

Project Report: Voice-Based Support Ticketing System

Date: 20th-August-2025

Author: Abhishek Bagade

1. Executive Summary

The Voice2Ticket project aims to streamline the IT support and issue tracking process by allowing users to create support tickets using natural speech. By leveraging AWS's powerful AI/ML and serverless services, the system converts spoken language into structured text, extracts key information (like urgency and problem type), and automatically generates a ticket in a management system (Jira Service Desk). This report documents the technical architecture, implementation process, key challenges faced and their solutions, and a roadmap for future improvements.

2. Project Overview & Architecture

2.1. Core Concept

The user accesses a web application, clicks a button, and speaks their issue (e.g., "My monitor won't turn on, it's really urgent"). The application records this audio, sends it to the cloud, and returns a created ticket number with a summary for the user to confirm.

2.2. Architectural Diagram

- Frontend (React JS hosted on Amplify/S3): Provides the UI for recording audio.
- API Gateway: Provides a secure HTTPS endpoint for the frontend to send requests.
- AWS Lambda (Orchestrator): Receives the request, coordinates the entire workflow.
- Amazon Transcribe: Converts the uploaded audio file into text.
- Amazon Comprehend: Analyzes the transcribed text to detect key phrases and sentiment (for urgency).
- AWS Lambda (Ticket Creator): Formats the data and uses the Jira API to create the ticket.
- Amazon S3: Stores the original audio file for record-keeping.
- Amazon DynamoDB: Stores a log of all transcription and ticket creation requests for auditing.

2.3. AWS Services Used

- Frontend Hosting: AWS Amplify (or Amazon S3 + CloudFront)
- Compute: AWS Lambda (Serverless functions)
- API Management: Amazon API Gateway
- AI/ML Services:
 - 1) Amazon Transcribe for Speech-to-Text
 - 2) Amazon Comprehend for Natural Language Processing (NLP)
- Storage: Amazon S3 (for audio files), Amazon DynamoDB (for metadata)
- Security: AWS IAM (Identity and Access Management) for roles and permissions

