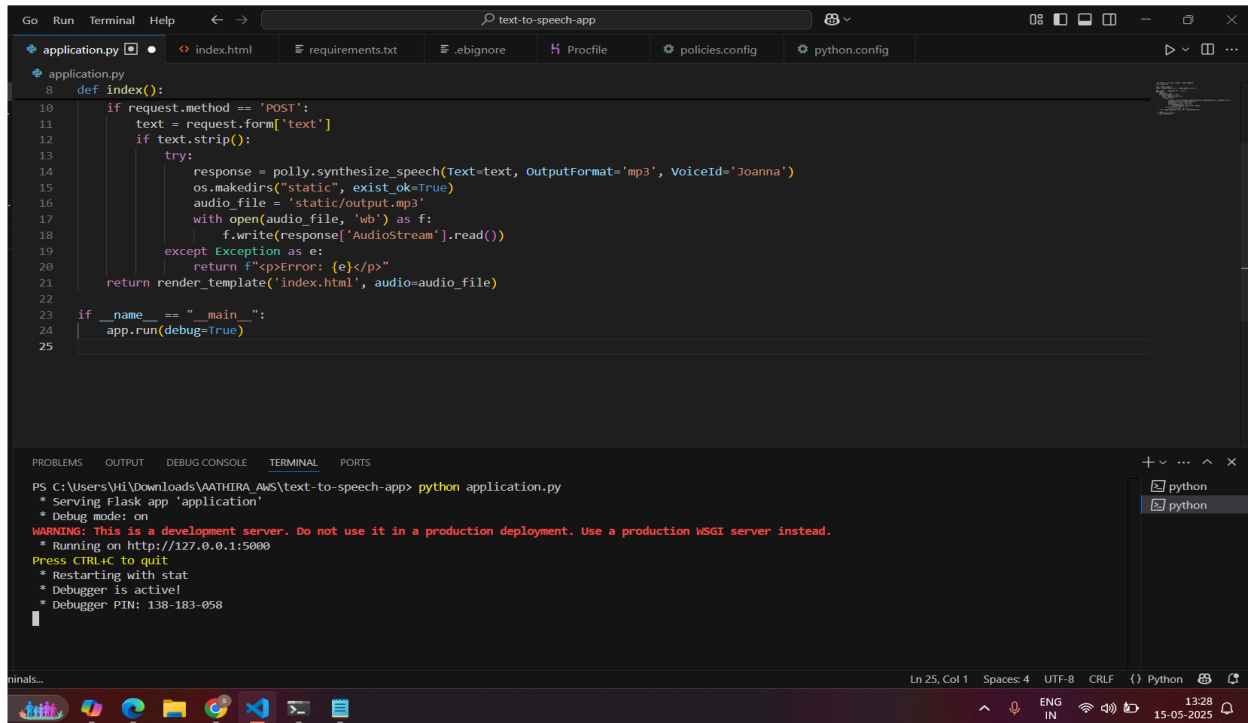
1. AWS Account: If you don't have an AWS account, sign up for a free account [here.](#)
2. Access AWS Management Console: Log in to the AWS Management Console.
3. Navigate to AWS Polly: Go to the AWS Polly console under Machine Learning. You can search for "Polly" in the AWS Management Console search bar.

Step 2: Create IAM Role and Policy

1. Go to the IAM console.
2. Click on "Roles" in the left sidebar and then "Create role."
3. Select "AWS service" as the type of trusted entity, and choose "Elastic Beanstalk" as the service that will use this role.
4. Create two separate IAM roles: one for the **Service Role** and one for the **EC2 Instance Profile**.

IAM > Roles > Create role

Step 1
Select trusted entity

Step 2
Add permissions

Step 3
Name, review, and create

Select trusted entity Info

Trusted entity type

● **AWS service**
Allow AWS services like EC2, Lambda, or others to perform actions in this account.

○ **AWS account**
Allow entities in other AWS accounts belonging to you or a 3rd party to perform actions in this account.

○ **Web identity**
Allows users federated by the specified external web identity provider to assume this role to perform actions in this account.

