

RelaxBot

Project Report

[***1***](#_gjdgxs) ***Overview 1***

[**Architectural Graph 1**](#_30j0zll)

[***2***](#_1fob9te) ***Function specification 2***

[***3***](#_3znysh7) ***Third-party functionalities reference 3***

[***4***](#_2et92p0) ***Implementation challenges: 3***

[***5***](#_3dy6vkm) ***User documentation 3***

[**Operating Environment 3**](#_1t3h5sf)

[**Using Process 3**](#_4d34og8)

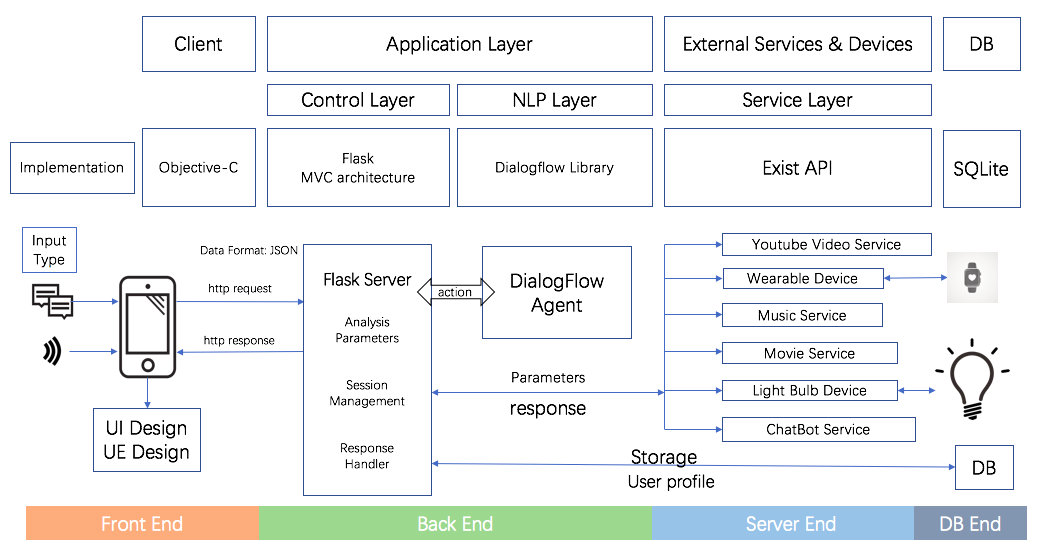
[***6***](#_2s8eyo1) ***Reference 4***

➢ ▪Overview - Architecture / design of the overall system & functionalities (for a joint group project, this section will be common to each group).

# Overview

The overall system is a chatbot-like app which based on Dialogflow and trying to provide some relax function for users. Our group notice that nowadays time is a significant precious resource, when they want to know something or doing something that prefer to let the machine do it for them by a simple voice command. So our project goal is to create a Chatbot app that integrate multiple functions and services to help users to achieve easy control.The app can provide functions like Multimedia player service, which include music service and movie information query. Interact with smart IOT devices (LIFX bubble in this project) and health monitors (Apple watch in this project).

## Architectural Graph



➢ ▪Descriptions of the functionalities developed by the team (and why not – the justifications of the functionalities that are missing/different from the project proposal).

# Function specification

1. **Multimedia player service**, include movie information query, music service
   1. **Movie information query** provide function that allows user to gain the movie information include movie plot, actors, directors, writers, released year, rating, trailer and what country it belongs.

**Use case example**: User can obtain those information by simply type in or a voice command such as “tell me some information about Spider-man” or “who is the actor of Green Book”.

**Notice**: What may occur here is that user may ask two movies or two types of information at a time. If two movies were asked, we return a warn message says “one movie at a time”. If two kinds of information were asked, we will only return second one. For example, if “director and actor of spider-man” was asked, we will only return the information about actor of spider-man.

For some possible reasons like duplicate movie name or inaccurate movie name user provided, if the movie information delivered is not user expect, RelaxBot can do fuzzy search for the movie title in database and provide the search result then re-provide the information of correct movie according to the user feedback.

**User case example**: user may want to gain the information about “2 Fast 2 Furious”, which is the second movie in a series of “Fast & Furious”. But some of them may not remember the name correctly, so by querying “information about Fast & Furious”, we will return a message like “is this the movie you expected?” and provided top 25 most likely movie to user to select if negative response was received.

