

# Spotify Music Recommendation Chat-bot



## Intelligent Reasoning Systems Practice Module

### **Group members:**

Shi Qirui	A0261896M
Chen Huizhou	A0261843A
Tian Hengyi	A0261841H
Tang Junbo	A0226717B

1. Executive Summary	3
1. Business justification	5
2. Knowledge modeling	7
3.1 Data	7
3.2 Recommendation Algorithm	8
3.2.1 Content-Based Recommendation	8
3.2.2 Similarity Measure	9
3.3 To Perform Natural Language Processing	9
3.3.1 Agent	10
3.3.2 Intent	10
3.3.3 Entity	13
3. System design	14
4.1 Workflow	14
4.2 Tools and Techniques	15
4.3 backend logic	16
4. Conclusion	18
5.1 Team members	18
5.2 Challenges we met	19
5.3 Further improvements	19
Appendix 1: Project Proposal	22
Appendix 2 Mapping Functionalities	27
Appendix 3: Reference	28
Appendix 4: User Guide	29
1. Install Python and Packages	29
2. Download datasets	30
3. Import the agent from Dialogflow	30
4. Get a GOOGLE_APPLICATION_CREDENTIALS private key	31
5. Database setting flow:	36
6. Spotify API	38
7. Final run	40
Appendix 5: Individual Report–Shi Qirui	41
Appendix 6: Individual Report–Tang Junbo	43

Appendix 7: Individual Report–Tian Hengyi	45
Appendix 8 Individual Report – Chen Huizhou	47

# 1.Executive Summary

Spotify, as one of the most well-known digital music services, is widely used by music lovers worldwide. Spotify is a powerful platform as people can find songs they want in its music library. There is no doubt that Spotify is one of the most popular music platforms all around the world.

In recent years, chat bots are becoming a popular human-computer interaction application by which people can customize their needs in an intuitive way. An increasing number of services are adding chatbot modules to improve user experience and the development of chat bot is a trend of the era.

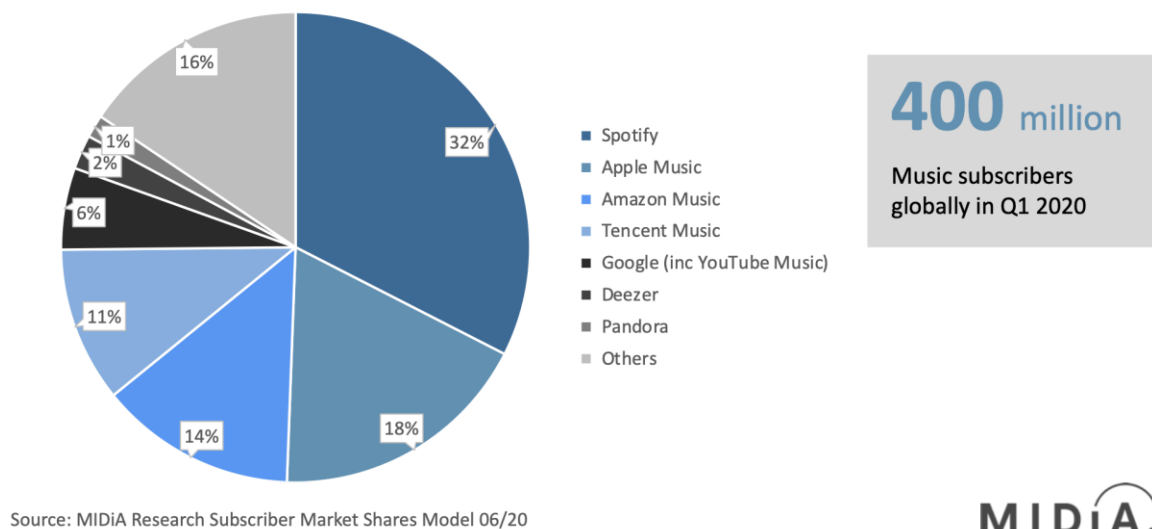
In this project, our team managed to create a chat bot which can respond to the user's input and perform interaction with the user's Spotify account. It can help them find songs they want, play a demo, and make recommendations. Users can talk with the chatbot, ask it some questions and ask for a recommendation for music in the Spotify library based on the artist's name, music genre or artist genre. The chatbot is also capable of obtaining a user's playlist on Spotify and making recommendations based on it. A HTML frontend is created for the chatbot and backend is built based on the Flask framework. This chatbot only considers music from 2000 to 2020 included in the Spotify music library so it is still limited to some extent. Nevertheless, it still has potential to be further developed in functions and databases.

Keywords: music recommendation, Flask, HTML, Dialogflow, Chatbot, Spotify

## 2.Business justification

As one of the most influential media services providers, Spotify offers digital copyright restricted recorded music and podcasts, including more than 82 million songs. With more than 433 million active users monthly, Spotify aims to make the largest music library for music lovers and provides music recommendation service to users. The research on the music service market shows that Spotify tops other music services in recent years.

Despite strong growth from Amazon and Apple, Spotify retains the same global music subscriber market share in Q1 2020 as Q1 2019

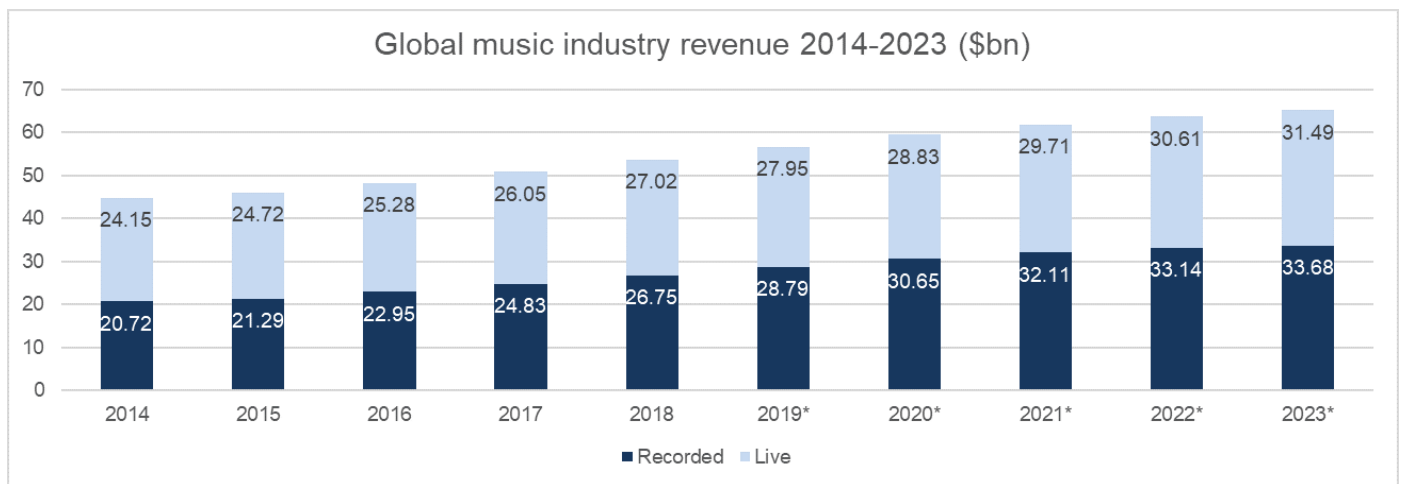


**Fig 2.1** A pie chart indicating the Music Subscriber Market Shares Model in 2020

With chatbot technology developing, a great number of users of media services are becoming willing to express their requirements in an intuitive and clear way like ‘chatting’ with a server. Such a trend leads to the systematic perfection of chatbots and some powerful chatbots are bringing convenience to users for certain.

The spread of COVID-19 influences people’s social life a lot as offline activities are restricted. Fewer commuting journeys and the gym closures have shifted listening to different parts of the day. An increasing number of people prefer using chatbot to get service as a simulation of social life, which stimulates the development of media chatbot technology.

Moreover, recorded music is now becoming much more popular than live music in recent years because of COVID-19. Live music and recorded music are two major income streams for the music industry while the concerts and live performances are restricted in many countries. A chatbot which can recommend music for users is definitely attractive in that case. Besides, traditional recommendation systems for music mostly focus on the features of a certain song. The idea of making use of the whole playlist is inspiring and will be a nice choice for better performance of chatbot service.

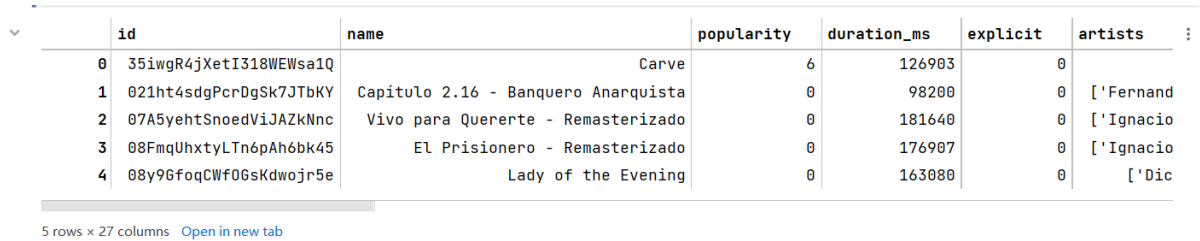


**Fig 2.2** A bar chart illustrating the changes in global music industry revenue from 2014 to 2023

## 3. Knowledge modeling

### 3.1 Data

The source of our data is the public spotify dataset published on Kaggle, which includes audio features of 600k+ songs and popularity metrics of 1M+ artists, ranging from 1921 to 2020. There are two csv files in the dataset, artists.csv and tracks.csv. The first one contains the basic information of artists like IDs, names and genres, while the audio features like attributes, popularity and released year are in the tracks.csv file. Some data preprocessing methods were implemented to extract the genres of artists and corresponding artists' IDs, which were then merged with track information. Finally, a processed track dataset was obtained, from which we recommended songs for users.



	id	name	popularity	duration_ms	explicit	artists	:
0	35iwgR4jXetI318WEWsa1Q	Carve	6	126903	0		
1	021ht4sdgPcrDgSk7JTbKY	Capitulo 2.16 - Banquero Anarquista	0	98200	0	['Fernand	
2	07A5yehtSnoedViJAZkNnc	Vivo para Quererte - Remasterizado	0	181640	0	['Ignacio	
3	08FmqUhxtyLTn6pAh6bk45	El Prisionero - Remasterizado	0	176907	0	['Ignacio	
4	08y9GfoqCWf0GskDwojr5e	Lady of the Evening	0	163080	0	['Dic	

5 rows x 27 columns [Open in new tab](#)

**Fig 3.1.1** An example of track dataset

Based on the processed track dataset, we generated another dataset including song features to calculate similarities. Beside attributes of songs, genres are expected to help the recommendation. Therefore, a Document-term matrix (DTM) of genres is produced and the genre term counts are weighted using the TF-IDF method. TF-IDF stands for term frequency-inverse document frequency, which can quantify the importance or relevance of string representations (words, phrases, lemmas, etc) in a document amongst a collection of documents. The TF-IDF scores for the word  $t$  in the document  $d$  from the document set  $D$  is calculated as follows:

$$tfidf(t, d, D) = tf(t, d) \times idf(t, D)$$



where,

$$tf(t, d) = \log(1 + freq(t, d))$$

$$idf(t, D) = \log\left(\frac{N}{count(d \in D: t \in d)}\right)$$

Not only the text based feature and track attributes, track's released year and its popularity are considered as well. The detailed data preprocessing could be seen through `data_preprocess.ipynb` file. As a result, a track dataset and its corresponding complete feature dataset are the foundation of our recommendation algorithm.

	genre adult_standards	genre album_rock	genre alternative_metal	genre alternative_rock	genre anime	:
0	0.619261	0.0	0.0	0.0	0.0	
1	0.000000	0.0	0.0	0.0	0.0	
2	0.619261	0.0	0.0	0.0	0.0	
3	0.619261	0.0	0.0	0.0	0.0	
4	1.000000	0.0	0.0	0.0	0.0	

5 rows × 180 columns [Open in new tab](#)

**Fig 3.1.2** An example of track dataset

To be mentioned, the produced feature dataset may be too large for storage and would significantly influence the running speed due to the 4000+ recognized genre kinds and 600k+ tracks. Therefore, a limitation on the released year was set. Only tracks released during 2000 and 2020 are considered, which still reaches the number of 180k+. What else, only genres which take up to 0.001% total frequency are chosen as the text based features. As a result, there are 165 most popular genres mentioned in our complete feature dataset.