○ **SAML 2.0 federation**
Allow users federated with SAML 2.0 from a corporate directory to perform actions in this account.

○ **Custom trust policy**
Create a custom trust policy to enable others to perform actions in this account.

Use case
Allow an AWS service like EC2, Lambda, or others to perform actions in this account.

Service or use case
Elastic Beanstalk ▾

Choose a use case for the specified service.
Use case
● **Elastic Beanstalk - Compute**
Allows your environment's EC2 instances to perform operations required for your application.

IAM > Roles > aws-elasticbeanstalk-service-role

aws-elasticbeanstalk-service-role Info   Delete

Summary   Edit

Creation date
May 09, 2025, 17:47 (UTC+05:30)

Last activity
⊘ 18 minutes ago

ARN
⧉ arn:aws:iam::615299759622:role/aws-elasticbeanstalk-service-role

Maximum session duration
1 hour

Permissions | Trust relationships | Tags | Last Accessed | Revoke sessions

Permissions policies (3) Info   Simulate ⧉   Remove   Add permissions ▾
You can attach up to 10 managed policies.

Search   Filter by Type: All types ▾   < 1 >

☐	Policy name ⧉ ▲	Type ▾	Attached entities ▾
☐ ⊞	AWSElasticBeanstalkEnhancedHealth	AWS managed	4
☐ ⊞	AWSElasticBeanstalkService	AWS managed	4
☐ ⊞	tts	Customer inline	0

IAM > Roles > eb-ec2-flask-texttospeech-role

eb-ec2-flask-texttospeech-role Info   Delete
Allows your environment's EC2 instances to perform operations required for your application.

Summary   Edit

Creation date
May 09, 2025, 12:22 (UTC+05:30)

Last activity
⊘ 15 minutes ago

ARN
⧉ arn:aws:iam::615299759622:role/eb-ec2-flask-texttospeech-role

Maximum session duration
1 hour

Instance profile ARN
⧉ arn:aws:iam::615299759622:instance-profile/eb-ec2-flask-texttospeech-role

Permissions | Trust relationships | Tags | Last Accessed | Revoke sessions

Permissions policies (11) Info   Simulate ⧉   Remove   Add permissions ▾
You can attach up to 10 managed policies.

Search   Filter by Type: All types ▾   < 1 >

☐	Policy name ⧉ ▲	Type ▾	Attached entities ▾
☐ ⊞	AdministratorAccess-AWSElasticBean...	AWS managed	3
☐ ⊞	AmazonEC2FullAccess	AWS managed	2