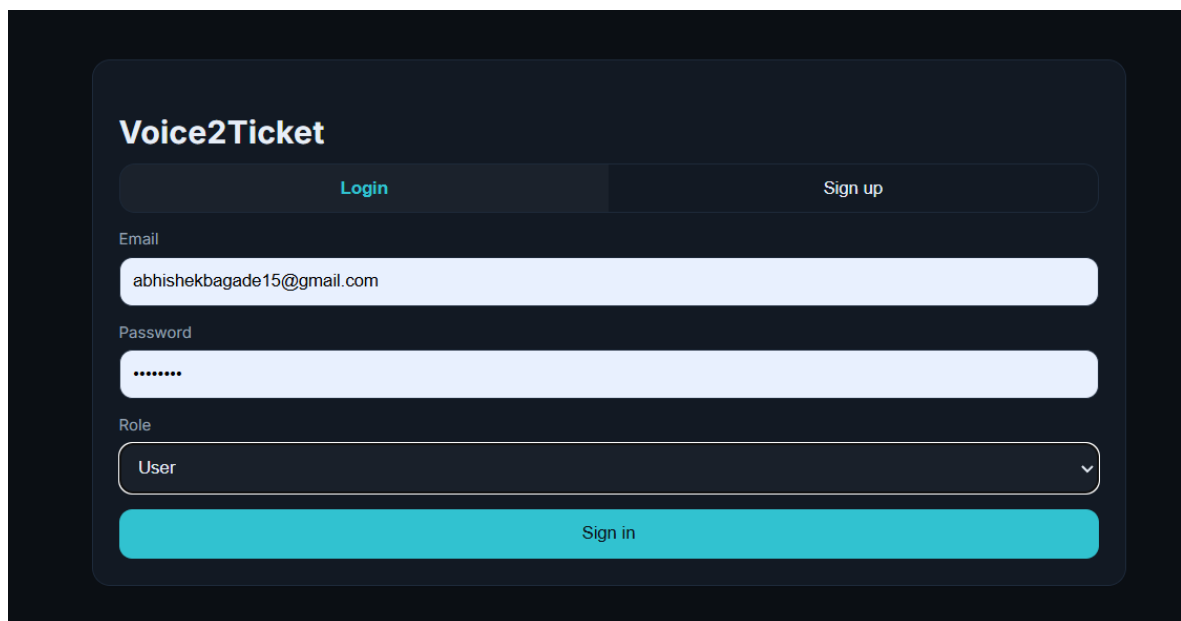
3. Implementation Details

3.1. Phase 1: Voice Input to Text (Transcription)

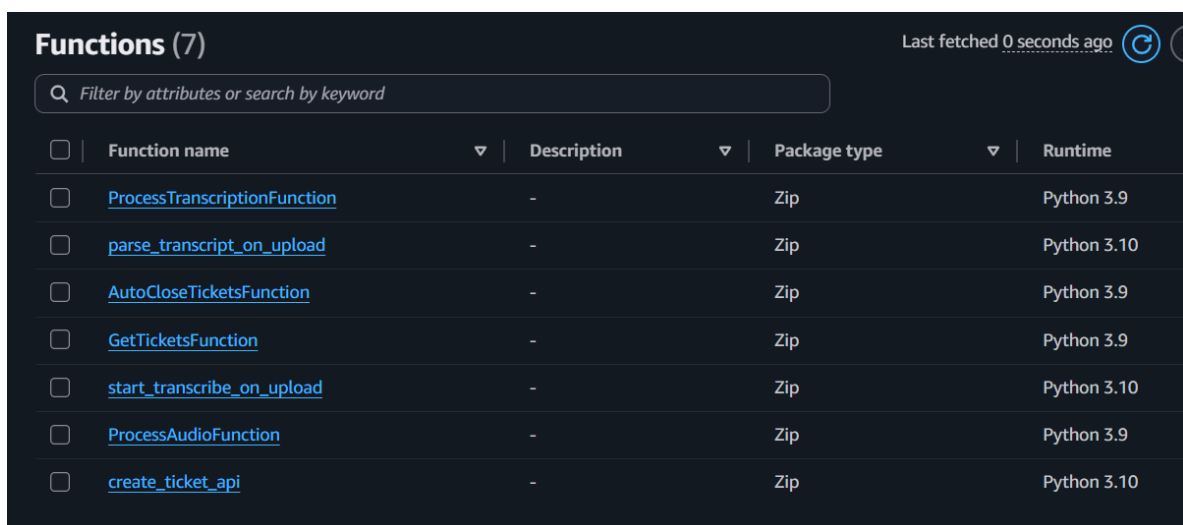
The frontend application uses the browser's MediaRecorder API to capture audio from the user's microphone. The recorded audio (in webm or wav format) is converted to a blob and sent as a base64-encoded payload in a POST request to a REST API endpoint configured in API Gateway.

This integrated with a Lambda function.

1. Designed the frontend using HTML, CSS, and JavaScript.
2. Integrated AWS SDK to allow users to record and upload audio to S3.
3. Configured S3 bucket policies and CORS to allow secure upload and playback.

