Coms 6998-009 Final Project Documentation

Name: Chelsea Cui (ac4788)

Project: Practiced

Problem Statement

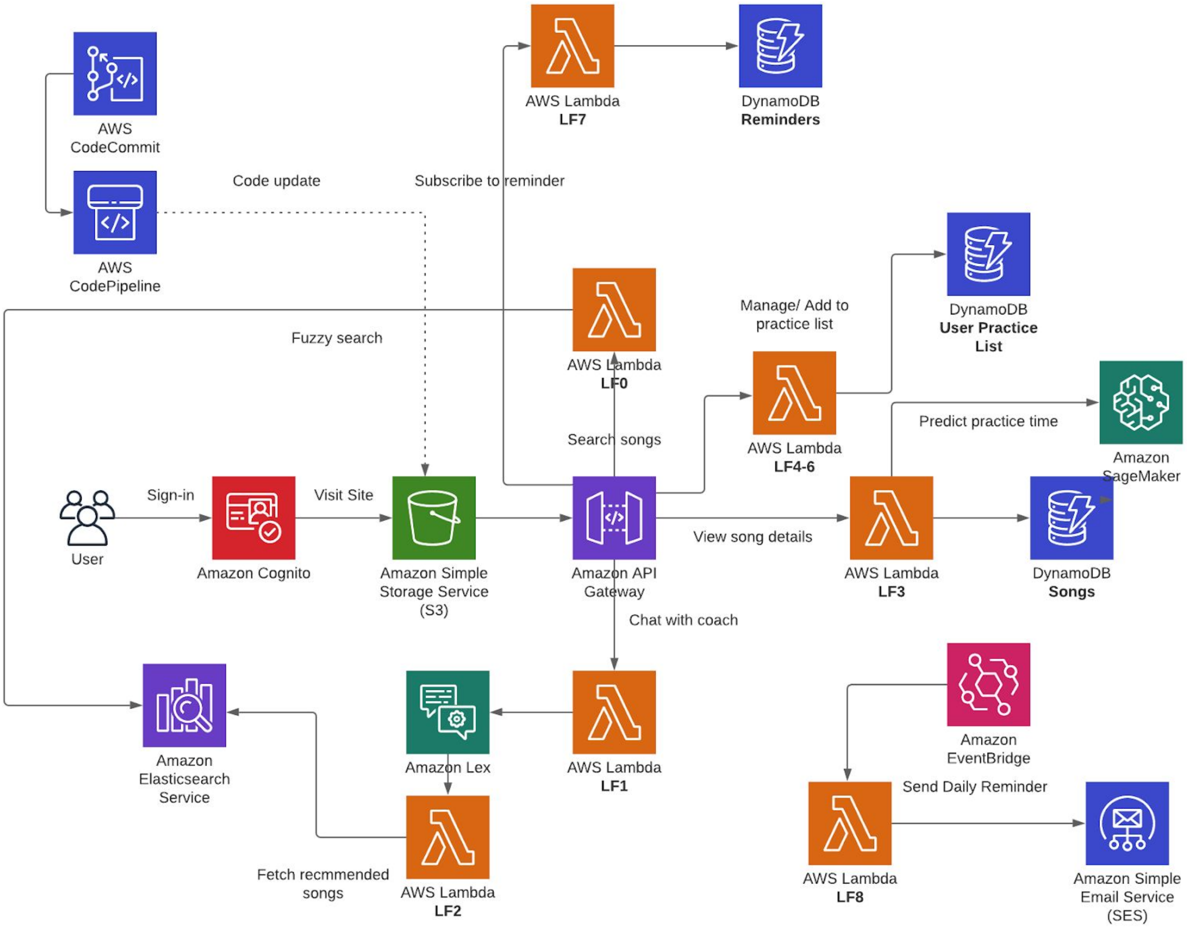
- **Target User**: Self-starter musicians
- **Problem**:
  - With limited professional guides, self-start musicians may be confused about what to do to get started.
  - There are too many articles and videos online, but there's no clear pathway on what songs people should choose.
  - Without any regular reminder by a teacher or friends, people may forget to practice and then give up silently.
- **Solution**:
  - A web application which could
    - Help people find songs they are interested and suitable for their level
    - Provide personal and conversational guidance
    - Remind people to practice regularly and build good habits

Use Cases

Case	**Description**
Search songs	• Search by artist, genre, song name • See song information, difficulty and expected practice time • Fuzzy search by a rough idea • Search by voice
Personal account	• Register for a new account • Log in/ log out to keep the experience personal

Practice List	● Add a song to practice list ● Record daily practice ● Finish practicing a song
Personal Coach	● Conversational suggestion
Reminder	● Subscribe to email daily reminder ● Receive daily practice reminder to build a habit