Step 3: Set up Flask for your web application, as it is developed using Python Flask as the backend framework and Implement Polly Integration in Your Web Application
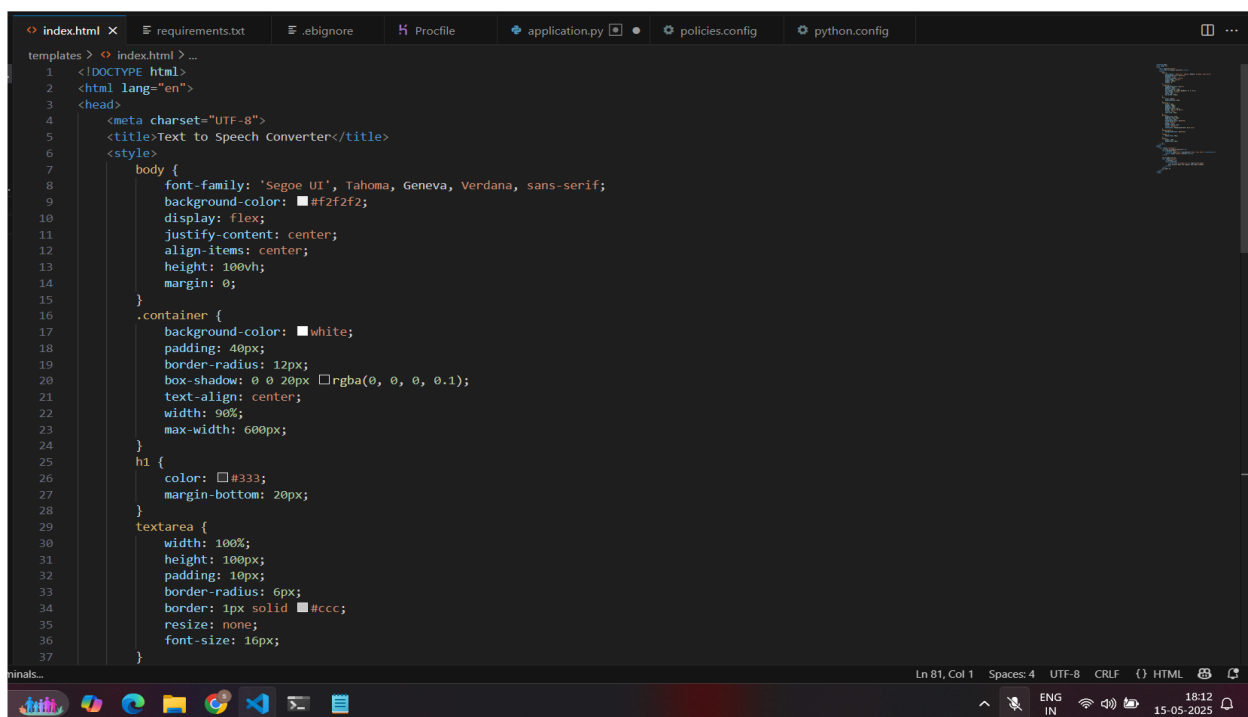


```python
    def index():
        if request.method == 'POST':
            text = request.form['text']
            if text.strip():
                try:
                    response = polly.synthesize_speech(Text=text, OutputFormat='mp3', VoiceId='Joanna')
                    os.makedirs("static", exist_ok=True)
                    audio_file = 'static/output.mp3'
                    with open(audio_file, 'wb') as f:
                        f.write(response['AudioStream'].read())
                except Exception as e:
                    return f"<p>Error: {e}</p>"
        return render_template('index.html', audio=audio_file)

if __name__ == "__main__":
    app.run(debug=True)
```

```
PS C:\Users\Hi\Downloads\AATHIRA_AWS\text-to-speech-app> python application.py
 * Serving Flask app 'application'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 138-183-058
```



```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Text to Speech Converter</title>
    <style>
        body {
            font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
            background-color: #f2f2f2;
            display: flex;
            justify-content: center;
            align-items: center;
            height: 100vh;
            margin: 0;
        }
        .container {
            background-color: white;
            padding: 40px;
            border-radius: 12px;
            box-shadow: 0 0 20px rgba(0, 0, 0, 0.1);
            text-align: center;
            width: 90%;
            max-width: 600px;
        }
        h1 {
            color: #333;
            margin-bottom: 20px;
        }
        textarea {
            width: 100%;
            height: 100px;
            padding: 10px;
            border-radius: 6px;
            border: 1px solid #ccc;
            resize: none;
            font-size: 16px;
        }
```

Step 4: Deploy Your Web Application on AWS Elastic Beanstalk and Authenticate AWS Polly with IAM role.