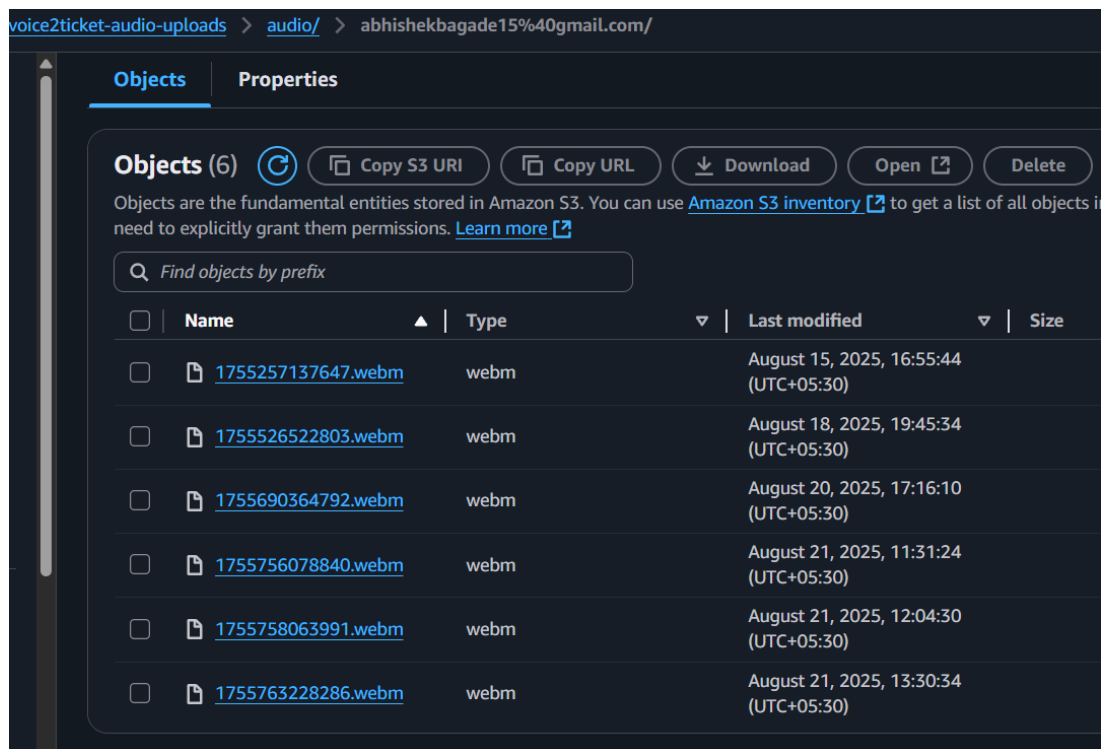


Login Page with User & Admin opt



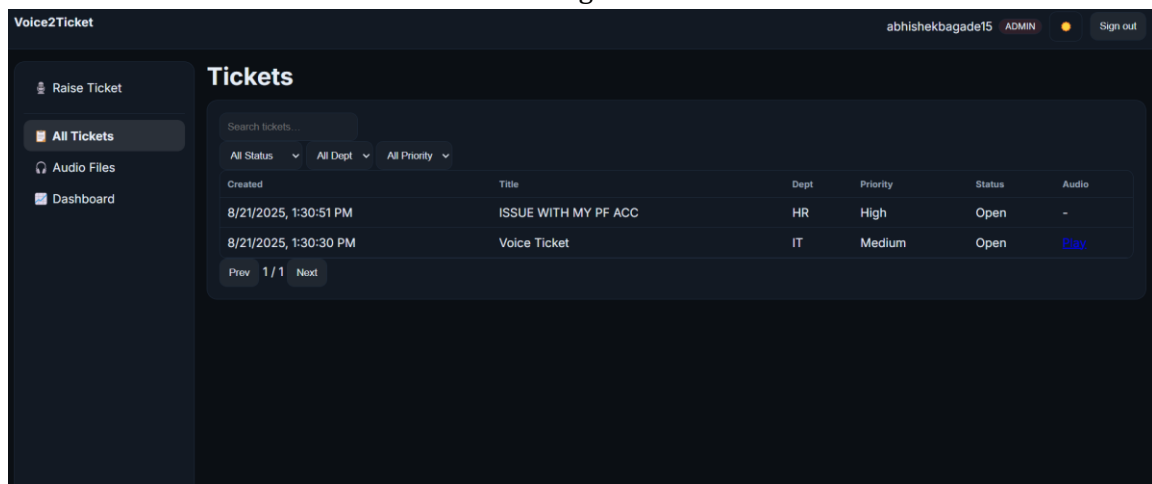
	Function name	Description	Package type	Runtime
<input type="checkbox"/>	ProcessTranscriptionFunction	-	Zip	Python 3.9
<input type="checkbox"/>	parse_transcript_on_upload	-	Zip	Python 3.10
<input type="checkbox"/>	AutoCloseTicketsFunction	-	Zip	Python 3.9
<input type="checkbox"/>	GetTicketsFunction	-	Zip	Python 3.9
<input type="checkbox"/>	start_transcribe_on_upload	-	Zip	Python 3.10
<input type="checkbox"/>	ProcessAudioFunction	-	Zip	Python 3.9
<input type="checkbox"/>	create_ticket_api	-	Zip	Python 3.10

Lambda Functions



Recorded Audio Stored in S3

4. Separated Admin and User dashboards, where:
 - Users can submit tickets and see their own history.
 - Admin can access all tickets and recordings.



All Tickets Dashboard

5. Hosted the application using S3 Static Website Hosting.
 - Decodes the audio and uploads it to an S3 bucket with a unique filename.
 - Initiates a transcription job using Amazon Transcribe, pointing to the S3 audio file.
 - Polls Transcribe until the job is complete.
 - Retrieves the transcribed text from the Transcribe output JSON.