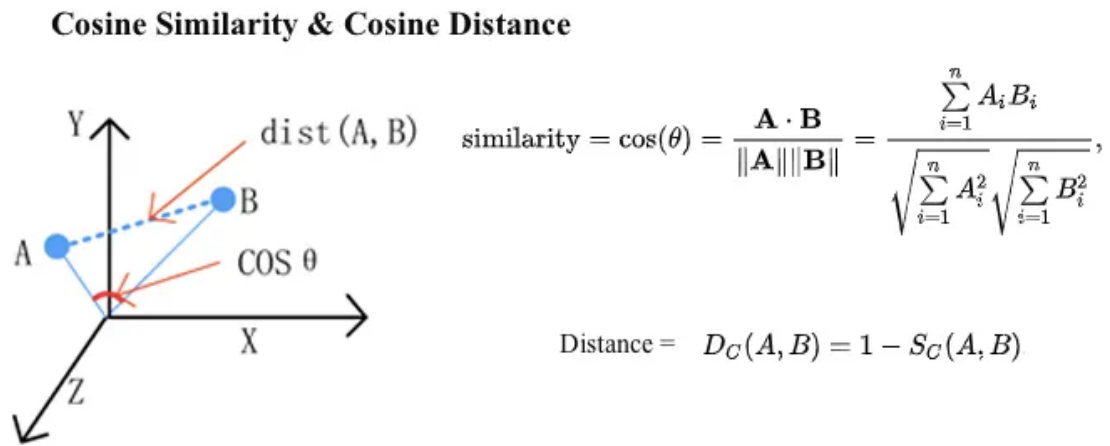
## 3.2 Recommendation Algorithm

### 3.2.1 Content-Based Recommendation

The recommendation algorithm used in this project is content-based recommendation. The system recommends similar tracks based on tracks in the current user's Spotify playlist, which can be obtained through Spotify API and the App created on the website of Spotify for Developers. After that, the database is looked up to filter out tracks in the playlist that do not

exist in the database. Also, there is a weight related to the period since the track was added to the playlist. The weights of recently added tracks is higher than weights of tracks that are in the playlist for a long time. After multiplying the weight, the track feature vectors take summation to form the playlist feature vector.

### 3.2.2 Similarity Measure



**Fig 3.2** An image demonstrating the definition of cosine similarity

For the content-based recommender system, cosine distance is typically used to measure the similarity. Therefore, the similarities between track vectors in the feature dataset and the produced playlist vector are calculated using cosine similarity. The top 10 tracks which have the high similarities are going to be recommended to the user.

## 3.3 To Perform Natural Language Processing

In our project, we used Dialogflow to help us perform natural language processing. DialogFlow is a natural language processing platform provided by Google, which allows us to easily design conversational interfaces and integrate them into our chat-bot. It plays an essential role in our project, as we use it to identify the intent of a user input and the content in the user's response text, which could help us to understand the desire of the user and handle the information from the user exactly. After obtaining intent and

parameters, our backend system would perform rule-based reasoning, it will make different responses and operations according to current intent.

### 3.3.1 Agent

An agent is a virtual agent in Dialogflow that handles concurrent conversations with the end-users, which are the users of our chat-bot. A Dialogflow agent is similar to a human call center agent. As we train them both to handle expected conversation scenarios. An agent can be considered as a natural language understanding module that understands the nuances of human language, in this case is the user input of our chat-bot. An agent can return structured data that Dialogflow provided based on the end-user text. Therefore, the chat-bot can use these structured data to make decisions and give responses.

To build a Dialogflow agent, we need to specify the language. In this project, we set it to “English — en”. And also, we can adjust its “ML CLASSIFICATION THRESHOLD” if we want more accurate classification on the intents. In our case, during testing, we found Dialogflow works fine with classifying the intents we set, so we still use its default value, which is 0.3.

### 3.3.2 Intent

An intent categorizes an end-user's intention for one conversation turn. Dialogflow allows us to define many intents in an agent, to combine these intents and handle a complete conversation. When an end-user writes or says something, referred to as an end-user expression, Dialogflow matches the end-user expression to the best intent in the agent. Matching an intent is also known as intent classification.

To create an intent, we must also provide training phrases for it. Training phrases means the text that users might type in for each specific intent. When a user expression resembles one of these phrases, Dialogflow would match the intent and do following operation like entities identification. One thing needs to be noticed is that we don't have to define every possible

example, because Dialogflow's built-in machine learning expands on the phrases list with other similar phrases.

For example, we create an intent named AddToPlaylist, which means the user wants to add a song to his current playlist. If the user types “I want to see Rap God in my playlist!” Dialogflow would classify this user text to AddToPlaylist intent and extract parameters like:

music-name              Rap God

playlist                  playlist

Then, our backend system could use these structured data to give relevant responses.

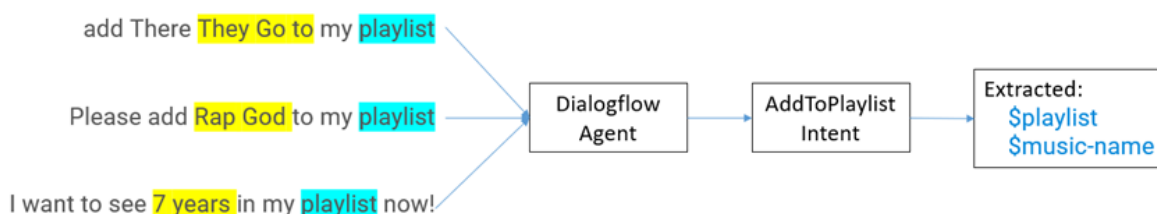


fig3.3 AddToPlaylist Intent

These are the intents we set for the chatbot:

1. Recommendation: This intent means the user wants our chat-bot to recommend some songs, based on the user's Spotify playlist or typed text. And it contains following parameters:

Parameter Name	Entity	Value
music-artist	@sys.music-artist	\$music-artist
Playlist	@Playlist	\$Playlist

music-name	@music_name	\$music-name
music-genre	@music-genre	\$music-genre

Playlist: if a user mentioned a playlist in the input text, this parameter will be given a value, hence our backend system could identify if the users want recommendation from their Spotify playlist or not.

music-artist, music-name, music-genre: seed artist, music, genre that users want recommendation from. Even if the users' input is implicit, we can also use the Spotify API to make recommendations.

2. AddToPlaylist: This intent means users want to add songs to their playlist. This intent contains the following parameters.

Parameter Name	Entity	Value
Playlist	@Playlist	\$Playlist
music-name	@music_name	\$music-name

Playlist: we set this parameter to “required”, which means unless users mentioned playlist, this intent won't be triggered, it can help Dialogflow classify intents more accurately.

music-name: the name of the song that users want to add to their playlist.

3. SetPlaylist : this intent means users want to set the current playlist, for recommendation or AddToPlaylist.

Parameter Name	Entity	Value
----------------	--------	-------

Playlist	@Playlist	\$Playlist
----------	-----------	------------

Playlist: similar to playlist in intent “AddToPlaylist”

4. Default Welcome Intent and Default Fallback Intent: the default intents provided by Dialogflow. Default Welcome Intent could make responses to users’ greetings, and Default Fallback Intent could handle the situation when users’ input text cannot be well classified (confidence score less than ML CLASSIFICATION THRESHOLD).

### 3.3.3 Entity

Each intent parameter has a type, called the entity type, which dictates exactly how data from an end-user expression is extracted.

Dialogflow provides predefined system entities that can match many common types of data. There are system entities for matching dates, times, colors, email addresses, and so on. And for the entities Dialogflow doesn’t provide, we can also create our own custom entities for matching custom data. And here are the entities that we created for the chat-bot:

1. music-genre: the genre of the music. Dialogflow provide a system entity for music genre, which is @sys.music-genre. Since the reference values of it don’t match the required genre of Spotify API, we used a function in Spotify, “recommendation\_genre\_seeds” to grab the available genre of Spotify API, and also tried to set synonyms of each genre.

2. music-name: the name of the music.

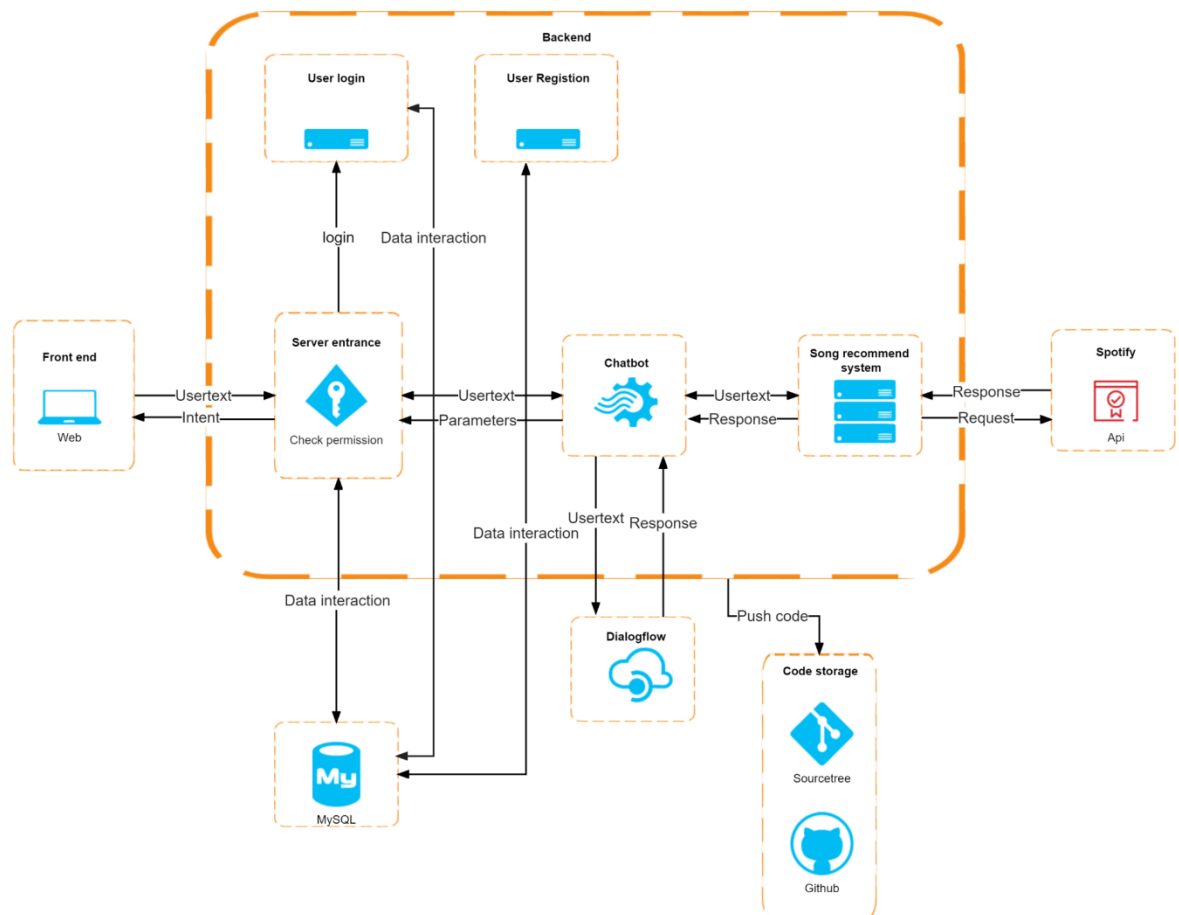
3. playlist: to see if the user mentioned playlist.

And we also used entity that provided by Dialogflow:

@sys.music-artist: the name of a music artist.

## 4. System design

### 4.1 Workflow



When a user logs in to the website and jumps to the chatbot page, the system first checks whether the user has login permission. If the user does not have the login permission, it will be forced to switch to the login page. In this case, the user can choose to log in or register. Secondly, the dialogue content will be sent to the backend for processing. After that, it will be sent to Dialogflow, which is deployed on google cloud platform and get intent and parameters. Then it will use the song-recommend-system to recommend and call spotifyApi to get the relevant music information, for example, singer name; Release time; The preview and so on. Finally, package the important data and send it to the front end for display and further interaction.

## 4.2 Tools and Techniques

**Flask** : Flask is a micro web framework written in Python. It is classified as a microframework because it does not require particular tools or libraries. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions. However, Flask supports extensions that can add application features as if they were implemented in Flask itself. Extensions exist for object-relational mappers, form validation, upload handling, various open authentication technologies and several common framework related tools.

**PermissionCheck** : In this part, we adopt the extended application features of flask, such as login\_user, generate\_password\_hash and other functions. And realized the password of the user pbkdf2:sha256 encryption, the realization of using remember\_token to save the login status of the users, and monitor and verify the login status of the users, so as to ensure the security of the system

**Mysql**: At the beginning of the design we had two options: SQLAlchemy, a database object Relational mapping (ORM) tool, and pymysql to inject sql statements. Due to our demand for databases, pressure is not very high, and there is no demand to write complex Sql statements. We want to use SQLAlchemy to operate the database, which not only helps to quickly solve the early stage of development when the database is still not fully designed, but also can quickly modify the content, and reduces the bug.