- Go to the **Compute** section in the AWS Management Console.
- Select **Elastic Beanstalk**.
- Click **Create Application**.
- After creating the application, click **Create Environment**.
- Configure the environment by choosing **Python as the platform**, and upload your Python Flask project as a ZIP file.
- Select IAM role for **Service Role** and **EC2 Instance Profile**.
- Finally, click on "Launch" to create and deploy your environment.
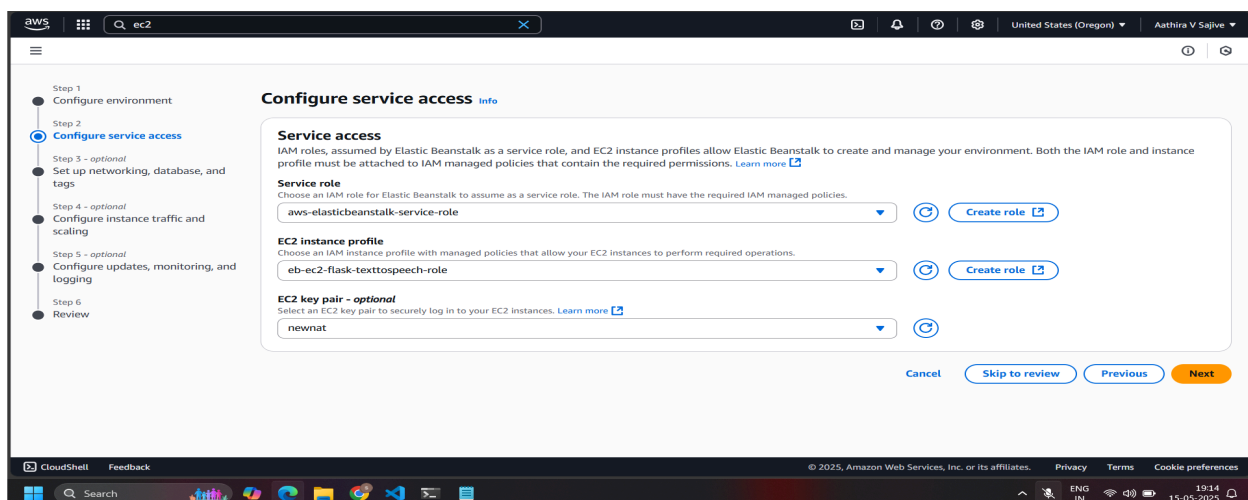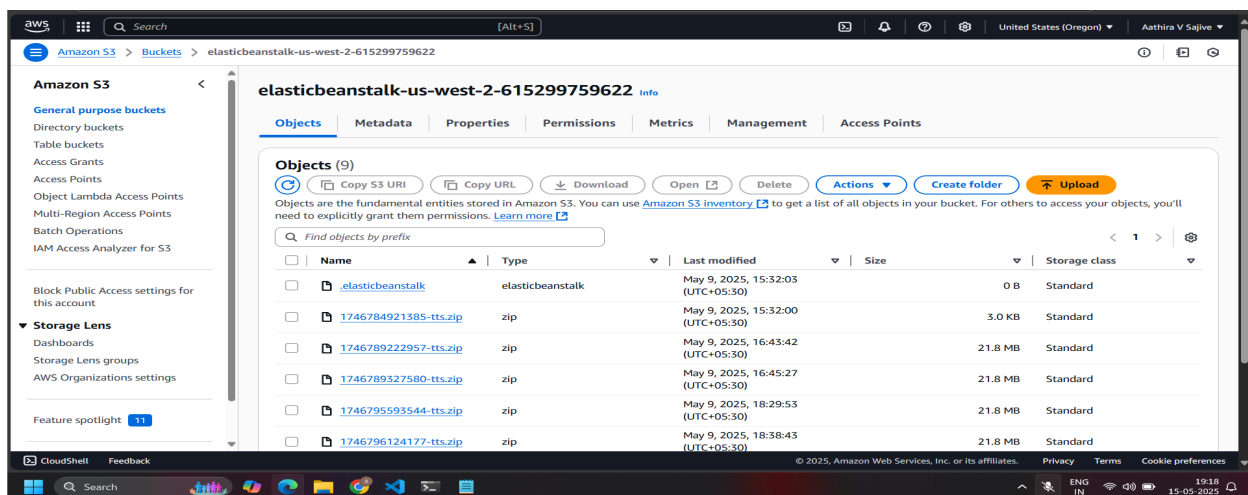
Step 5: Access the Website

- Wait for the environment health to show "OK".
- Once it's ready, click on the provided domain URL to open your deployed application.
- Open the application URL.
- Enter the desired text in the input box.
- Click on the "Convert" button.
- Use audio controls to play the generated speech.
- Click "Download" to save the audio file.