**Notice:** what may occur here is that if the movie provided in the first was not in the database that we can’t find any result. Our dataset contain 18 thousands of movie so that may not occur often. Besides if the movie name provided was too far away from the original one we can’t find what you want either. Moreover, if the index selected is beyond the scope of list, a warning message will return.

* 1. **Music Service** provide function that allows user to select tracks from the album or from user’s own playlist and play on our build in music player. The music player function include pause, stop, next, previous and the way of playing the list like play track, loop track, loop playlist, randomize.

**User case example:** by querying something like “I want to hear a song of Ed Sheeran ” that specify what is wanted is the “songs”, we will return a playlist that contain 10 most hit songs of Ed Sheeran for user to choose. Besides, any query like “any albums of Ed Sheeran?” will receive a list of 10 most heat albums of Ed Sheeran, where user can play a chosen album or enter the album and play a specific song in it.

**Notice:** what may occur here is that the index selected is beyond the scope of list, if so a warning message will return.

To control the music we provide two ways. One way is to interact with our build-in music player. Another way is by type/voice command like “play/stop the music”, which we didn’t implement it as we proposed. Because when we developing this function, we found it’s easy to mix up this Control intent with Play Song inten. since when user say “play the song”, Dialogflow will sometime jump into Play Song intent and treat “the song” as a name of singer.

1. **Health monitor function** allow user extract their personal information from their wearable devices(Apple Watch in this project). We provide information include max/min heart rate, walking distance, steps in real time/specific time, calorie consumption in real time/specific time and sleep analysis.

**User case example:** real time information will be returned if user querying ”how many steps I walk”. If querying like “how many calories did I burn last week”, the total consumption last week will be returned.

**Notice:** the heart rate will only return the maximum and minimum heart rate of the given date.

1. **Interact with smart home devices function** allow the app connect with the smart device and control it by voice or type-in command, in this project we develop with the LIFX bubble. We provide a easy way for user to control the light on-off, change the color tone, color temperature and brightness easily. Except from the 9 build in color LIFX provided, we add another 151 colors like “midnight blue”, “sienna” and so on.

**User case example:** User can control the light by simple command like “turn on/off the light”, “switch to red”. Commands like “turn down the lights” will decrease the brightness by 10% and if any specific percentage in the command, for example, ”dim the light by 50%” then it will decrease the brightness by 50%. Relevant response message will be return to user if the command is successfully execute.

**Notice:** if the color user wanted to set are not in our scope, then we can’t do anything but return a warning message that “light can’t set to that color”.

➢Proper references and brief descriptions of ALL third-party functionalities (clouds/services/APIs/libraries/code) used by the team, with justification for their use and discussion how their licensing terms impact results of this project.

# Third-party functionalities reference

1. **Dialogflow**: mainly used to detect intents and entities. Sentence uploaded to Dialogflow can be divided into intents and entities, which can help us find out what users want easily. For example, when user asking “who is the actor and director of spider-man”, Dialogflow can tell us this query is about movie, this part is done by intents. Also it can tell us the information about director, actor and spider-man is needed, this part is done by entities. By doing this we can extract important information from a query and give proper feedback.
2. **LIFX API:** used in Interact with smart home devices function, allow us control the LIFX bubble in our own app
3. **HealthKit**: used in Health monitor function, allow us gain the information from Apple Watch.
4. **YouTube Video API:** used in Movie information function to obtain the movie trailer, allow us search the relevant trailer in YouTube and gain the thumbnail and video id. The licensing terms of YouTube API only allows at most 100 queries per day, so if the query about trailer are extend we can’t obtain result any more.
5. **NLTK:** used in movie query function for providing more accurate search results.
6. **Flask**: it’s the foundation of our application. It helps us easily handle http requests and responses.
7. **Spotify-python**: used in the front and back end to provide user music control, music display, music query, 3rd party authentication and token refresh.
8. **MongoDB-python**: interact with the back end to query, insert, modify, delete the data saved in MongoDB.

➢Implementation challenges: descriptions of any tricks, non-trivial algorithms, special architecture/design, etc.

# Implementation challenges:

1. Data Process

Movie part:  
Nowadays, movie information can not be presented accurately by only a movie title. There are lots of remade and reproduced movies in the market and some movies have same title. To make our project more close to real product and more accurate, We grab the movie data from the IMDB database, delete unnecessary information and save in our database and dialogflow intent so that as long as your input is a real movie name, our product will provide the results as close as possible.

Music part:  
Music can not be same as Movie. There are only at most 20 thousand movies since the birth of movie, but the number of music is at least 10 times than movie’s. Thus, we cannot save all musics in the database and dialogflow. Our strategy is to collect the top one thousand artist in the history and recent top 100 artists in the billboard that could cover most users. We can provide user the albums and top ten songs through the artists, meanwhile give user’s playlist in their Spotify account.