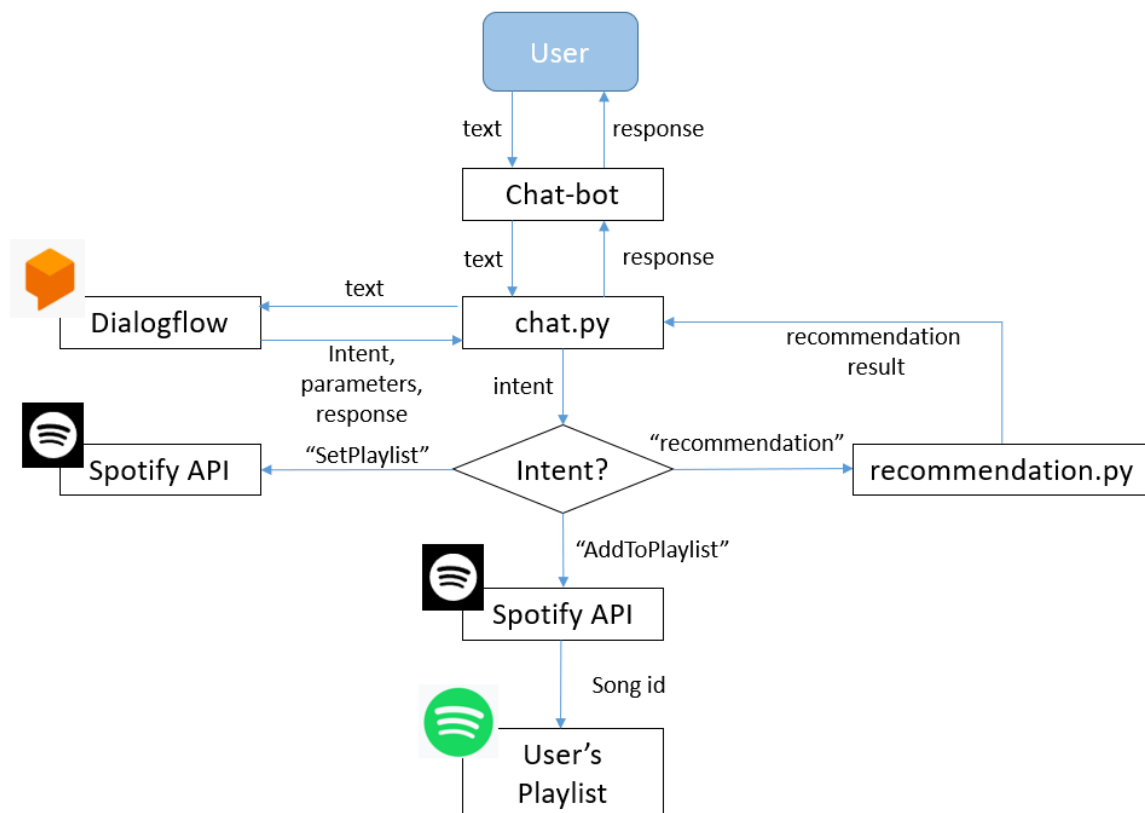
**SpotifyApi**: Based on simple REST principles, the Spotify Web API endpoints return JSON metadata about music artists, albums, and tracks, directly from the Spotify Data Catalog. Web API also provides access to user related data, like playlists and music that the user saves in the Your Music library. Such access is enabled through selective authorization, by the user. The base address of Web API is <https://api.spotify.com>. The API provides a set of endpoints, each with its own unique path. To access private data through the Web API, such as user profiles and playlists, an application must get the user's permission to access the data. Authorization is via the Spotify Accounts service.



**Dialogflow :** By creating a service account in Google Cloud and obtaining the private key, we connect the back-end of our project to Dialogflow. After discussion, we think that fulfillment is not conducive to our back-end logic, because we may first do some operations on the content delivered from the front end to the back end. Then call Dialogflow API, so we are more inclined to use the service account of Google Cloud for interaction.

### 4.3 Backend logic

For the backend system, as mentioned earlier, we used the Flask framework with python to build it.



After a user talks to our chat-bot, our backend system will capture the text that users sent. Then, the backend will call Dialogflow and pass the user text to it, then obtain the intent, parameters and response from it. Then perform different operations according to the intent.

If the intent is “Recommendation”, our system will firstly check if the user already chose a playlist as the base of recommendation. If not, the system will grab the user’s playlist and let the user choose one. Then the system will call the recommendation algorithm to get the result. After providing the recommendation result to the user (each song in a button), the user could click on each song to get a 30s preview of the song, provided by Spotify. If Spotify doesn’t provide a preview of that song, we will provide the URL of that song in Spotify. Users can check the song on Spotify directly through the URL.

If the intent is “AddToPlaylist”, our system will first check if the user has already chosen a playlist. If not, the system will grab the user’s playlist and let the user choose one. Then check if the song is in the recommendation result (in which case it would be easy to get the id of the song), the system would directly add the song into the chosen playlist. If the song is not in the result, the system would call a Spotify API to search the song based on the user text, and grab the id from the response, then add the song into the playlist.

If the intent is “SetPlaylist”, the system would get the user’s playlists through Spotify API, and let users choose one of them.

If the intent is “Default Welcome Intent”, the system will directly return the response from Dialogflow to the user.

## 5.Conclusion

### 5.1 Team members

Name of group member	Work items
Chen Huizhou	<ul style="list-style-type: none"><li>● Project ideation</li><li>● Designing Dialogflow agent</li><li>● Implementing Backend logic</li><li>● Manipulating Spotify API</li><li>● System debugging</li></ul>
Shi Qirui	<ul style="list-style-type: none"><li>● Engaged data preprocessing</li><li>● Designed recommendation model</li><li>● Added authentication (sign up and login) using Flask</li><li>● Integrated the whole code and debugging the system</li></ul>
Tang Junbo	<ul style="list-style-type: none"><li>● Build the project framework</li><li>● Implement frontend to backend data interaction, such as audio playing.</li><li>● Test connects to dialogflow and backend through gcp</li><li>● Test spotify certification.</li></ul>
Tian Hengyi	<ul style="list-style-type: none"><li>● Generated the project idea</li><li>● Designed the HTML frontend for chatbot using CSS</li><li>● Engaged in data preprocessing</li><li>● Generated the intents to be recognized by chatbot using Dialogflow</li><li>● Debugged the chatbot</li></ul>

## 5.2 Challenges we met

Overall, it was a wonderful time working on this project. Everyone in the team has tried our best to make contribution and we performed like a good team with individual tasks and in-time communication with team members. To decide the topic of a project is the initial problem we met. We should choose a topic based on knowledge and techniques we acquired which meets the requirements given. We also learnt how to make great teamwork by task assigning and cooperation: interpreting the works we have done to peers in case of redundant works and discussing about functions we want to add with each other to see whether it is feasible from other teammates' perspective. Besides, as the database we chose at first contains more than 600,000 samples of songs, we found the preprocessed dataset containing too many features, leading to higher memory footprint and lower and lower computational speed for cosine similarity algorithm. To address the problem we decreased the number of songs to be considered and the final chatbot is capable of recommending music from 2000 to 2020. Moreover, a filter to select noteworthy features is added to improve performance of the recommendation system. We also tried to use SQLAlchemy instead of pymysql as the toolkit for database operation for a more proficient and convenient way for database access.

In addition, how to connect the backend we constructed with Dialogflow troubled us when trying to get intents we defined before using chatbot. We searched for the solution and finally decided to use a private key to access Dialogflow agent, which is introduced in the 'tools and techniques' before.

## 5.3 Further improvements

With regard to the further improvement for this project, the first problem to be solved is the limitation of the database. As there are more than 600,000 songs in the original database we chose while only 180,000 of them are

taken into consideration because the features extracted would be too sparse when too many samples are included, which gives rise to a great load.

Besides, too many sample data would also lead to an untimely commendation because of excessive calculation for cosine similarity algorithm. To remove the redundant and fuzzy features in the database, only the features (e.g., genre) existing in more than 0.01% of songs in the database are adopted. However, to make a more precise recommendation, it is helpful to include more songs and features.

There are two possible solutions for problem mentioned above:

1. Improve the method for feature extraction

While only features existing in 0.01% of songs are selected in this project, the proportion for feature selection can definitely be adjusted to improve the precision of recommendation. In addition, some features that rarely occur can be merged with frequent features, which is another helpful way to improve the performance of the recommendation algorithm.

2. Improve the algorithm

The cosine similarity algorithm we use for recommendation can be improved. A improvement is to use unsupervised algorithms (e.g., K-means clustering) for recommendation and compare the similarity of each song to other samples in the database. This is a preferable way to improve the speed for recommendation because of the decrease in number of features to be considered. Nevertheless, to use a supervised algorithm helps in implementing other functions as the precise attributes of recommended songs can be returned. The developer should take the demerits of both methods into consideration and try achieving a balance.

Another way to improve this project is to find a better way to give each song of the playlist a proper weight. For now, the solution provided for adding weight to songs in the playlist is to check the time duration since a certain song is added into the playlist. However, this is not a quite precise way to

give weights to each song as some songs in the playlist can be the favorite ones for a user even if it was added in the playlist for a long time.

One possible solution is to obtain how often each song in the playlist is played recently and add weight to them according to frequency. Besides, a feedback function can be developed in this chatbot to check whether the users are satisfied with the recommended songs returned and ask them to add songs they prefer from the recommended song list to the present playlist so that the chatbot can return a new recommendation list based on the updated playlist.

Due to the limited time for this project, we have not realized these functions yet while those who are interested in this field can certainly try improving the project in these ways.

# Appendix 1: Project Proposal

**Date of proposal: 02/10/2022**

**Project Title: Spotify Music Recommendation Chat-Bot**

**Group ID (As Enrolled in LumiNUS Class Groups): Group 10**

**Group Members (name , Student ID):**

**Shi Qirui                A0261896M**

**Chen Huizhou   A0261843A**

**Tian Hengyi        A0261841H**

**Tang Junbo        A0226717B**

**Sponsor/Client:** (*Company Name, Address and Contact Name, Email, if any*)

Institute of Systems Science (ISS) at 25 Heng Mui Keng Terrace,  
Singapore

NATIONAL UNIVERSITY OF SINGAPORE (NUS)

Contact: Mr. GU ZHAN / Lecturer & Consultant

Telephone No.: 65-6516 8021

Email: zhan.gu@nus.edu.sg

### **Background/Aims/Objectives:**

Spotify, as one of the most well-known digital music services, is widely used by music lovers worldwide. It is a powerful platform as people can find almost all songs they want in its music library it provides. There is no doubt that Spotify is one of the most popular music platforms all around the world.

In recent years, chat bots are becoming a popular human-computer interaction application by which people can customize their needs in an intuitive way. An increasing number of services are adding chatbot modules to improve user experience and the development of chatbot is a trend of the era.

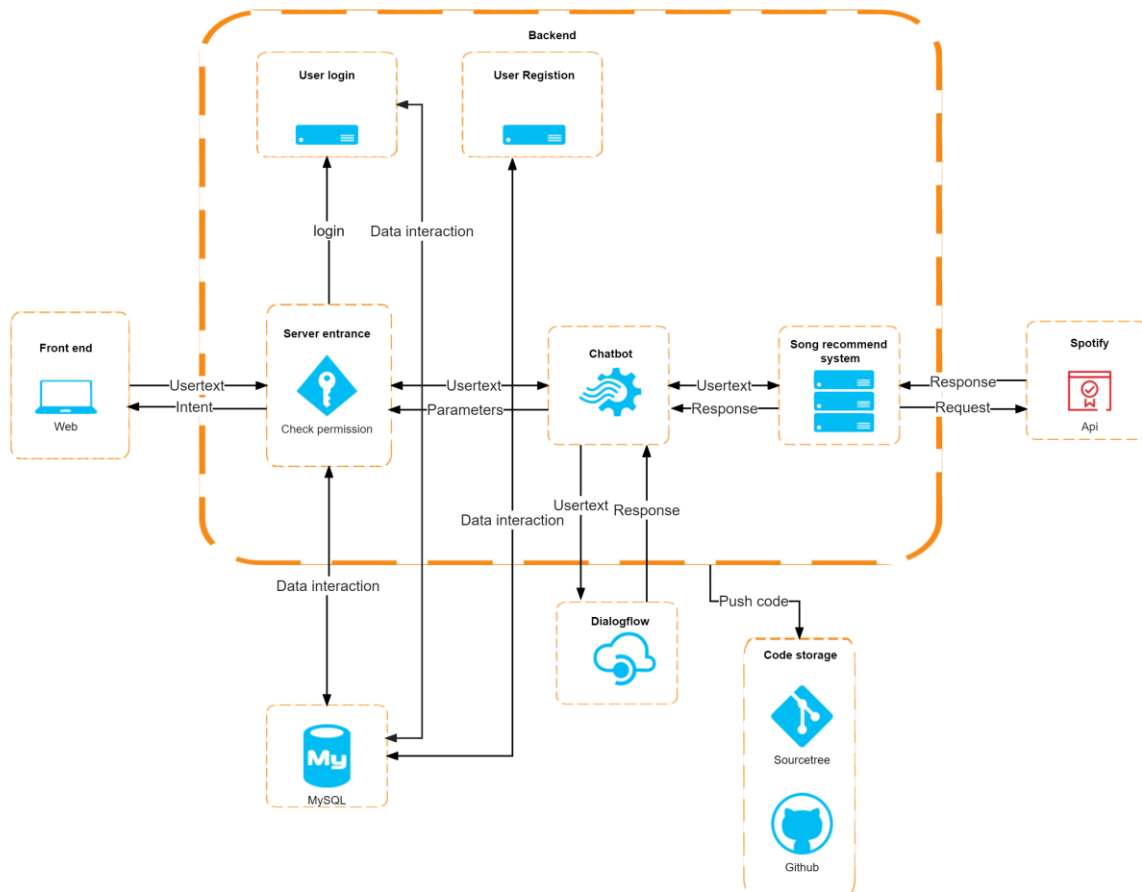
In this project, our team aims to create a chat bot which can respond to the user's input and perform interaction with the user's Spotify account. It can help them find songs they want, play a demo, and make recommendations. Users can talk with the chat bot, ask it some questions and ask for a recommendation for music in the Spotify library based on the artist's name, music genre or artist genre. The chat bot is also capable of obtaining a user's playlist on Spotify and making recommendations based on it. A HTML frontend is created for the chat bot and backend is



built based on the Flask framework. This chat bot only considers music from 2000 to 2020 included in the Spotify music library so it is still limited to some extent. Nevertheless, it still has potential to be further developed in functions and databases in the future.