http://text-to-speech-app-env.eba-xzmducgr.us-west-2.elasticbeanstalk.com/
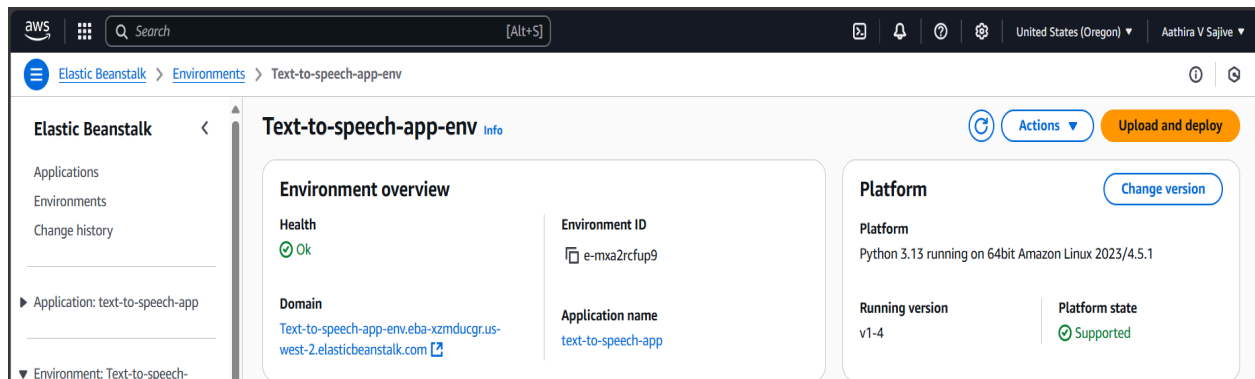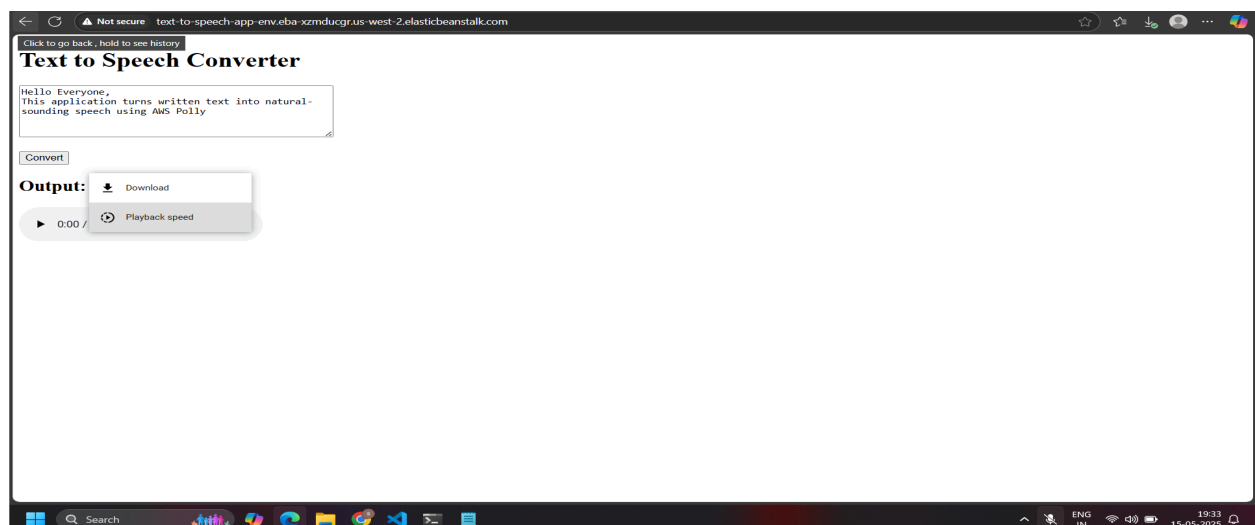
Future Enhancements

- Enable S3 storage for persistent audio storage.
- Add History Section
- Add Language & Voice Selection
- Add Text File Upload

Conclusion

This project highlights the practical use of **AWS Polly** in creating voice-enabled applications. By deploying it through **Elastic Beanstalk**, the solution ensures high availability, scalability, and manageability, making it suitable for both individual users and enterprise-grade applications.

Access the source code via the GitHub link provided.

https://github.com/AATHIRAVSAJIVE/AWS-Text-to-Speech-Converter