Solution Architecture



Key Design Decisions

- **Efficient Search:** Maintain song keywords (name, artist, tags) in ES for fuzzy search and recommendation.
- **Cost Effective:** Maintain detailed song information in DynamoDB.
- **Personalized Experience and Account Safety:** User management system with Cognito.
- **Flexible Record:** After a song is finished, the record will stay at the bottom. In this way, users can build a sense of accomplishment along the way.
- **Easy Update:** Code versioning with CodePipeline and CodeCommit.
- **Smart Prediction**: Using Sagemaker to predict practice time

Database Design

Elasticsearch

Index: Songs

```
{"artist": "N. Paganini",

"name": "Paganini Caprice No.5",

"difficulty": 5,

"youtube": "0jXXWBt5URw",

"resource-link":
"https://violinsheetmusic.org/files/download/classical/paganini-caprice-5.pdf",

"resource-name": "Music Sheet for Violin and Piano",

"tags": ["Classical Music", "Ling Ling's favorite"],
}
```

DynamoDB

Table: Songs

```
{
  "artist": "N. Paganini",
  "difficulty": 5,
  "Id": "1",
  "name": "Paganini Caprice No.5",
  "resource-link":
"https://musescore.com/musemeister/paganini-cap5",
  "resource-name": "Music Sheet for Violin and Piano",
  "tags": [
    "Classical Music",
    "Ling Ling's favorite"
  ],
  "youtube": "0jXXWBt5URw"
}
```

Table: Practices

```
{
  "endDate": "2020-12-27",
  "finished": 1,
  "practiceTime": 95,
  "songId": "1",
  "songName": "Paganini Caprice No.5",
  "startDate": "2020-12-27",
  "username": "chelseatest"
}
```

Table: Reminders

```
{
  "email": "aiqichel1652@gmail.com",
  "username": "chelseatest"
}
```

API Design

/coachbot

- POST
    - Post message to Lex bot from frontend interface

/practice-list

- GET
    - Get all songs in user’s practice list
- PATCH
    - Add new song to user’s practice list
- PUT
    - Finish practice a song or add practice time

/song-info

- GET
    - Get song information from DynamoDB

/songs

- GET
    - Fuzzy search songs on ES

/reminders

- POST
    - Add subscriber email to DynameDB

Lambda Documentation

This project used 9 Lambda Functions in total. The functionalities are described below:

Name	Description

LF0-search	Connect frontend search bar with ES to conduct fuzzy search
LF1-SendChat	Send chat from frontend bot to AWS Lex bot
LF2-BotReply	Generate song recommendation based on conversation
LF3-SongInfoGet	Get song detailed information from DynamoDB and predict practice time with Sagemaker model
LF4-GetPracticeList	Get practice list for a user
LF5-AddToList	Add a new song to user's practice list
LF6-UpdatePractice	Update user's practice record (add time/finish practice)
LF7-AdjustReminder	Add subscriber's email to DynamoDB
LF8-DailyRemind	Listen to EventBridge and send daily email via SES

Environment and Requirements

- Lambda functions are written in Python 3.8
- For LF0 and LF2, install elasticsearch module in lambda is required

Selected Test Cases

I did many tests along the way, here are some selected ones to keep in mind.

Cognito

Case	**Status**
User should be able to create an account	Passed
User should be able to log in to an account	Passed
User should be able to log out from an account	Passed
Before login, user cannot add songs to list	Passed

Before login, user cannot view practice list	Passed
Before login, user cannot view profile page	Passed

Practice List

Case	**Status**
The practice list should be personal to the logged in user	Passed
User should be able to add practice time	Passed
User should be able to finish practicing a song	Passed
User should be able to add a song to practice list	Passed
FInished songs should be at the bottom of the list	Passed
Finished songs should be shown as “completed”	Passed

Coach Bot

Case	**Status**
Bot should be connected when the site is visited	Passed
Bot should be able to greet user	Passed
Bot should be able to provide song suggestion if related record exists in ES	Passed
Bot should be able to indicate the result if no related record found for song recommendation	Passed

Search

Case	**Status**

User should be able to search songs based on artist name	Passed
User should be able to search songs based on song name	Passed
User should be able to search songs based on fuzzy genre or style keywords (if tag exists in DB), e.g. Pop, Classical. Love	Passed

Song Info

Case	**Status**
Each song should have a predicted practice time based on difficulty. Result should be generated from sagemaker	Passed

Codepipeline

Case	**Status**
Code should be updated when new commit is pushed to remote repository	Passed

Daily Reminder

Case	**Status**
User who subscribed to daily reminder should receive a reminder every day	Passed
User can only subscribe to reminder once. Only one email for each user.	Passed
User must enter a valid email to subscribe	Passed

Further Steps

I think this project has a potential and recognizable market. The conversational coach could be a very helpful tool for self-starter musicians if the backend could be combined with some recommendation model that can learn user interests or assess user groups with similar expertise levels.