**Project playbook**

1. **Project Onboarding:**

**1.a Overview:**

- Enable users to listen to pronunciation of employees and contractors either with standard or non-standard option.

- Ability to override standard pronunciation and provide non-standard pronunciation.

- Non-standard option if available should be the default option.

- Provide ability to lookup pronunciation for users.

- Provide the solution as an API as a service.

1.b **Team goals:** Team defined the following goals to achieve as milestones.

- Identify 3rd party API solution which provided the pronunciation service.

- Develop an API solution with public endpoint which internally calls 3rd party API service.

- Develop an UI interface which has the following features:

- Allows the user to provide first name and last name and listen to the standard pronunciation.

- Allows user to select locale for standard pronunciation.

- Allows the user to opt in for non-standard pronunciation.

- Allows the user to record, playback and upload non-standard pronunciation.

- Allows the system to playback non-standard pronunciation for an existing user who has previously saved the recording in Yugabyte database.

**1.c Dev Environment Setup:**

- Install IDE of your choice for java and angular development, JDK, Maven, SQL editor of your choice, node js, postman for API testing.

- Create account in MS azure and set up the following services:

- Speech Service: To provide the pronunciation for a given user input.

- App service: To deploy and host web application in cloud

- CI/CD: Configure GIT action to integrate git repo to azure web service for CI/CD.

- Steps to run the project end-to-end:

- Checkout code from git repo: https://github.com/wellshlsq/speech

- Import project in IDE

- Configure to run SpeechApplication.java as a Java Application.

- pom.xml is setup to build and deploy both UI and backend changes.

- launch localhost:8080 to bring up the application.

**1.d Project Building Blocks:**

- Researched different API vendors who provide pronunciation service.

- Narrowed down to Azure speech service.

- Did proof of concept to test Azure speech service.

- Built spring boot application to create an rest call to the azure service.

- Created an app service in Azure to deploy the spring boot application in azure.

- Integrated GitHub repo to the app service for CI/CD.

- Built UI layer with angular framework.

**1.e Deployment steps:**

- Commit code to feature branch.

- Create pull request in GitHub.

- Merge code to main branch.

- CI/CD will deploy the latest changes in azure cloud.

- Repositories: https://github.com/wellshlsq/speech

- Azure Services:

Speech service: https://eastus.api.cognitive.microsoft.com/sts/v1.0/issuetoken

App service: <https://hacksquadspeechappservices.azurewebsites.net>

**- Test users:**

**User 1: u3000 / password(User is existing in database with non-std pronunciation)**

**User 2: u2000 / password(User is existing without non-std pronunciation)**

**1.f Team Contacts:**

Shalaka Kulkarni

Tyler Walseth

Trish Austin

Karthikeyan Veera

Amaldass Marcellene

1. **Setup:**

**Programming language:** Java, Typescript

**Frameworks:** Angular, spring boot

**Tools:** Azure cloud service, GitHub, Maven, IntelliJ, Visual Studio.

1. **Contributing Guide:**

**Repository:** https://github.com/wellshlsq/speech

**Master branch:** main

**User journey workflow:**

Diagram

Description automatically generated

**Architecture diagram:**

**Diagram

Description automatically generated**

**Sequence diagram:**

Diagram

Description automatically generated

4. **REST APIs:**

**- Standard pronunciation:** https://hacksquadspeechappservices.azurewebsites.net/message/{firstName}/{lastName}/{neural region}

**- Non-standard pronunciation:** https://hacksquadspeechappservices.azurewebsites.net/getCustomRecording/{firstName}/{lastName}

**- Upload non-standard pronunciation:** https://hacksquadspeechappservices.azurewebsites.net//uploadRecording/{fName}/{lName}