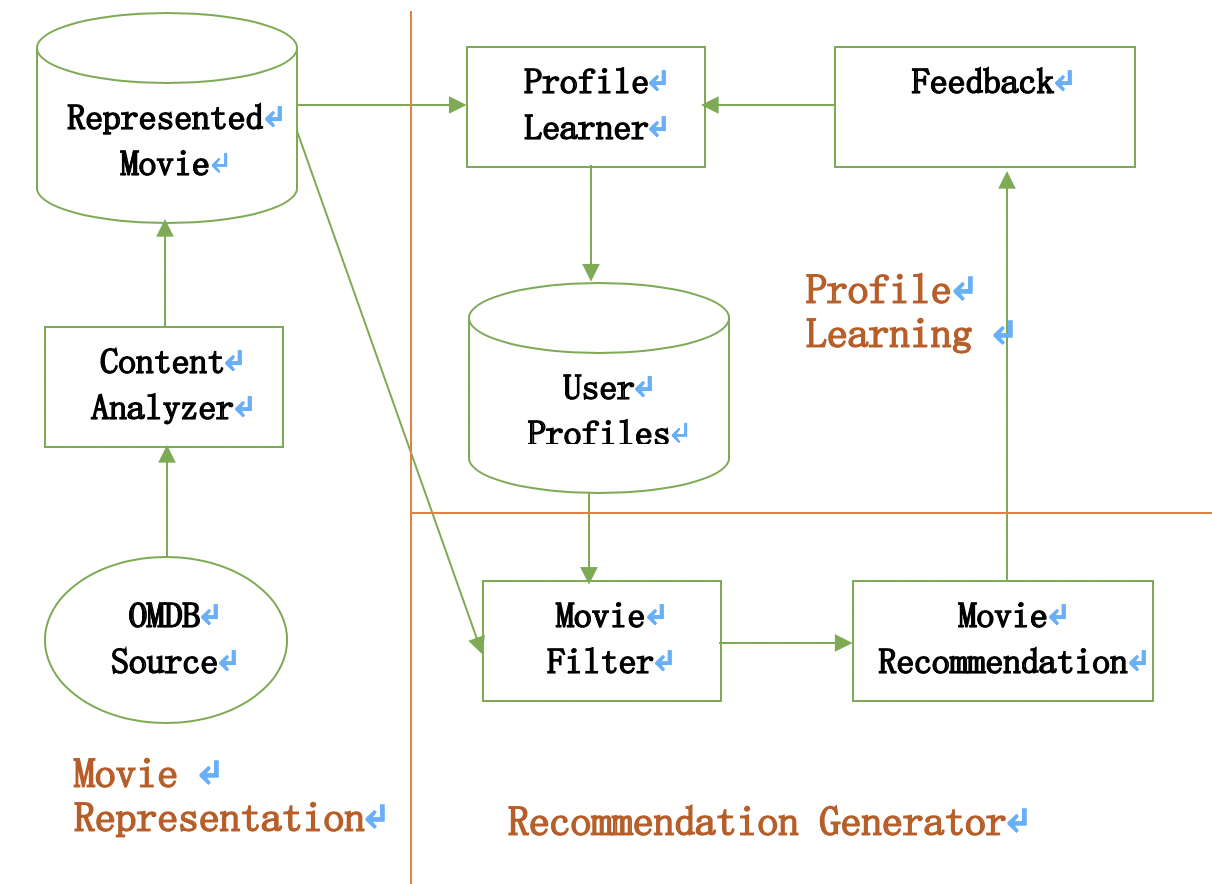
The aim of these 2 data processing is to make our project not just a demo, but can perform the functionalities in the real productive environment.

1. Model Design  
   We design a model class for both front end and back end that keep data consistency between transmission. The reason that we design this is it could speed up process of our development. And during the test period, we can easily figure out if the data between front and end is correct or not.
2. User Object Management  
   Our project has one user manager which is built through singleton pattern. As long as user login the app, we can get all user’s information at any time, and modify it.
3. Select another music/movie from the list after one selection has been processed

By the design of our function, fuzzy search will return a list of movies, and in music function we will return a list of songs/albums, after that user can select one from the list. We would like to implement that after the user made a choose, he can still make another choose without doing the query again, here’s the challenge occur. Based on the dialogflow we designed, those list selection response like “3” or “the third one” will consider as the “selection” intent only after the following process was done: movie/music querying --> movie/music list provided -> selection. Any other sentence will not consider as “selection” even it looks like one. The challenge here is that we don’t know what the user want if he didn’t go through the process. For example, by querying “3”, he may want to choose another move with index of 3, but it may also means he want the information about a movie named “3”. We can’t figure a way to solve this conflict so we didn’t implement this function

1. Movie Recommendation

For recommend movie function, there are 2 types recommendation systems, one is recommended by movies’ hotness at current time and the other is recommended in a personalized way.