## Project Descriptions:

### 1. System Architecture:



Precisely, we are going to build a complete system consisting of front end and back end. The overall structure is demonstrated in the flow chart above and here the detailed description will be given.

When a user logs in to the website and jumps to the chatbot page, the system first checks whether the user has login permission. If the user does not have the permission, he will be led to the login page. In this case, the user can choose to either log in or register.

Secondly, the dialogue content is sent to the back end for processing and then sent to Dialogflow which is deployed on google cloud platform to get intents and parameters.

After that, the song-recommend-system starts using recommendation algorithm-- cosine similarity to turn the feature songs in the playlist into a vector for comparing the vector with all other songs in the database and call spotifyApi to get the relevant music information. e.g. the preview of the song and so on.

Finally, the back end packages the important data and sends it to the front end for display and further interaction.

## ***2. Data Acquisition:***

Strictly speaking, there are two databases for our project: the database of songs recorded in Spotify from 2000 to 2020 (a csv file) and other data(like detailed information of songs) we are going to show to the user from the Spotify API.

The songs in our pre-processed data base only contains the songs from 2000 to 2020 obtained from Spotify music library. We will pre-process the database so that it will meet the system's requirements.

## ***3. Resource Requirements:***

Hardware : Any device with browser

Software : Song recommend system project

- Spotify developer account

- Spotify App with client\_id and client\_secret

- GOOGLE\_APPLICATION\_CREDENTIALS private key

- dbtest.sql

## Appendix 2 Mapping Functionalities

Mapped System Functionalities against knowledge, techniques, and skills of modular courses

Machine Reasoning	<ul style="list-style-type: none"><li>· <b>Built our own knowledge base:</b> Built a database with MySQL to store the users' registration information</li><li>· <b>Big data, data mining:</b> We extract data from raw databases, as the learnt knowledge for the following reasoning operation.</li></ul>
Reasoning System	<ul style="list-style-type: none"><li>· <b>Reasoning:</b> we use a recommendation algorithm to perform reasoning from acquired knowledge (processed data) and new input information (user's playlist). And the output of reasoning is the result of recommendation.</li></ul>
Cognitive System	<ul style="list-style-type: none"><li>· <b>Natural language processing:</b> We used natural language processing techniques as the cognitive part of our system, with the help of Dialogflow, to handle the text sent by user (receive new information)</li></ul>

## Appendix 3: Reference

- 1.<https://en.wikipedia.org/wiki/Spotify>
- 2.<https://www.midiaresearch.com/blog/music-subscriber-market-shares-q1-2020>
- 3.<https://www.weforum.org/agenda/2020/05/this-is-how-covid-19-is-affecting-the-music-industry/>

# Appendix 4: User Guide

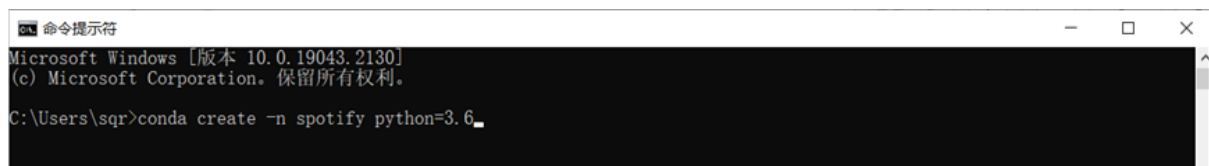
## 1. Install Python and Packages

Go to <https://www.python.org/> download the installer. Follow the installation guide. The preferred python version is 3.6.

Alternatively, you can use Anaconda to install the python, you can follow this link: <https://docs.anaconda.com/anaconda/install/index.html>. This guide prefers Anaconda as the tools.

If you are using Anaconda, please open the CMD window and enter conda commands:

**conda create -n spotify python=3.6**

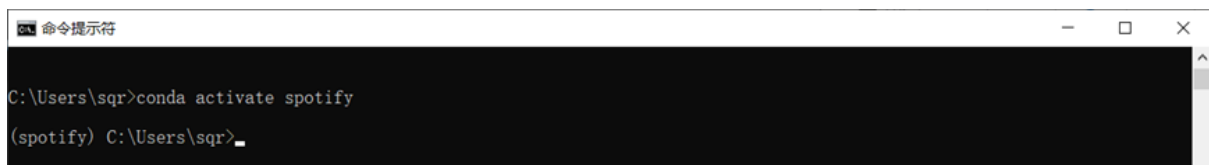


```
命令提示符
Microsoft Windows [版本 10.0.19043.2130]
(c) Microsoft Corporation. 保留所有权利。

C:\Users\sqr>conda create -n spotify python=3.6 _
```

After creating a new python environment named spotify, please activate this python environment by entering following commands:

**conda activate spotify**



```
命令提示符

C:\Users\sqr>conda activate spotify
(spotify) C:\Users\sqr>
```

Change the directory to the source code folder, using

**cd path to source code folder**

Then install the required packages by using requirements.txt file.

**pip install -r requirements.txt**



```
命令提示符 - pip install -r requirements.txt

(spotify) D:\Project1>
(spotify) D:\Project1>pip install -r requirements.txt
```

## 2. Download datasets

You can download the datasets for our project through google drive link below:

[https://drive.google.com/drive/folders/1Mi5tgIvZXmy-N2SPZSqPR8bPAICR\\_teU?usp=sharing](https://drive.google.com/drive/folders/1Mi5tgIvZXmy-N2SPZSqPR8bPAICR_teU?usp=sharing)

Please put the datasets (draft.csv and complete\_feature\_set.csv) to the route of our project.

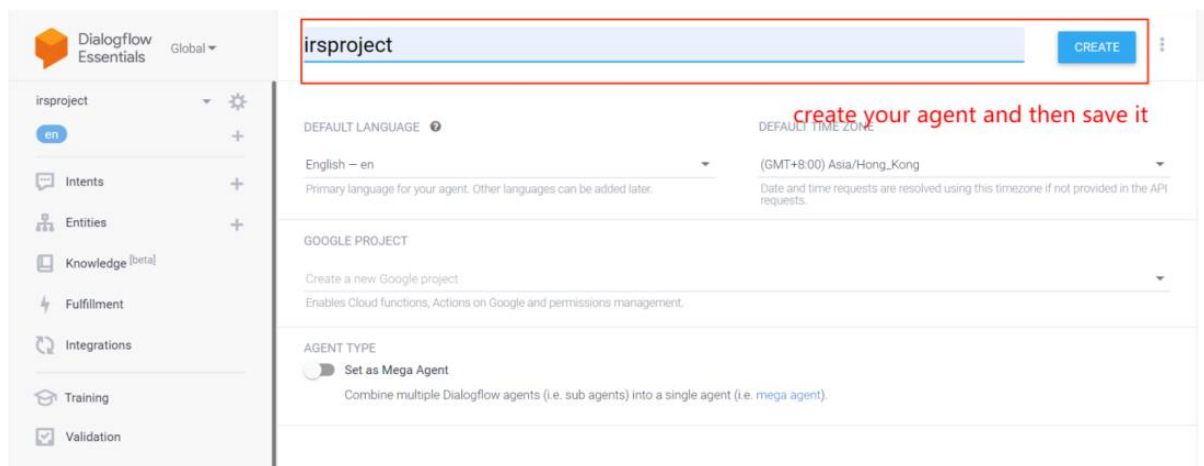
## 3. Import the agent from Dialogflow

Now you should import the agent given by us and check the intents included. It would be helpful to figure out the use case that different intents represent in this way.

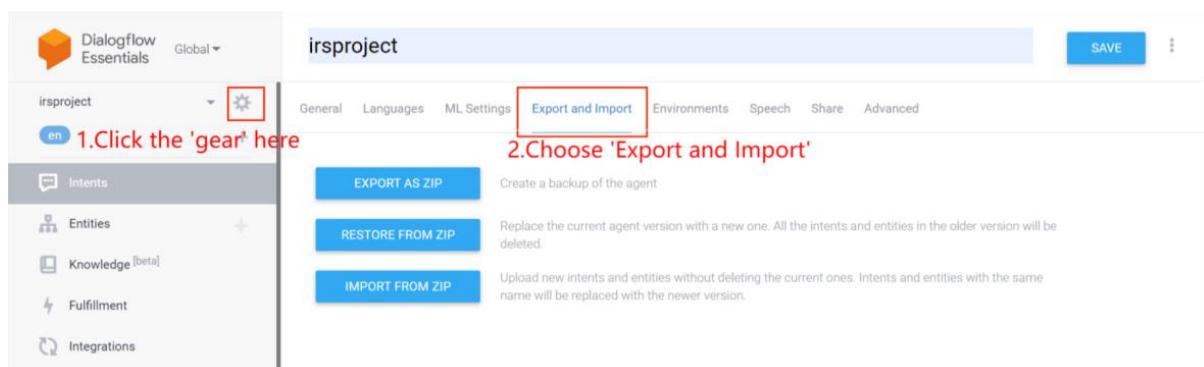
1. Create an account on Google Dialogflow

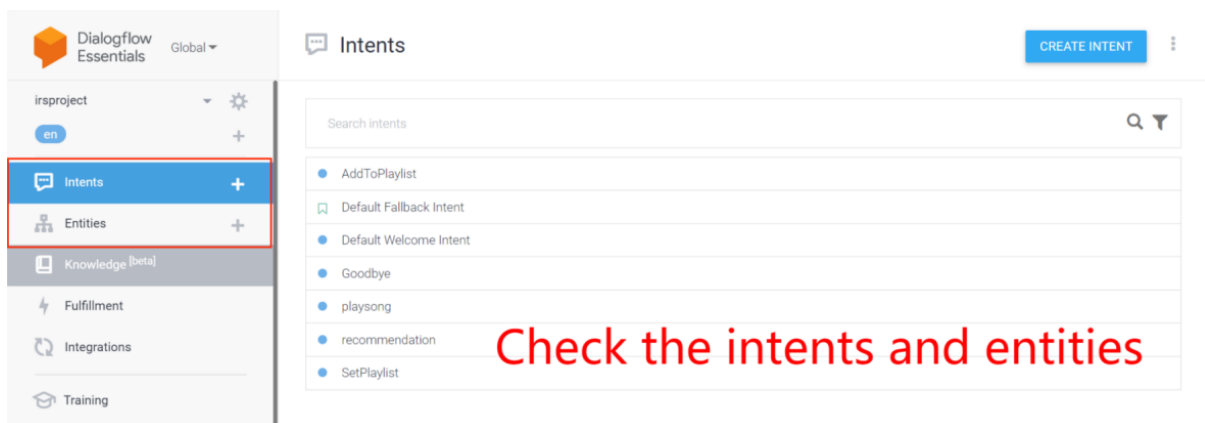
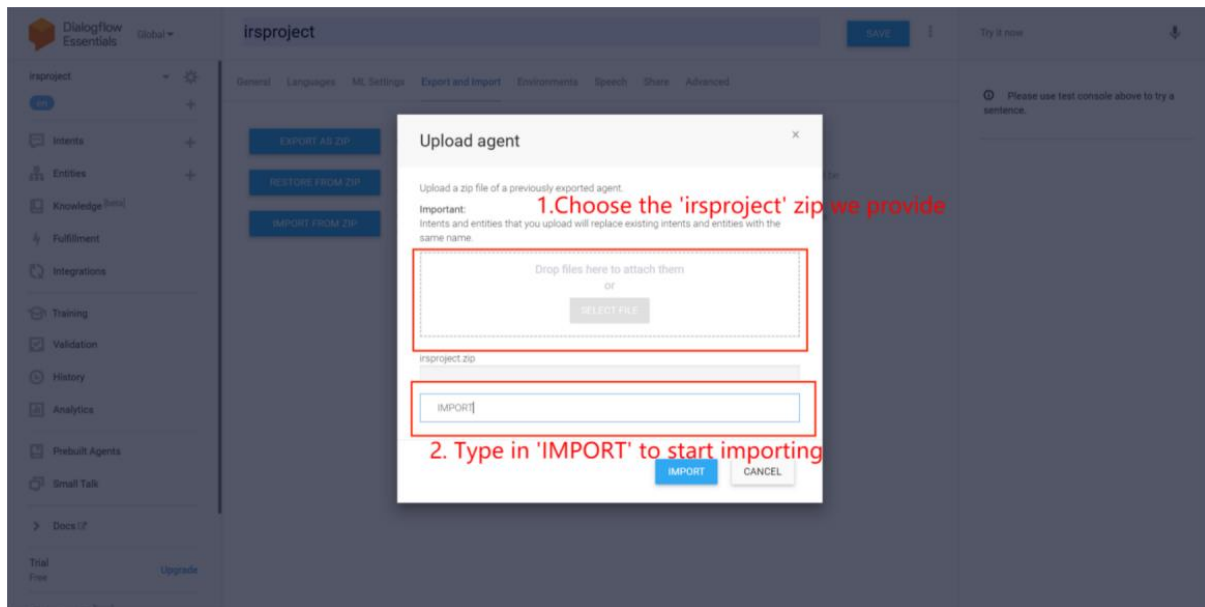
(<https://cloud.google.com/dialogflow>)

2..Visit Dialogflow, create a new agent and name it as the name of your project.