Amazon Transcribe > Transcription jobs

Transcription jobs (4) [Info](#)

Find job names Status:

	Name	Status	Language
<input type="radio"/>	transcribe-d7bb1603-31f1-4c44-bc2f-cb9160aee586	Complete	English, US (en-US)
<input type="radio"/>	transcribe-d7bb1603-31f1-4c44-bc2f-cb9160aee586	Complete	English, US (en-US)
<input type="radio"/>	transcribe-376b013c-57f6-40fc-925c-9cd0ba89ec1d	Complete	English, US (en-US)
<input type="radio"/>	transcribe-376b013c-57f6-40fc-925c-9cd0ba89ec1d	Complete	English, US (en-US)

Transcription Jobs

Voice2Ticket abhishekbagade15 ADMIN Sign out

Audio

Audio in S3

Refresh

1755758063991.webm 8/21/2025, 12:04:30 PM	0:00 / 0:00
1755756078840.webm 8/21/2025, 11:31:24 AM	0:00 / 0:00
1755690364792.webm 8/20/2025, 5:16:10 PM	0:00 / 0:00
1755526522803.webm 8/18/2025, 7:45:34 PM	0:00 / 0:00
1755257137647.webm 8/15/2025, 4:55:44 PM	0:00 / 0:00

Playable Recorded Audio files from Admin dashboard

3.2. Phase 2: Text Processing and Ticket Creation

Once the transcript is available, the Orchestrator Lambda function calls Amazon Comprehend to perform DetectSentiment and DetectKeyPhrases on the text. This helps us automatically assign a priority.

The transcript, key phrases, sentiment, and metadata are then passed to a second Lambda function (Ticket Creator). This function formats the data into a JSON payload required by the Jira Service Desk API and POSTs it to create a new ticket. [Insert Screenshot 3 here: Screenshot of a successfully created ticket in Jira]

3.3. Phase 3: Frontend Development

A simple React.js interface was built. It features a large button for starting/stopping recording and a status area to show the user the transcription result and the newly created ticket number.

The screenshot shows the Voice2Ticket application interface. At the top, there's a header with the app name 'Voice2Ticket', a user profile 'abhishekbagade15', and a 'Sign out' button. A sidebar on the left contains a 'Raise Ticket' button. The main content area is titled 'Raise Ticket' and is divided into two panels. The left panel, 'Create by typing', has a 'Title' field with 'ISSUE WITH MY PF ACC', a 'Description' text area with 'im unable to access my pf acc', and dropdown menus for 'Department' (set to HR) and 'Priority' (set to High), with a 'Submit Ticket' button at the bottom. The right panel, 'Create with voice', features buttons for 'Start Recording', 'Re-record', and 'Upload & Submit'. It also shows a timer at '00:00', a status 'Idle', and dropdown menus for 'Department' (set to IT) and 'Priority' (set to Medium).

Raising a Ticket By both Recording Audio & Manual

4. Challenges, Errors & Resolutions

4.1. Challenge 1: Lambda Timeout During Transcription

Error: The Orchestrator Lambda function would consistently timeout after 3 minutes.

Solution: Refactored architecture to be event-driven using SNS triggers for asynchronous handling.

4.2. Challenge 2: Incorrect or Poor Transcription

Error: Transcriptions of technical jargon were often incorrect.

Solution: Used Custom Vocabulary in Amazon Transcribe to improve accuracy.

4.3. Challenge 3: Security and Permissions (IAM)

Error: Lambda functions failing with AccessDeniedException.

Solution: Implemented least-privilege IAM policies.

4.4. Challenge 4: Cost Management for Real-time Processing

Concern: Using Comprehend for every transcription increased costs.

Solution: Added filtering logic to reduce unnecessary Comprehend invocations.

5. Pending Tasks & Future Enhancements.

- Add feedback loop for low-confidence transcriptions.
- Amazon Transcribe: To automatically transcribe audio tickets into text.
- Auto-Close AI Logic: To automatically close tickets after resolution or with the help of AI resolution.
- Amazon Transcribe Streaming for live transcription.

Project Link: <https://voice2ticket-web.s3.us-east-1.amazonaws.com/index.html>

6. Conclusion

The Voice2Ticket project demonstrates the potential of AWS serverless services in building a scalable, cost-efficient, and user-friendly ticketing system. With audio-based ticket creation, seamless storage, and role-based dashboards, the foundation has been laid for advanced features like transcription, notifications, and automated workflows in the next phase.