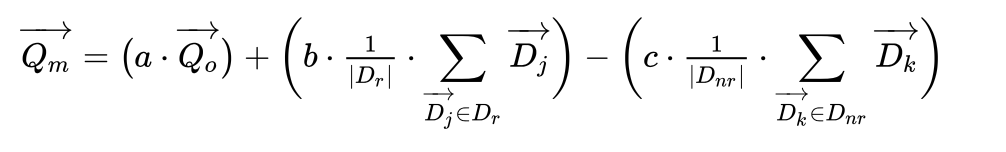


Movie source data is acquired from OMDB. Then we extract the useful description for each movie source data and store them with a structured movie data representation. Base on the movie content user interact with, user's preference can be obtained, updated and stored easily. For personalized recommendation, we compare the candidate movies and this user profile to find the best-matching movies for recommendation. For hotness recommendation, we collect all the users’ interaction with the movie and give the hottest movies.

* 1. Relevance Feedback

Rocchio algorithm is applied to classify the feedback from users' actions and learn user preferences. It can modify user’s profile in a short time and remarkable improve the precision rate and the recall rate.

The formula and variable definitions for Rocchio relevance feedback are as follows:

As demonstrated in the Rocchio formula, the associated weights (a, b, c) are responsible for shaping the modified vector in a direction closer, or farther away, from the original query, related documents, and non-related documents. In our system, when a is 1, b is 0.75 and c is 0.15 that the corresponding calculation result is reliable.

* 1. Migration of Interest

As the migration of interest occurs to user, we apply Newton’s Law of Cooling algorithm for simulating how the interest will drop over time.

In the formula, is current time, is last time, is the time interval since last, is cooling rate. We choose in this project.

1. Fuzzy search and ranking method

When user ask for a certain movie’s information and RelaxBot gives an unexpected result, we will do fuzzy search in our movie information database. First, the movie title will be tokenized into a list of terms. Then the candidate movies, which contain any of tokenized term, will be filtered out by doing regex search. Considering rare terms in movie name collection are more informative than frequent term, some stop words are removed from the term list and tf-idf weighting scheme is applied to calculate similarity between the given movie title and each candidate movie.

1. is given movie title
2. is candidate movie title

In this way, the fuzzy search results will be ranked by the and present top-k for user to select.

➢ ▪User documentation/manual: how to build, setup, configure, and use your system and functionalities. (For a joint group project, this section can be common for each group, if they merge all functionalities together in one code repositories for final submission; or otherwise, each group shall have a different description in this section.)

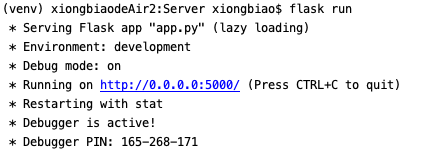
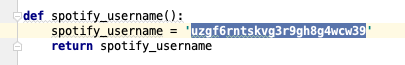
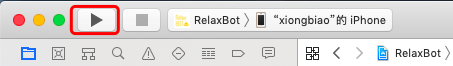
# User documentation

## Operating Environment

Front End: iPhone (iOS 11.0 +)

Back End: Python 3.6+ with flask, mongodb

## Deploy Configuration

1. Make sure you have a Python 3.6+ environment
2. install all packages: pip3 install requirements.txt
3. create virtualenv : virtualenv venv
4. activate virtualenv : source venv/bin/activate
5. back to Server dir: flask run  
   
6. Modify the spotify username and lifx bulb token to control your own light and music app.  
     
   
7. Download Xcode 10.2, update your iOS version to 12.2 Plug your phone on the MacBook.
8. Open the iOS/RelaxBot dictionary then click  
    
9. Modify the Server Address to your own local address in Utilities/Config.h and keep the port is same as server  
     
   
10. Select your device and click run to run the app on your phone  
    

➢ ▪Use either APA (https://student.unsw.edu.au/apa) or Harvard (https://student.unsw.edu.au/harvard-referencing) referencing style.

# Reference