3.Import the zip file we provide named 'irsproject', check the intents and entities.



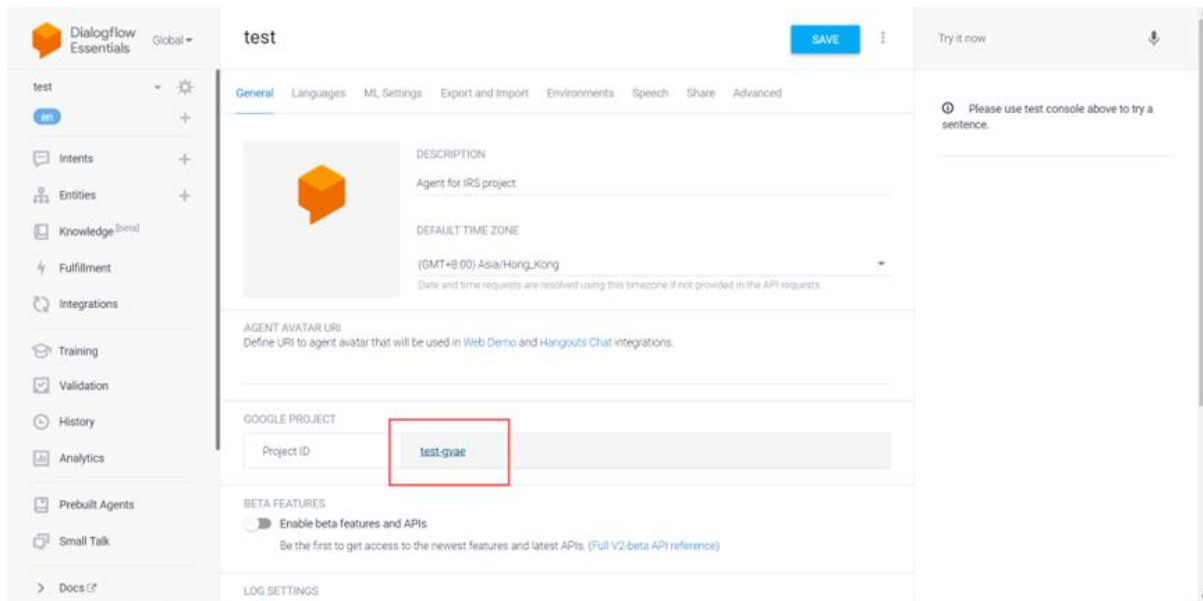


## 4. Get a GOOGLE\_APPLICATION\_CREDENTIALS private key

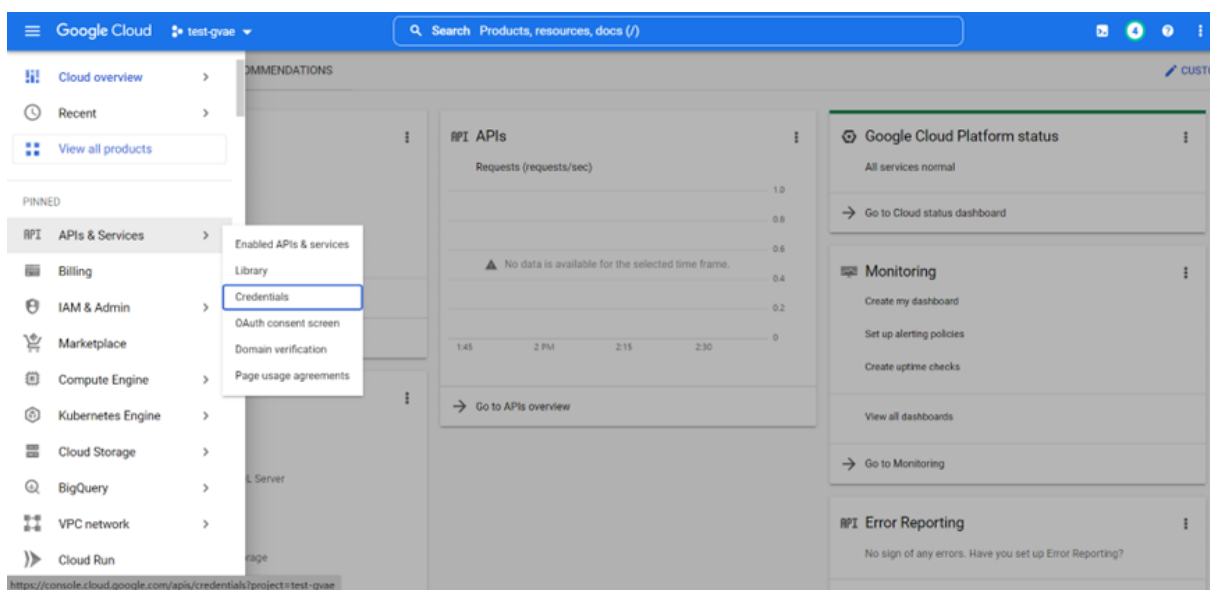
After importing the agent into Dialogflow, you need to obtain a private key to access this Dialogflow agent.

1. click the project ID under Agent -> General and you will go to the Google Cloud Console for this project:

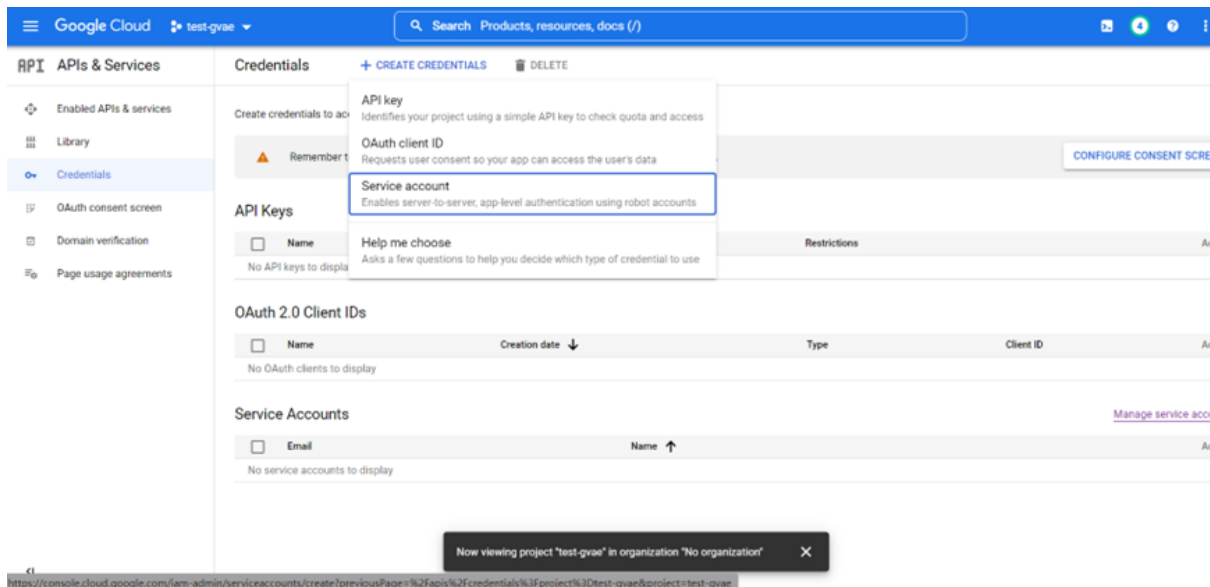




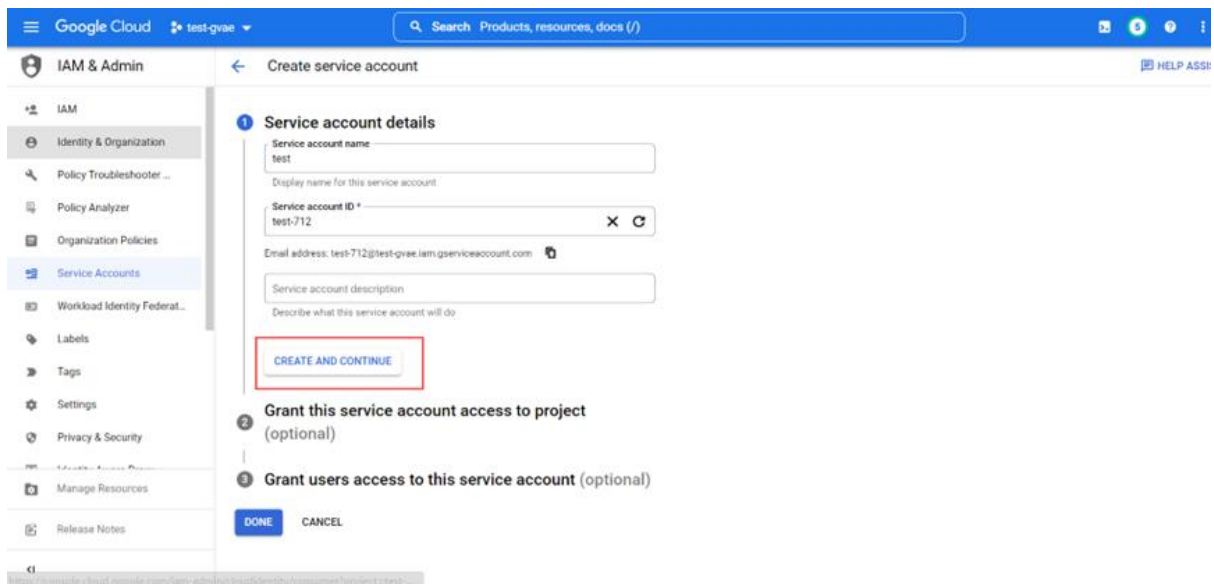
2. Navigate to Navigation menu-> APIs & Services->Credentials:



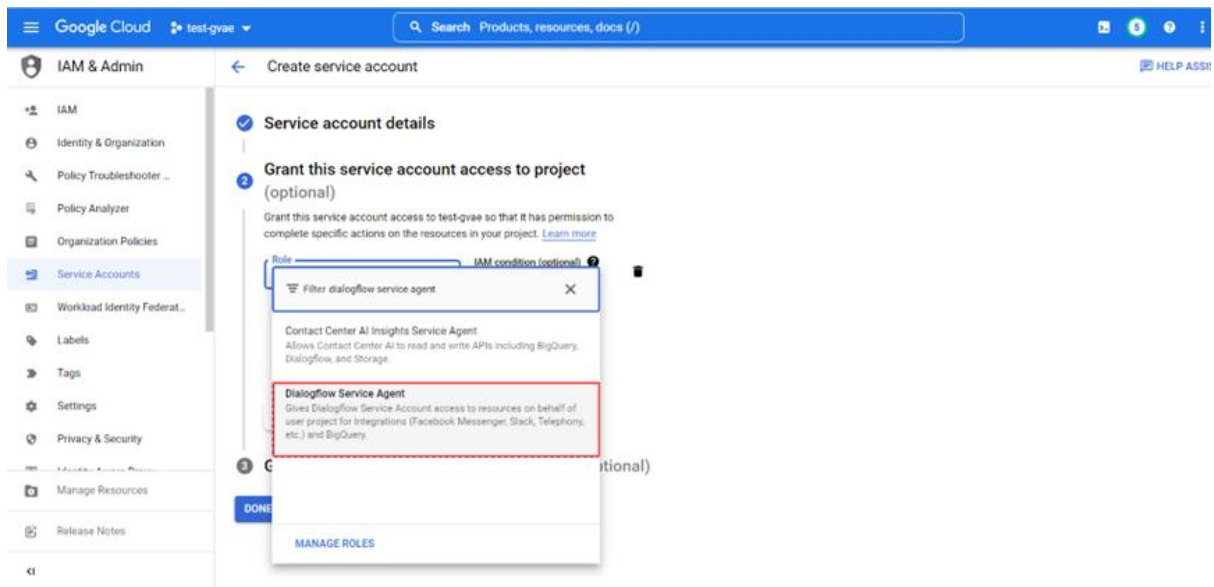
3. Click + CREATE CREDENTIALS->Service account:



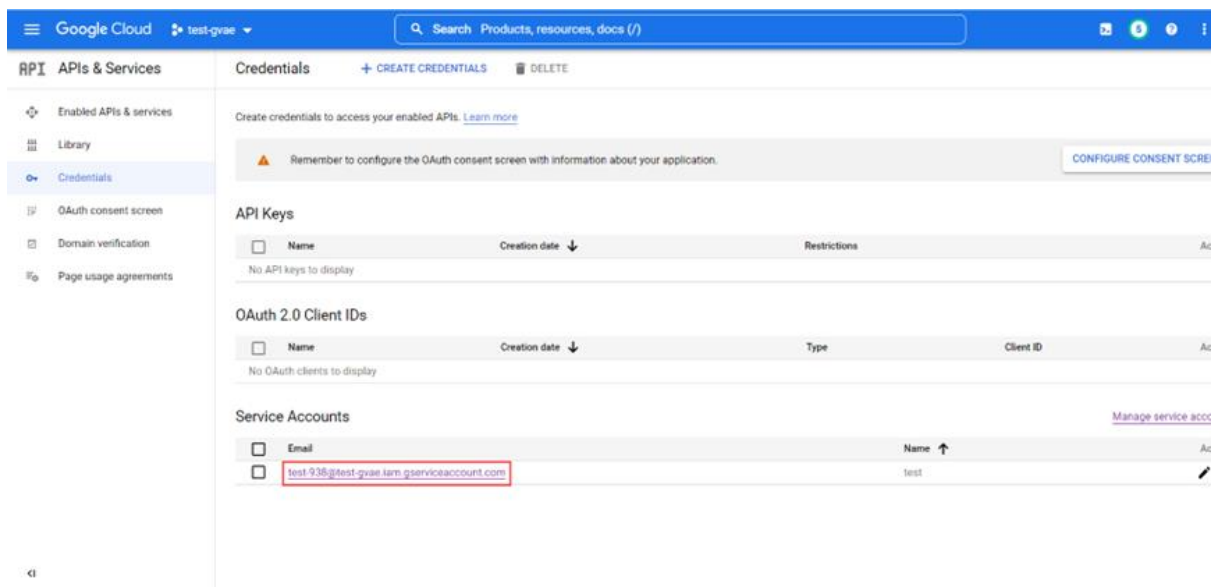
4. Input “service account details”, then click CREATE AND CONTINUE:



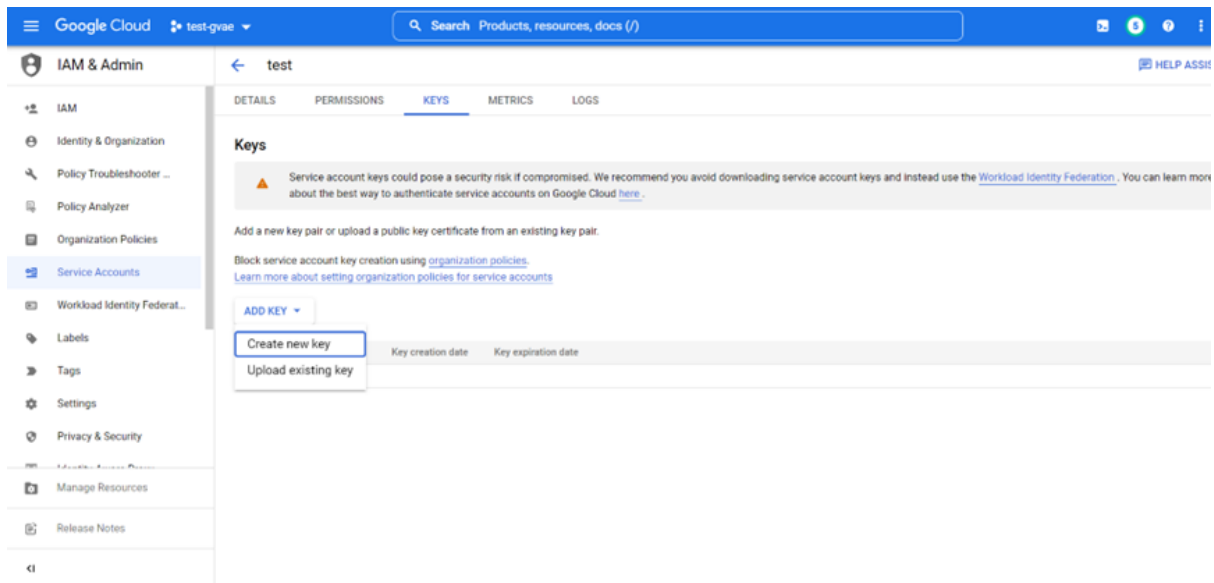
5. Grant this service account access to project, select a role->choose “Dialogflow Service Agent”, then click “DONE”:



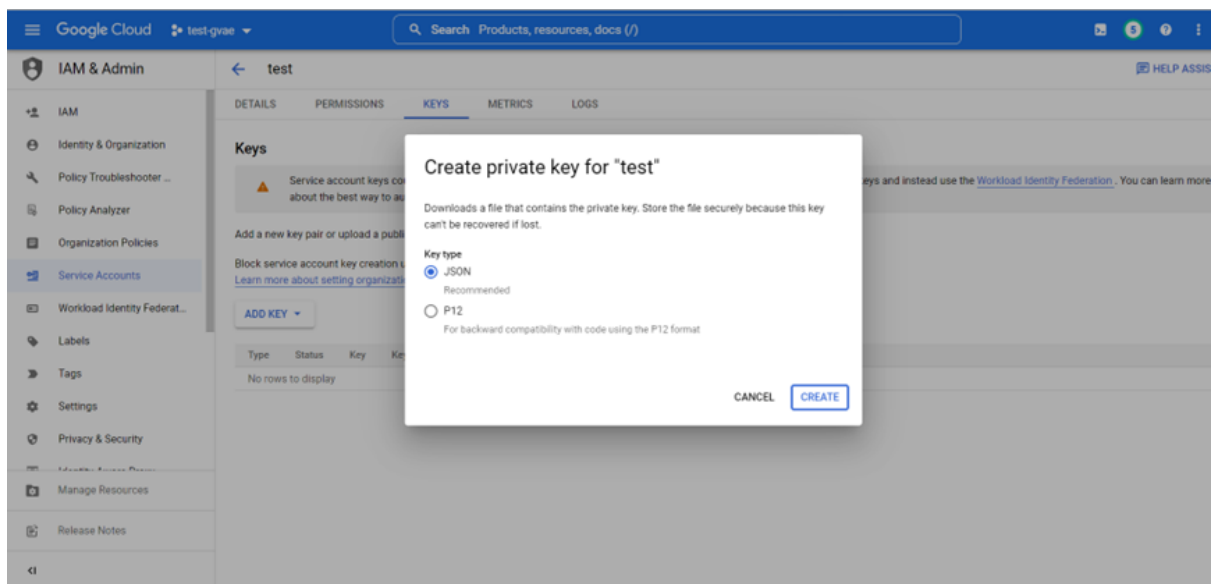
6. After creating a service account, you will your account here, click the email address, you will get into your account:



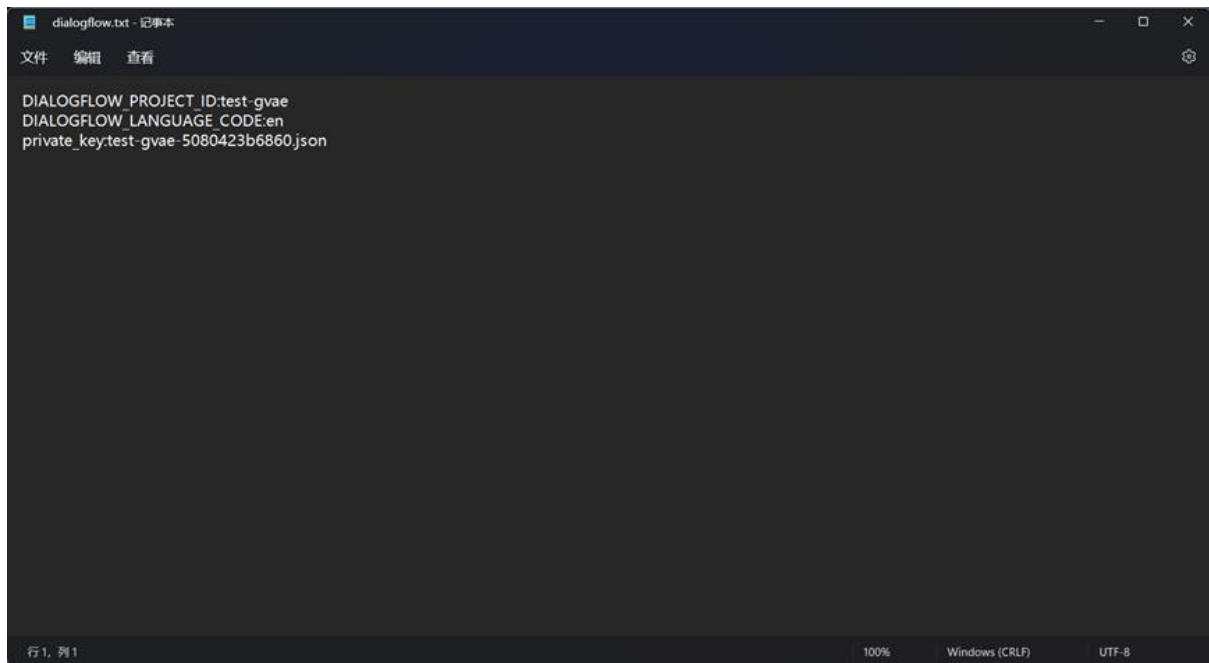
Go to KEYS, then click ADD KEY->Create new key:



Set key type to “JSON”, then click CREATE, the json file will be downloaded to you PC, then move it to the root dir of the project:

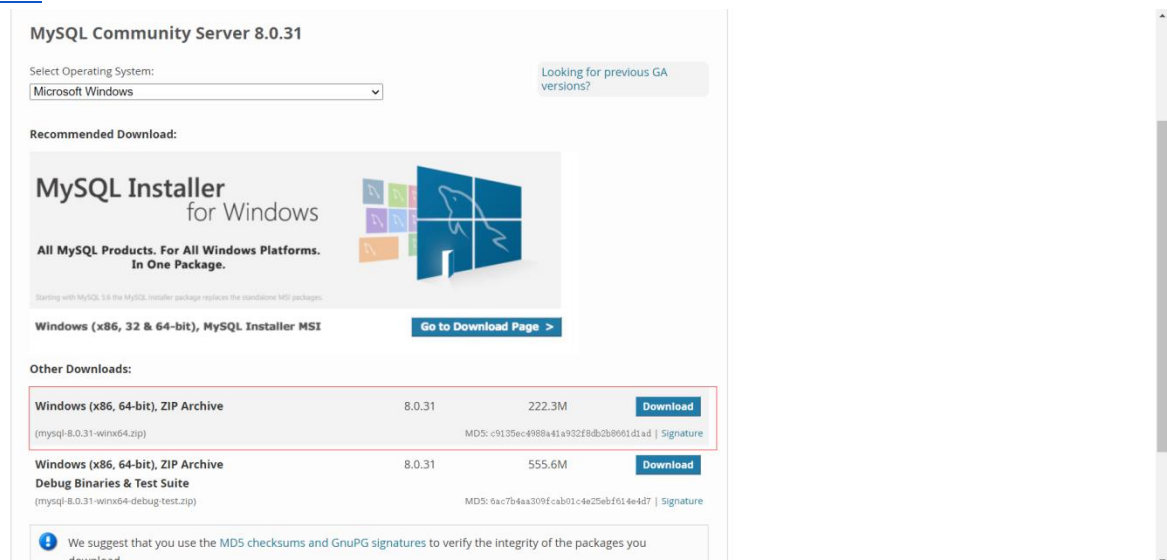


7. update your key information in dialogflow.txt as shown below:



## 5. Database setting flow:

1.Install MYSQL: View the latest version and download it [MySQL :: Download MySQL Community Server](#)



2.After download you can follow this guid [MySQL Installation on Microsoft Windows \(w3schools.com\)](#)

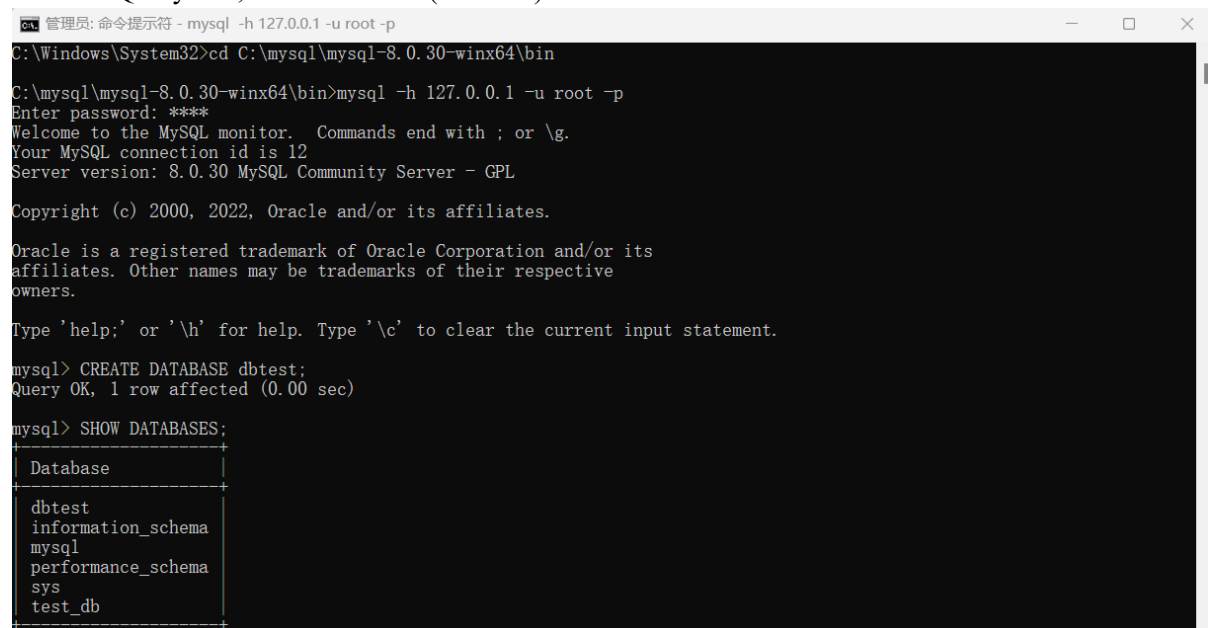
3.After installing and setting Mysql, Click”search”→”cmd”, use right click to choose Administrator statue.

4.Open the prompt and enter the command:” cd C:\mysql\mysql-8.0.30-winx64\bin”(need to replace your root)

5. You can run the login command on the prompt :mysql -h 127.0.0.1 -u root -p Connect to the MySQL database.

6. After the login is successful, the MySQL initial page is displayed “Welcome to the MySQL monitor”

7. Use command to create the database: mysql> CREATE DATABASE dbtest; And it will shows :”Query OK, 1 row affected (0.00 sec)”



```
管理员: 命令提示符 - mysql -h 127.0.0.1 -u root -p
C:\Windows\System32>cd C:\mysql\mysql-8.0.30-winx64\bin
C:\mysql\mysql-8.0.30-winx64\bin>mysql -h 127.0.0.1 -u root -p
Enter password: ****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 12
Server version: 8.0.30 MySQL Community Server - GPL

Copyright (c) 2000, 2022, Oracle and/or its affiliates.

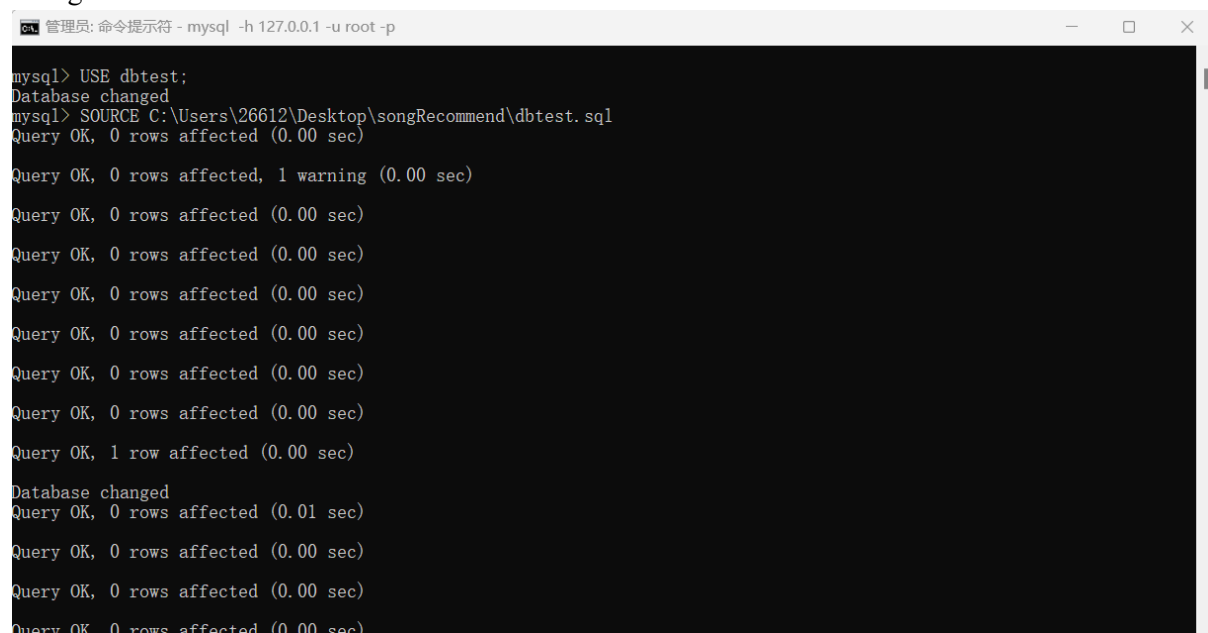
Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> CREATE DATABASE dbtest;
Query OK, 1 row affected (0.00 sec)

mysql> SHOW DATABASES;
+-----+
| Database |
+-----+
| dbtest   |
| information_schema |
| mysql    |
| performance_schema |
| sys      |
| test_db  |
+-----+
```

8. Use command to choose the correct database: mysql> USE dbtest. And it will shows :” Database changed”



```
管理员: 命令提示符 - mysql -h 127.0.0.1 -u root -p

mysql> USE dbtest;
Database changed
mysql> SOURCE C:\Users\26612\Desktop\songRecommend\dbtest.sql
Query OK, 0 rows affected (0.00 sec)

Query OK, 0 rows affected, 1 warning (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

Query OK, 1 row affected (0.00 sec)

Database changed
Query OK, 0 rows affected (0.01 sec)

Query OK, 0 rows affected (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

Query OK, 0 rows affected (0.00 sec)
```

9. Use command to run .sql file mysql> SOURCE C:\Users\26612\Desktop\songRecommend\dbtest.sql

```
管理员: 命令提示符 - mysql -h 127.0.0.1 -u root -p
Query OK, 0 rows affected (0.00 sec)
Query OK, 1 row affected (0.00 sec)
Database changed
Query OK, 0 rows affected (0.01 sec)
Query OK, 0 rows affected (0.00 sec)
Query OK, 0 rows affected (0.00 sec)
Query OK, 0 rows affected (0.00 sec)
Query OK, 0 rows affected (0.00 sec)
Query OK, 0 rows affected (0.00 sec)
mysql> DESCRIBE user;
+-----+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| id     | int  | NO   | PRI | NULL    | auto_increment |
| email  | varchar(100) | NO   |     | NULL    |                |
| password | varchar(100) | NO   |     | NULL    |                |
| name   | varchar(100) | NO   |     | NULL    |                |
+-----+-----+-----+-----+-----+-----+
4 rows in set (0.00 sec)
mysql>
```

## 6. Spotify API

Due to the change in usage of Spotify API posted in 2021, If a new user wants to authenticate with a third-party application, the developer needs to add the user in development mode manually, which means that users can not get the authorization of our application to access to Spotify API automatically when they are running the project. We can do nothing to avoid or skip the manual part. Therefore, there are two methods to access to the Spotify API:

- 1) Informing us for the authorization so that we could add your account in our application.
- 2) Create your own application and make the settings by yourself.

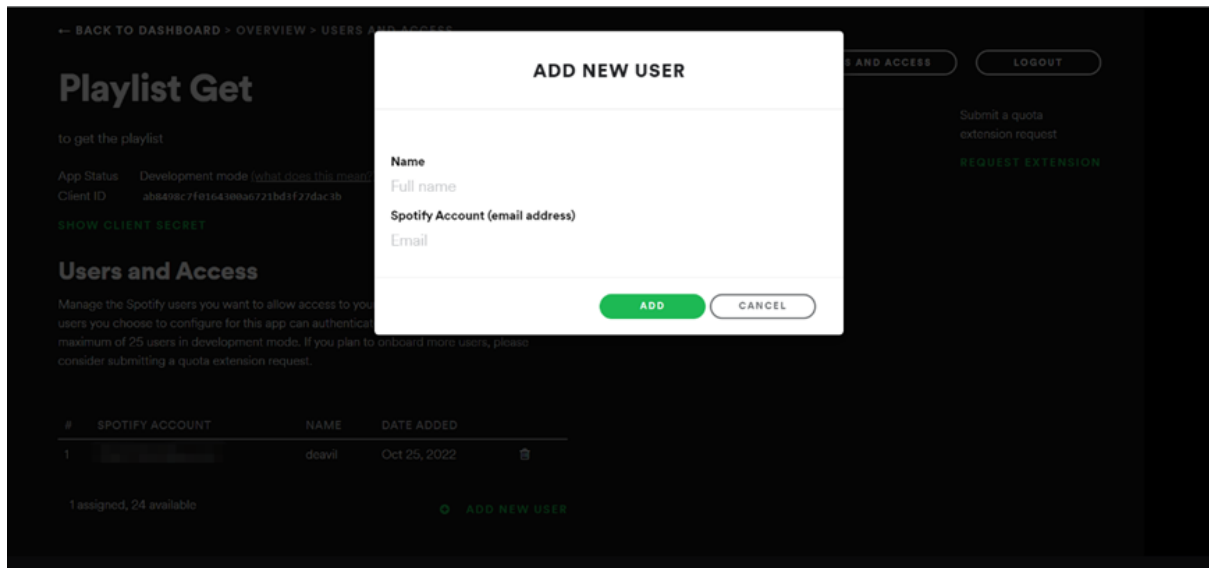
<https://developer.spotify.com/community/news/2021/05/27/improving-the-developer-and-user-experience-for-third-party-apps/>

Before continuing the following steps, please ensure that you have already had a Spotify account. If not, please go to <https://open.spotify.com/> and sign up for an account.

Here are the details of two mentioned methods.

### 1) Inform us

To authenticate your account with our application, we need your Spotify name and email address to add your account in development mode in the website of Spotify for Developer (<https://developer.spotify.com/>)

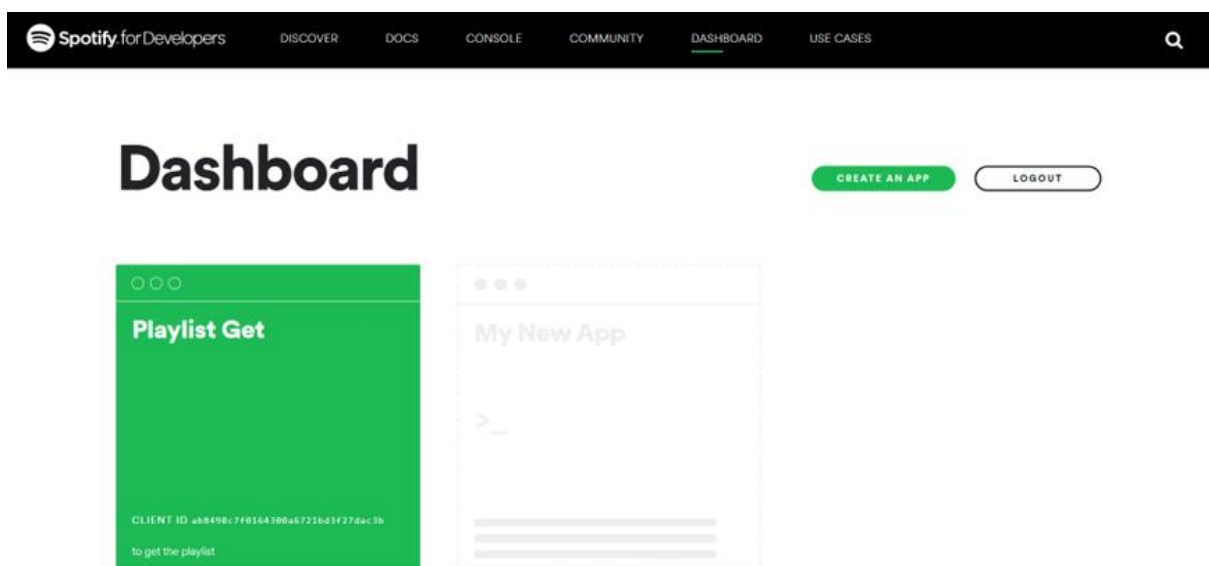


To be mentioned, the “Name” in “ADD NEW USER” is not the Username in your Profile. It is the name you set by yourself when signing up for the account.

If you choose this method, please email to our group by email address e0983290@u.nus.edu. Please state your intent in the title and we will add your account in development mode as soon as possible.

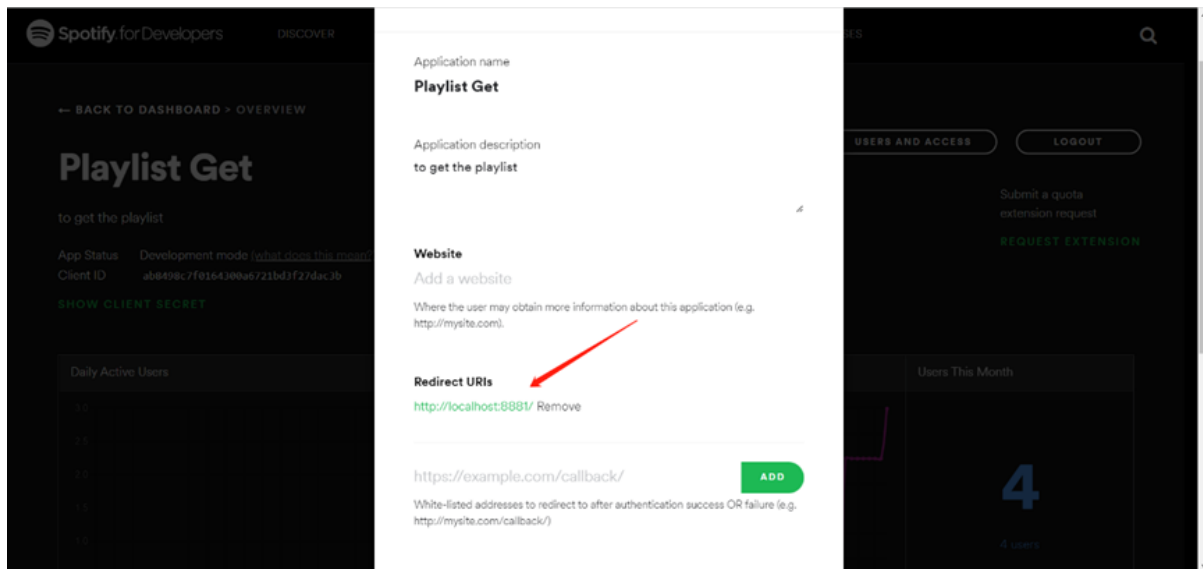
## 1) Create your own App

If you choose to create your own application, please use your Spotify account to sign up for Spotify of Developer. After that, please enter your “DASHBOARD” and create your own application by clicking “CREATE AN APP”.



Enter your application, you will see a website like this. Please click “EDIT SETTINGS” to change the Redirect URLs to http://localhost:8881/.





It needs to be mentioned that you can directly access the Spotify API using an application set up in your own developer account without adding your account into the User. If you want to invite your friend, please refer to the part 1 Inform us to add users in development mode.

If user account is not added in the application, the running error will be shown like this:

**User not registered in the developer dashboard, reason: none**

## 7. Final run

After setting all the requirements, open the command window and set root to your project location. Activate the python environment and type the “flask run”.

# Appendix 5: Individual Report–Shi Qirui

**Shi Qirui**

**A0261896M**

## **1. Your personal contribution to the project.**

In this project, I am mainly responsible for the authentication to our application using Flask in the front-end and the recommendation system design.

For the front-end, I participated in the authentication part of our system including user's sign in and login, whose personal information will be stored in the local MySQL database.

First, before designing the recommendation system, the suitable dataset is needed. The Spotify dataset was found on Kaggle, which seems to contain enough information our recommendation needed. Then, I did all the data preprocessing. I extracted artists' information and their genres from the artist dataset and merged it with the track dataset to generate a new dataset. Then, based on the new dataset, I used the TF-IDF method to obtain text-based genre features and abstracted the track information into feature vectors for the following recommendation. After that, I achieved the content-based recommendation with cosine similarity measurement.

Also, I integrated codes of group members into a complete version and participated in the debugging period.

## **2. What you have learnt from the project.**

I actually learnt a lot from this project. I started to get in touch with the front-end logic by using Flask and the back-end database as well, since I need to install MySQL and create a database in my own computer and then connect it with the front-end part. I became much familiar with the working logic between front-end and back-end.

What is else, this project stabilizes my recognition towards recommendation algorithms. Although the algorithm used in the project is very simple and easy (Content-based recommendation), I also considered other recommendation methods like collaborative filtering. Due to the limitation of the dataset (lacking rating and user information), the current algorithm was chosen.

The most useful thing I learnt from the project is the experience of engaging in almost all the process of an App creation including front-end and back-end.

### **3. How you can apply this in future work-related projects.**

Practically, the experience of handling large scale dataset and participation of whole knowledge modeling design is sure to be useful for my future. And my coding ability also got trained through this project.

What is more, this project experience helps me to become much more familiar with the whole development of an App instead of just focusing on a specific part of a project. Also, the team collaboration enhances my communication ability towards my group member, which helps me represent my own requirements and understand teammates' need throughout the development period.

# Appendix 6: Individual Report–Tang Junbo

**Tang Junbo      A0226717B**

## **1. Your personal contribution to the project.**

In this project, I'm mainly responsible for the framework construction. Because I've been working on the back-end development for one year. I'm responsible for the framework construction of Flask, data transmission from the front end to the back end; front-end identity authentication including token storage of user information; Data transfer between database and backend; By creating Google's service account, the backend is connected to dialogflow.

Discuss with team members and give data preprocessing solutions together, try to avoid some features with only a few songs, so that the data will not be divided into two discrete, and the data can be well integrated.

We also tested the function of spotifyapi together and analyzed the json file in the response to determine how to get the information we want. Since spotify's json file is a bit complicated, we spent some time testing preview and other content and trying to figure out how to play music in our system. Discussed what algorithm to use together. Based on the available data, we could not obtain the User rating and other information, so we did not adopt User Similarity.

I teach other team members how to use sourcetree so that we can synchronize the code with git at any time and keep the consistency of development.

## **2. What you have learnt from the project.**

First of all, I have never used the flask framework before, and this project enables me to have a good understanding of this microframework. Secondly, I have a good comparison and understanding of the current mainstream recommendation algorithms through this project. We discussed and picked the right algorithm, which made me understand that there is no best algorithm in the algorithm, but I need to choose the most suitable algorithm according to the current project. At the same time, in the process of connecting the backend and Dialogflow, I further learned about the development of Google Cloud Platform, and through testing and

investigating the apis of different websites in the early stage of the project, I further learned about related interactions and call mode.

### **3. How you can apply this in future work-related projects.**

First of all, I think about the data preprocessing and the site of the API planning is likely to work in my future, because we need to design functions according to the website of the API, in turn, we need to use useful data. Then I think that in dealing with Google cloud Platform certification, company certification also can be used in work to in the future

# Appendix 7: Individual Report–Tian Hengyi

**Tian Hengyi      A0261841H**

## **1. Your personal contribution to the project.**

During the journey of this project, the first thing I have done is to share the ideas of what to do and how to make progress with my teammates frequently. We discussed the models ,tools and database to be used in our project and did research on related techniques. I found cosine similarity a wonderful method to build a recommendation system and came up with the idea of making use of a whole playlist of users for recommendation together with Shi Qirui. After that, we found a suitable database of the Spotify music library and together we pre-processed it to meet the requirements of our system.

In addition, I have also created the intents to be detected by chatbot using Dialogflow. I have designed the significant intents for activation of a variety of functions of chatbot as well as some basic intents like welcome and goodbye to improve user experience.

Besides, I have used CSS to design the HTML frontend of our system, which also helps improving the system by improving user interaction experience.

## **2. What you have learnt from the project.**

From this project, I have acquired more about the use of Dialogflow as I need to design the intents and think about the possible requirements as inputs for the chatbot system from a user's perspective. Besides, it is my first time to design the frontend by myself and figure out the way to build the frontend and backend in a project. It was time-consuming but still fun and I chose the background picture, text style as well as the position of each block using CSS. Additionally, I did the pre-processing together with Shi

Qirui, which helps me better understand the importance of data pre-processing and how it will help your system. At the beginning, there were two csv files representing the music library of Spotify focusing on the ‘artist’ and ‘song’ separately. We worked together and successfully merged them into one csv file to be processed. Nevertheless, we found that there were too many features included in the database and some of which only occurs in low frequency. So, we decided to filter out some insignificant features and in turn did Dimensionality reduction. By decreasing the dimension we increased the calculation speed for the recommendation system and also reduced the size of the database, which was helpful for the whole system.

To discuss and try sharing some ideas with my teammates is a must in excellent teamwork. This project taught me about how to do research on a certain topic efficiently. I tried to search for useful databases as well as recommendation algorithms and shared my breakthrough with my teammates.

### **3. How you can apply this in future work-related projects.**

Firstly, I am now quite familiar with some basic techniques to pre-process databases. I believe that I will be able to process the dataset obtained by myself easily in the future. Now I also have a better understanding on which method to use to make my database meet the requirements of the project.

Besides, I have a better understanding of Dialogflow now. It is a helpful tool which can be applied in many cases related to user interaction in AI projects nowadays and to be proficient in using Dialogflow to do the project improves me a lot.

Finally, the knowledge I learnt from UI design helps me understand the function of the frontend better. Now I believe that I can make a beautiful UI in projects in the future by myself and the basic knowledge I learnt about frontend as well as backend would benefit me a lot in the future.

# Appendix 8 Individual Report – Chen Huizhou

**Chen Huizhou A0261843A**

## **1. Personal Contribution to Group Project**

This project experience is very valuable to me. In this project, I not only practiced what I learned in Intelligent Reasoning System courses, but also self-studied some knowledge not involved in class, both of which helped me complete this project and implement planned results and functions.

Before we started this project, we discussed together to generate this idea, and I took part in the discussion, gave some ideas and came up with some implementation methods.

During developing the project, I am mainly responsible for the backend coding and Dialogflow agent building and connecting. For Dialogflow agent building part, Tian Hengyi and me both took part in designing the intents, and the corresponding training phrases. For the backend coding part, I wrote code to process the users' input text, and forward that to Dialogflow and get information like an intent from it. After that, the code would handle different intent and implement corresponding operations like calling recommendation algorithm to make a recommendation. Last but not least, I also implemented calling the Spotify API, and grab information from the API response, which can be used by other operations like returning a preview of a song to the user.

## **2. what learnt is most useful for you**

I learnt various things in developing this project. Since it is my first time actually using a flask frame and Dialogflow in a project, I did encounter lots of problems and gain useful experience.

Firstly, I learnt how to build a backend system under flask frame. When I got started, I felt a little bit frustrated because it is difficult to make everything work as I expected with a frame I have never used, especially when I tried to implement different functions under each route. After



searching for tutorials for flask and referring to some flask programmes, I managed to make things work gradually, then implement some more complicated logic.

Besides, I learnt how to build a decent Dialogflow agent and how to reach this agent with the API provided by Dialogflow. Before actually using this agent, I searched a lot about how to design an agent and spared no effort in learning the concept of each part of Dialogflow like intents, entities. We firstly designed some intents according to the expected function of our chat-bot. And for the connection part, at the very first, I decided to use Ngrok to generate a fulfillment for Dialogflow, but it was kind of troublesome since you need to run a server every time you try to communicate with Dialogflow. Then, I decided to use a python package provided by Dialogflow and create a client session with python code every time, in which case it will only be troublesome the first time you implement it, because you need to get a private key for it from Google.

What's more, I also gain experience in manipulating APIs with python, and grab the useful information from the response of the API. It could be challenging because the response of some APIs is an extremely complicated json file, which contains lots of data and text information.

### **3. How you can apply this in future work-related projects.**

Firstly, after accomplishing this project, I learnt the core techniques of building a chat-bot. I gain experience in using tools and techniques like Dialogflow and flask frame. I could use this ability to build a chat-bot for other demands, such as an intelligent agent which could respond to the questions from customers.

Meanwhile, I got the ability to acquire information and data from APIs, in this case the Spotify API and Dialogflow API. It means in other projects which might have more complicated demands that need to obtain information from some website or database, I have the ability to realize that through APIs.